EUROPEAN JOURNALS OF EMERGING COMPUTER VISION AND NATURAL LANGUAGE PROCESSING

Service Placement Strategies Across the Cloud-Fog-Edge Continuum: A Comprehensive Survey

John M. Pando
Department of Computer Science and Technology, University of Cambridge, United Kingdom

Gianna Trentini
Department of Information Engineering, University of Padua, Italy

V0LUME01 ISSUE01 (2024)
Published Date: 19 December 2024 // Page no.: - 60-78

ABSTRACT

The proliferation of Internet of Things (IoT) devices and latency-sensitive applications has driven a paradigm shift from centralized cloud computing to a more distributed continuum encompassing fog and edge computing. This integrated environment, often referred to as the Cloud-Fog-Edge continuum, brings computation and services closer to data sources, addressing critical concerns such as latency, bandwidth, and energy consumption. However, efficiently placing services within this heterogeneous and dynamic infrastructure is a complex, NP-hard problem. This survey provides a comprehensive review of service placement algorithms and strategies proposed for integrated Cloud-Fog-Edge environments. We categorize existing approaches based on their underlying methodologies, including optimization-based, heuristic and meta-heuristic, graph-based, and machine learning-driven techniques. Furthermore, we discuss key challenges, such as mobility, resource heterogeneity, multi-objective optimization, and the increasing adoption of microservices and containerization. Finally, we highlight open research directions and future trends to guide further advancements in this critical area.

Keywords: Service Placement, Cloud Computing, Fog Computing, Edge Computing, IoT, Distributed Systems, Optimization, Heuristics, Machine Learning, Microservices, Containerization.

INTRODUCTION

The rapid evolution of information technology has seen cloud computing emerge as a dominant paradigm, offering on-demand access to a shared pool of configurable computing resources [65]. This centralized model, characterized by vast data centers, has traditionally provided scalable and flexible solutions for various computational needs. However, with the exponential growth of Internet of Things (IoT) devices, projected to reach 100 billion by 2030 [4, 60], and the emergence of latency-sensitive applications (e.g., augmented reality, autonomous vehicles, smart cities), the traditional centralized cloud model faces significant challenges. These challenges primarily relate to high latency due to geographical distance, bandwidth limitations, and network congestion caused by the massive data generated at the edge [22, 58].

To address these critical issues, new computing paradigms, namely fog computing and edge computing, have emerged, extending the cloud's capabilities closer to the data sources [29, 72]. Fog computing acts as an intermediary layer between the edge devices and the distant cloud data centers, providing localized computation, storage, and networking services. This distributed nature allows for reduced data transfer to the cloud, leading to lower latency and bandwidth usage [11]. Edge computing, on the other hand, refers to processing data at or near the source of data generation, directly on edge devices or nearby edge servers, offering the lowest possible latency for real-time applications [29].

The seamless integration of cloud, fog, and edge computing forms a cohesive continuum, enabling flexible and efficient resource utilization across diverse geographical locations and computational capabilities. This distributed architecture offers significant benefits, including reduced latency, decreased bandwidth consumption, improved security, and enhanced energy efficiency, which is becoming increasingly important given the environmental impact of growing digital infrastructure [4, 29]. The energy required for connected objects is estimated to exceed global energy production by 2040, highlighting the urgent need for sustainable computing solutions [4, 24].

A critical and complex challenge within this integrated Cloud-Fog-Edge environment is the efficient placement of services and applications [15, 94, 102]. Service placement, also known as application placement or task offloading, involves deciding where to deploy various components of an application (e.g., virtual machines, containers, microservices) across the available cloud, fog, and edge

nodes. The goal is to meet specific Quality of Service (QoS) requirements, such as minimizing latency, throughput, energy consumption, and cost, while maximizing resource utilization and ensuring reliability [3, 15, 36, 110]. The problem is further compounded by the dynamic nature of these environments, including device mobility [59, 77], fluctuating resource availability, and varying application demands. The inherent complexity of service placement problems often renders them NP-hard [42], necessitating the development of sophisticated algorithms that can provide near-optimal solutions within acceptable timeframes.

The increasing adoption of microservices architecture further complicates service placement. Microservices are small, independent services that communicate with each other, offering significant benefits such as improved scalability, maintainability, reliability, reusability, and faster response times compared to monolithic applications [16, 28, 52]. The ease of implementing such architectures is facilitated by containerization technologies, which require fewer resources than virtual machines and enhance application management and orchestration [19, 78]. However, deploying and orchestrating these fine-grained services across a distributed continuum requires careful consideration of inter-service dependencies, resource requirements, and network topology to minimize communication overhead and optimize overall system performance [79, 81].

Several existing surveys have addressed aspects of service placement in cloud, fog, or edge environments individually [45, 75, 94, 102]. However, a comprehensive review focusing specifically on service placement algorithms within the integrated Cloud-Fog-Edge continuum, considering the intricate interplay between these hierarchical layers and the diverse algorithmic approaches, is still needed. This survey aims to fill this gap by providing a structured overview of the state-of-the-art service placement algorithms, categorizing them based on their methodological foundations, and discussing the challenges and future research directions in this evolving landscape. We also emphasize the growing importance of green computing and the need for energy-efficient placement strategies.

The remainder of this article is structured as follows: Section 2 outlines the methodology used for this survey. Section 3 provides essential background on the Service Placement Problem (SPP) architecture, including environmental considerations, application types, placement modes, and criteria. Section 4 delves into problem modeling and exact approaches. Sections 5, 6, and 7 present a detailed taxonomy and analysis of graph-based, heuristic, and machine learning solutions, respectively. Section 8 offers a comprehensive analysis and discussion of the surveyed works, highlighting trends, limitations, and the prioritization of various criteria. Section 9 identifies open research challenges and future directions. Finally, Section 10 concludes the survey. An Appendix is also provided for a list of important acronyms used throughout the survey.

METHODOLOGY

This survey was conducted using a systematic literature review approach to ensure comprehensive coverage and minimize bias. The methodology largely aligns with established guidelines for systematic reviews, such as PRISMA (Preferred Reporting Items for Systematic Reviews and Meta-Analyses) [41, 70], adapted for the scope of a survey article. This structured approach helps in identifying, selecting, and critically appraising relevant research, thereby providing a robust foundation for the survey's findings.

2.1 Search Strategy

The literature search was primarily performed on major academic databases and digital libraries, including IEEE Xplore, ACM Digital Library, SpringerLink, ScienceDirect, and Google Scholar. These platforms were chosen for their extensive coverage of computer science, networking, and distributed systems literature. The search queries combined keywords related to the computing paradigms and the core problem, using boolean operators to refine the search:

- Core Problem Keywords: "service placement" OR "application placement" OR "task offloading" OR "service migration"
- Computing Environment Keywords: "cloud-fog" OR "cloud-edge" OR "fog-edge" OR "cloud-fog-edge continuum" OR "edge computing" OR "fog computing"
- Algorithmic Approach Keywords: "algorithms" OR "optimization" OR "heuristic" OR "meta-heuristic" OR "machine learning" OR "reinforcement learning" OR "graph-based"

The search focused on publications from 2015 to 2024, given the relatively recent emergence and widespread adoption of fog and edge computing concepts. However, foundational works pre-dating this period were also considered if they significantly contributed to the understanding of service placement problems relevant to the continuum. The initial search focused on titles and abstracts to quickly filter out less relevant papers.

2.2 Selection Criteria

To ensure the relevance and quality of the included literature, strict inclusion and exclusion criteria were applied during the screening process:

Inclusion Criteria:

- Peer-reviewed journal articles, conference papers, and book chapters.
- Publications explicitly addressing service placement, deployment, or offloading in integrated cloud-fog-edge environments. This includes studies that consider the interaction and hierarchy between these

layers.

- Works proposing or analyzing specific algorithms, models, or frameworks for service placement, with a focus on their methodological details.
- Articles written exclusively in English.

Exclusion Criteria:

- Publications solely focused on cloud-only or edge-only environments without considering the continuum.
- Articles that do not propose or analyze specific placement algorithms (e.g., general surveys without algorithmic details, conceptual papers without technical contributions).
- Short papers, posters, workshop abstracts, or extended abstracts lacking substantial technical content and detailed methodology.
- Duplicate publications across different databases or different versions of the same paper (e.g., workshop version and journal version, prioritizing the most complete and recent version).

2.3 Data Extraction and Categorization

From the selected papers, relevant information was systematically extracted to facilitate analysis and categorization. The extracted data points included:

- The proposed algorithm/approach: Detailed description of the core technique used (e.g., Genetic Algorithm, Q-learning, Graph Partitioning).
- The computing environment: Specification of the architectural layers considered (e.g., Cloud, Fog, Edge, Cloud-Fog, Fog-Edge, Cloud-Fog-Edge).
- The primary objective(s) of the placement: Explicitly stated goals of the proposed solution (e.g., minimizing latency, energy consumption, cost; maximizing throughput, resource utilization, reliability, Quality of Experience (QoE)).
- The type of service: The granularity of the application component being placed (e.g., Virtual Machine (VM), container, microservice, Virtual Network Function (VNF)).
- The evaluation methodology: How the proposed solution was tested (e.g., simulation, testbed, theoretical analysis, real-world deployment).
- Key findings and performance metrics: Quantitative or qualitative results demonstrating the effectiveness of the approach.

Based on the underlying principles and techniques employed, the extracted algorithms were then categorized into a comprehensive taxonomy. This categorization forms the basis of the results section, allowing for a structured analysis of the state-of-the-art and a clear comparison of different algorithmic paradigms. The process involved iterative refinement of categories as more papers were reviewed, ensuring that the taxonomy accurately reflects the diversity of approaches in the literature.

Background on SPP Architecture

The Service Placement Problem (SPP) in integrated Cloud-Fog-Edge (iCFE) environments is a multifaceted challenge influenced by various architectural and operational factors. To understand the complexities of SPP, it is crucial to first establish a foundational understanding of the underlying infrastructure patterns, the types of applications being deployed, the different modes of placement, and the diverse criteria that drive placement decisions. This section elaborates on these foundational elements, as illustrated in Figure 1 of the accompanying PDF, which provides a useful taxonomy of SPP dependencies.

3.1 Environment and Infrastructure Pattern

The modern computing infrastructure is typically structured in a hierarchical manner, comprising three distinct yet interconnected layers: Cloud, Fog, and Edge. Service placement strategies are highly dependent on the characteristics and capabilities of each of these layers.

- Cloud: As the furthest tier from end-users, the Cloud represents a centralized environment composed of vast physical resources, such as large data centers and powerful servers [65]. It offers virtually unlimited computing, storage, and networking capabilities, ensuring high levels of performance and scalability. However, the geographical distance between cloud data centers and edge devices often results in higher latency and response times, making it unsuitable for applications requiring real-time processing [58]. The Cloud primarily serves as a repository for large-scale data analytics, long-term storage, and non-latency-critical applications.
- Fog: Positioned as the middle tier, Fog computing extends the cloud's capabilities closer to the edge of the network. It is geographically distributed, hierarchical, and decentralized, providing localized computing, storage, and networking resources to users [22]. Fog nodes can vary significantly in their availability and resource capacities, ranging from powerful micro-data centers to industrial controllers and routers. These devices can be organized into clusters or cells based on their location and capabilities, with higher-level fog nodes possessing greater resources suitable for more demanding components [100]. Increasingly, the distinction between Cloud and Fog is blurring, with many integrated Cloud-Fog frameworks treating the Cloud as the highest hierarchical level of the Fog, fostering a seamless continuum [72]. This survey adopts this integrated perspective to address the challenges of SPP.
- Edge: This tier is the closest to the end-users and data sources, aiming to minimize network response time and bandwidth overhead. Edge devices are geographically

distributed and typically have limited processing capabilities, embedded in sensors, wearable devices, smart appliances, and other measurement and computing units [29]. They are ideal for initial data processing, filtering, and real-time decision-making that requires ultra-low latency.

The efficient distribution of application sub-tasks—including monitoring, analyzing, executing, and planning—across these hierarchical layers is crucial for optimizing performance, reducing latency, and enhancing scalability [11, 58].

3.2 Types of Applications

The structure and characteristics of applications significantly influence service placement decisions. According to Salaht et al. [94], IoT-related applications can be broadly classified into three main structural groups:

- Monolithic Applications: These represent complex applications where several closely coupled functions are encompassed within a single, cohesive component. If such a monolithic service is stopped, all its functionalities are affected. Deploying a monolithic application requires placing the entire single component on a single physical node. While simpler in deployment for isolated cases, their rigidity makes them less suitable for dynamic, distributed environments where flexibility and fault tolerance are paramount.

- Inter-dependent Services: This category refers to applications that are divided into a set of services, with each service responsible for a particular task. Such applications are typically developed with an enterprise scope, and individual services cannot be put into production independently, similar to traditional Service-Oriented Architectures (SOA). The dependencies between these services necessitate careful co-location or optimized communication paths to minimize latency and ensure overall application performance.

- Independent Services (Microservices): In this model, applications are decomposed into small, independent modules, each responsible for a specific task. These modules communicate with each other, typically through well-defined APIs, to fulfill user requests. Lewis and Fowler [52] define microservices as a functional decomposition driven by logical domain and business design. Each service is autonomous, meaning that changing the implementation of one service does not affect others. Microservices applications and their interactions can often be represented as a connected directed acyclic graph (DAG), where vertices model individual services and edges represent the communications or interdependencies between them. This granular structure, while offering flexibility and scalability, introduces significant complexity in service placement due to the sheer number of components and their intricate communication patterns.

Service placement decisions are thus influenced by both the specific resource requirements of individual services (e.g., CPU, memory, storage) and the capacities of the available nodes. For multi-component applications, especially those built on microservices, the volume and nature of inter-service communication play a crucial role, often necessitating the co-location of highly communicative services to minimize latency and enhance overall system performance.

3.3 Placement Mode

The mode of service placement adapts to the dynamism and resource availability of the network, varying significantly from the static, centralized nature often associated with traditional cloud deployments to more dynamic and decentralized approaches prevalent in fog and edge environments.

- Centralized Placement: In this mode, all information regarding environment resources and application services must be known in advance by a central orchestrator. While this approach can theoretically provide an optimal solution from a global resource perspective [8], it is rarely considered practical in large-scale IoT contexts due to excessive latency and bandwidth limitations associated with collecting and processing all information at a single point [114].

- Decentralized Placement: This mode offers greater flexibility by enabling local optimization decisions based on each node's resources and data usage. Decisions are made independently of the total number of devices, allowing for dynamic management of system model changes. While highly powerful locally, decentralized solutions generally do not guarantee a globally optimal solution [38].

- Static Placement: This assumes an a priori known state of the infrastructure, where the assignment of services to nodes is decided upfront and remains fixed. This mode does not account for node mobility or changes in network conditions and resource capacities [54].

- Dynamic Placement: In contrast to static placement, dynamic placement can effectively deal with the perpetual changes in devices, application structures, and resource capacities. This is particularly efficient in Fog/Edge environments, where adaptive configurations manage the heterogeneous and distributed environment and, in some cases, predict future requirements [77].

- Reactive Service Placement: In a reactive scenario, service migration occurs only when run-time nodes become overloaded or no longer possess sufficient resources to handle new user requests. The goal is to alleviate processing burdens and restore system performance after a problem has manifested [59].

- Proactive Service Placement: This advanced mode anticipates the interdependence of services, user mobility, and network dynamism. Initial placement and subsequent migrations are planned to foresee and mitigate future

problems before they impact performance. This approach aims to prevent issues rather than reacting to them [90].

3.4 Placement Criteria

The algorithms surveyed in this article attempt to optimize service placement according to various objectives, reflecting the diverse priorities of different applications and system operators. The criteria most commonly used in the literature include:

- Resources: This encompasses metrics used to evaluate resource usage and requirements, including CPU utilization, memory consumption, storage capacity, and network bandwidth. It also represents the correlation between resource use patterns and other performance measures [53]. Efficient resource allocation is crucial for maximizing infrastructure utilization and avoiding bottlenecks.

- Cost: This criterion represents any financial costs associated with the SPP, including service deployment costs, infrastructure operational costs, and data transmission costs. Costs can vary significantly depending on the service provider, geographical location, and resource consumption [77]. Minimizing operational costs is a key objective for service providers.

- Energy and Greenhouse/CO2 Gas Emissions: This refers to the total energy consumption and associated greenhouse gas emissions of the entire system. It depends on the number and types of servers used, the resource utilization within each server, and the overhead energy consumption of communication between servers and nodes. Additionally, CO2 emissions are influenced not only by the amount of energy consumed but also by the sources of that energy, as different energy sources have varying levels of carbon intensity. This criterion is gaining increasing importance due to environmental concerns [4].

- Latency: Representing the time required for a data packet to travel from a source to a destination in the network, latency is a crucial criterion for delay-sensitive applications. The main components affecting latency include transmission media, propagation delays, router processing times, and storage delays. Minimizing end-to-end latency is often a primary objective for real-time IoT applications [61].

- Quality of Service (QoS): QoS refers to the level of user satisfaction and performance achieved by placing services on appropriate nodes in a distributed environment. It is a broad metric that can encompass various factors, including latency, throughput, reliability, and resource availability. Ensuring high QoS is paramount for delivering a satisfactory user experience.

- Performance: This is a comprehensive criterion that includes all other parameters considered for the SPP, such as violations of Service-Level Agreements (SLAs), congestion probability, and Quality of Experience (QoE). QoE, in particular, is a user-centric metric that captures the overall subjective experience of an application or service [62].

In addition to service placement, other critical concepts such as task scheduling and offloading are actively studied to enhance computing environments. While SPP focuses on the strategic deployment of application components to ensure optimal QoS, scheduling involves the real-time allocation and sequencing of tasks based on available resources and current workloads, ensuring efficient execution [6]. Offloading, however, refers to the delegation of computational tasks to edge servers to optimize processing efficiency and reduce latency, particularly for mobile devices [55]. Thus, SPP sets the foundational deployment of services, whereas scheduling and offloading manage the dynamic execution of tasks within that deployed framework. These concepts are often intertwined and require a holistic approach for optimal system management.

Problem Modeling and Exact Approaches

The Service Placement Problem (SPP) in integrated Cloud-Fog-Edge (iCFE) environments is fundamentally a complex optimization challenge. This section formalizes the problem, discusses its inherent computational difficulty, and reviews exact solution approaches, highlighting their applicability and limitations.

4.1 Problem Formulation

The SPP can be formally defined as an optimization problem where the goal is to map a set of application components to a set of available computing nodes within the iCFE infrastructure, subject to various constraints and aiming to optimize one or more objectives. Using notations similar to those found in the literature [61], we consider the following sets:

- $H=\{a1,a2,...,ai\}$: The set of applications to be placed.

- $S=\{s1,s2,...,sj\}$: The set of services or components, where each application $a \in H$ consists of a subset of these services. In microservices architectures, these sj are fine-grained, independent modules.

- $N=\{n1,n2,...,nk\}$: The set of available computing nodes within the iCFE infrastructure. These nodes represent data centers, servers, Fog nodes, edge devices, and so on, each with specific resource capacities.

- $Q=\{q1,q2,...,ql\}$: The set of requests, where each request $q \in Q$ has a source, a destination, and associated parameters or constraints (e.g., a maximum allowable latency).

Each service/component $s \in S$ of each application $a \in H$ must be placed and executed on one node $n \in N$ to respond to the incoming requests $q \in Q$. Conversely, each node n can execute several application components up to its capacity limits. The core of the problem lies in determining the optimal assignment of services to nodes.

The generic formulation of the optimization problem can be expressed as:

- Mono-objective Optimization:

$x \in X \min F(x) S.t.\ C1, C2, C3, ..., Cn.$

Here, X represents the set of all feasible solutions (i.e., valid service placements), F(x) is a single objective function that evaluates the quality of a given placement $x \in X$, and Ci are additional constraints that must be satisfied. For instance, if the primary objective is to minimize latency, as explored in Reference [61], the objective function could be $F(x) = \sum_{q \in Q} r(q,x)$, where r(q,x) quantifies the difference between the required latency and the achieved latency for a request q, given the placement x. Solving this problem aims to minimize the total deviation from desired latencies across all requests.

- Multi-objective Optimization:

$\{\min_{x \in X} (F1(x), F2(x), ..., Fp(x)), s.t. c1, c2, c3, ..., cn,$

In real-world scenarios, SPP often involves multiple, conflicting objectives. For example, F1(x) might represent latency deviation, F2(x) energy consumption, and F3(x) network costs. Since it is generally not possible to find a single placement x that simultaneously minimizes all Fi functions, multi-objective optimization approaches are employed [66]. One common strategy is to search for Pareto optimal solutions—placements where no objective can be improved without degrading at least one other objective. Another classical approach is scalarization, which transforms the multi-objective problem into a single-objective one, often using a linear combination: $F(x) = \sum_{i=1...p} wiFi(x)$, where wi are weights assigned to each objective to reflect their relative importance.

The placement problem is also subject to several key constraints that reflect the physical and operational realities of the iCFE environment. Typical constraints found in the literature include:

- Computing Resource Constraints [39]: Each node $n \in N$ has limited resources (e.g., CPU, memory, storage). The total resource requirement of all services placed on a node cannot exceed its capacity:

$$\sum_{q \in Q} \sum_{s \in S} xsnq \cdot Rs \leq Rn \quad \forall n \in N$$

where xsnq is a binary variable (1 if service s is placed on node n for query q, 0 otherwise), Rs is the resource requirement of service s, and Rn is the total resource available on node n.

- Latency Constraints [61]: The sum of latencies incurred for a given request must not exceed a predefined limit:

$$\sum_{(n1,n2) \in N \times N} an1n2q \cdot \Delta n1n2 \leq \Lambda q \quad \forall q \in Q$$

where an1n2q is a binary variable indicating that nodes n1 and n2 are involved in processing request q, Δn1n2 is the latency between nodes n1 and n2, and Λq is the maximum allowable latency for request q.

- Limitation on the Number of Queries [1]: Each service s on each node n can serve at most t queries:

$$\sum_{q \in Q} xsnq \leq t \quad \forall s \in S, \forall n \in N$$

where xsnq is 1 if service s is placed on node n for query q, and 0 otherwise.

The service placement problem, particularly in its multi-criteria and discrete nature (combinatorial optimization), is inherently NP-hard [40, 42]. This means that the computational time required to find an optimal solution grows exponentially with the size of the problem instance, making it infeasible for large-scale, real-world deployments.

4.2 Exact Solutions

Despite the NP-hard nature of the SPP, some researchers have proposed exact mathematical models, primarily based on Integer Linear Programming (ILP) or Mixed Integer Linear Programming (MILP). These formulations aim to provide optimal solutions by precisely defining the problem as a set of linear equations and inequalities.

- ILP and MILP Formulations: ILP formulations handle problems with linear objective functions and discrete (integer) variables, while MILP formulations allow for both discrete and continuous variables. Both are NP-hard problems [44, 94] that can be solved using techniques like the simplex method or Branch and Bound (B&B) algorithms. B&B systematically explores the solution space, pruning branches that cannot lead to an optimal solution. However, even with B&B, the search space for complex combinatorial problems is often too vast.

  - VNF Placement and Routing: Addis et al. [1] formulated two ILP problems to address the service function chain of Virtual Network Functions (VNFs), aiming to minimize the number of service instances. Their modeling strategy involved splitting each demand path into sub-paths and installing service instances at articulation points of biconnected components. This model, solved with CPLEX using SNDlib network data, focused on a single VNF type.

  - Network Resource Optimization: Gupta et al. [39] developed an ILP model for service function chains to optimize network-resource consumption. Their findings indicated that near-optimal resource consumption could be achieved by selectively upgrading nodes to support VNFs and carefully managing core allocation and traffic.

  - Cloud Service Modeling: Espling et al. [30] utilized graph structures to represent service specifications, component relationships, and placement constraints. They formulated an ILP to minimize the total service affinity placement of VMs across multiple nodes, subject to various

constraints. However, their evaluation was limited to a very small testbed.

○ Ultra-low Latency Microservices: Magnouche et al. [61] proposed an optimization model for service function chains, specifically targeting ultra-low latency for microservices.

● Multi-objective Optimization with Scalarization: Liu et al. [56] addressed the multi-objective problem of mobile device (MD) placement, balancing energy consumption, response time, and cost. They employed three queuing models for MD, Fog, and Cloud data centers, considering data rate and wireless link power consumption. The multi-objective problem was converted into a single-objective one using linear scalarization, which was then solved using the Interior Point Method (IPM). Their results showed a trade-off: energy consumption decreased with increased offloading probability, but execution time increased.

Limitations of Exact Approaches:

While exact methods guarantee optimal solutions, their high computational complexity makes them impractical for large-scale, real-world iCFE environments. The exponential growth of the solution space means that even with sophisticated solvers, these methods can only handle small problem instances. This limitation has driven the research community to focus predominantly on approximate algorithms, which can provide near-optimal solutions within reasonable timeframes, as discussed in the subsequent sections.

4.3 SPP Approaches Categorization

Given the computational intractability of exact solutions for most real-world SPP instances, the majority of research has focused on approximate algorithms. Rather than categorizing solutions solely by the architecture (Cloud, Fog, Edge) they target, this survey adopts a classification based on the fundamental algorithmic approaches employed. This choice is motivated by the observation that these methods are often generic enough to be adapted across different iCFE contexts, even if their initial application might be limited. For instance, metaheuristics offer robust exploration of vast search spaces, while graph-based methods leverage network topology for efficient placement.

We have grouped the proposed solutions for the SPP into three main families, based on their underlying algorithmic principles, as depicted in Figure 3 of the accompanying PDF:

● Graph-based Approaches: Graphs are ubiquitous in modeling service placement problems. They are used to represent the physical network of resources (nodes and links) and the communication dependencies between application components (e.g., microservices as Directed Acyclic Graphs or DAGs). Consequently, many algorithmic solutions leverage methods from graph theory or complex networks, such as graph partitioning, graph traversal, or matrix-based solutions using adjacency matrices. These approaches often originate from the graph theory or complex network communities.

● Heuristic and Meta-heuristic Solutions: Since exact solutions are impractical for most cases, a significant portion of the literature focuses on approximate methods. This category includes heuristics specifically designed for SPP, as well as tailored implementations of traditional meta-heuristics (primarily population-based algorithms like Genetic Algorithms, Ant Colony Optimization, Particle Swarm Optimization). These methods are often developed by the operational research community, aiming to guide the exploration of the search space to find local or global optima efficiently.

● Machine Learning (ML) Algorithms: The dynamic and data-rich nature of iCFE environments makes them well-suited for machine learning techniques. This category includes approaches based on supervised learning, reinforcement learning, and neural networks (including deep learning). These solutions leverage historical data to train models that can make predictive or optimized decisions for service placement, often originating from the data mining and machine learning communities.

It is important to note that while we propose a strict categorization, many solutions in the literature combine principles from different families. In such cases, we classify the proposal based on its primary or most significant algorithmic contribution. These families, while distinct in their core methodologies, are not mutually exclusive and often complement each other in hybrid approaches. The following sections delve into each of these categories in detail, discussing their strengths, weaknesses, and specific applications within the SPP domain.

Graph-based Approaches

Graphs provide a powerful and intuitive abstraction for modeling the complex relationships inherent in service placement problems within the Cloud-Fog-Edge continuum. In this context, computing resources (data centers, fog nodes, edge devices) are typically represented as nodes, and network links between them are represented as edges. Similarly, microservices applications can be modeled as Directed Acyclic Graphs (DAGs), where services are nodes and inter-service communications are weighted edges (representing cost, latency, or data volume). This section explores various graph-based solutions for SPP, categorized by their primary graph-theoretic technique. Table 3 in the accompanying PDF provides a summary of these approaches.

5.1 Graph Partitioning Techniques

Graph partitioning and community detection are processes of dividing a network graph into smaller, cohesive subgraphs or "communities," where nodes within a community are more densely connected to each

other than to nodes outside the community. These techniques are highly relevant to SPP as they can help in grouping services or resources that exhibit strong interactions or geographical proximity. Many off-the-shelf solutions exist for community detection [33], and researchers have adapted them for SPP. The general principle involves identifying these communities and then using heuristics to place services on the compute nodes within these identified communities.

- Community Detection for Dynamic Environments: Coimbra et al. [23] proposed an approach for service placement in community networks, involving two steps: a centralized community detection based on graph properties, followed by a decentralized election heuristic. This method allows for incremental processing of services and adapts to network variations and dynamism, demonstrating good performance in simulations.

- Availability-Aware Placement: Lera et al. [51] presented a two-phase placement approach focused on service availability and QoS. They first mapped and partitioned the graph of connections between Fog nodes using community detection. Subsequently, a first-fit decreasing approach was used to place applications within these communities based on execution deadlines. The second phase involved allocating application services to devices within a selected fog community by analyzing the transitive closure of the application graph. While effective, the community detection algorithm used [76] might not be the most efficient in terms of speed and quality compared to more recent advancements.

- Energy- and Resource-Aware Microservices Placement: Taleb et al. [105] introduced a new model and heuristic for microservices placement in the iCFE continuum, specifically designed to optimize energy consumption while adhering to resource constraints and ensuring acceptable response times. This approach leverages community detection to identify groups of densely connected network nodes. Following community identification, a greedy best-fit algorithm allocates microservices based on their resource requirements and the capabilities of the selected nodes. Experimental results indicate significant reductions in energy consumption while maintaining acceptable response times.

- VNF Placement and Migration in Data Centers: Zu et al. [122] investigated the placement and migration of Virtual Network Functions (VNFs) in data centers, focusing on the user's Service Function Chain (SFC). Their objective was to minimize long-term operational costs by formulating the problem as a dynamic programming model. They proposed a two-stage online heuristic algorithm, combining a greedy algorithm based on community detection (using the Louvain algorithm [12]) with an iterative migration algorithm. Simulations showed successful traffic prediction and good performance.

- Multilayer Resource-Aware Partitioning: Samani et al. [95] considered the heterogeneity of Fog computing devices, modeling Fog resources as a four-layered graph where each layer represents a resource type (network, CPU, memory, storage). Each layer is partitioned using the Louvain community detection algorithm to group similar nodes. A "compressed" graph is then constructed, with nodes representing these groups and edges representing inter-community connections. This compressed graph is further split into disjoint clusters of similar Fog device resources, onto which applications are placed according to their requirements. This method demonstrated reduced resource usage and improved adherence to application request deadlines compared to existing methods.

- K-way Partitioning for IoT Data Placement: Naas et al. [73] presented a heuristic for IoT data placement in Fog environments, aiming to decrease data access time. They modeled the physical infrastructure as an undirected weighted graph, where vertex weights represent data storage needs and edge weights represent data flow. Instead of traditional community detection, they divided the graph into k subgraphs using the Metis k-way partitioning method [48]. This process ensures an even distribution of vertex weights among sub-graphs and minimizes the sum of cut edge weights, showing proximity to optimal solutions for a fixed number of partitions.

- Clustering for Micro-cloud Service Deployment: Selimi et al. [98] proposed an approach to reduce the complexity of service deployment in community micro-clouds by leveraging network state information. Their heuristic algorithm, "Bandwidth and Availability-aware Service Placement," adapts to changing compute node topologies. It involves three parts: K-means clustering to check node availability, aggregation to find cluster heads maximizing bandwidth, and cluster recompilation and placement. Experimental results showed a significant improvement in bandwidth gain compared to random placement strategies.

5.2 Graph Traversal Solutions

Another category of graph-based approaches models the problem using graphs but primarily relies on graph traversal algorithms to find placement solutions.

- Latency-Optimized Microservice Placement: Wang et al. [113] introduced a latency estimator for Microservice Placement in Edge-Cloud Collaborative Smart Manufacturing (MPCSM) to minimize end-to-end latency while respecting resource and location constraints. They modeled the system as a directed graph of devices and communication links, representing applications as DAGs with microservice dependencies. Services are sorted by latency sensitivity, and Breadth-first Search (BFS) is used to assign placement orders. Microservices are then placed on edge devices with the lowest latency that meet the specified constraints. While showing improved performance, the training process for this approach can be time-consuming and static, limiting dynamic adaptation.

- Energy and Carbon Emission-Efficient Placement: Ahvar et al. [2] proposed DECA (Dynamic Energy cost and Carbon emission-efficient Application) placement method for Edge Cloud computing. This method optimizes both initial service placement on Edge-Clouds (ECs) and manages migrations to re-optimize costs. It considers geographical distribution, energy costs of network equipment, and associated CO2 emissions. Each EC is represented as a weighted graph, and applications are sets of components with communication represented by a matrix. To tackle the optimization, the authors combine an A* traversal algorithm with a solution to select the best additional vertex at each step, balancing energy and cost metrics. Simulations demonstrated a good trade-off between these objectives.

5.3 Adjacency Matrix and Eigenvectors

Some graph-based approaches utilize the adjacency matrix of microservices calls, or variations thereof, combined with eigenvector computations or eigendecompositions to determine service assignments.

- Rank-based Matrix Optimization for Green Computing: Saboor et al. [93] developed a rank matrix optimization technique for dynamic provisioning of containerized microservices placement in a Cloud environment. They construct a stochastic matrix from the adjacency matrix of microservices call graphs. If a microservice Mi calls other microservices {O1,...,ON}, it passes 1/N of its value to each Oj. This iterative process of value distribution converges to an eigenvector associated with eigenvalue 1, yielding ranks for microservices. Microservices with the highest rank are grouped into a container and deployed on data centers with the highest Green Energy Index (optimized for carbon emission and renewable energy use). However, this solution's relevance diminishes for non-strongly connected graphs (e.g., DAGs), as sources (uncalled MS) will have the lowest rank and sinks (no outgoing calls) the highest.

- Eigendecomposition for Service Function Chains: Mechtri et al. [64] combined greedy and ILP models for service placement, proposing a novel eigendecomposition of the adjacency matrix using an analytical approach. This method efficiently matches weighted VNF graphs to the infrastructure graph of nodes, extending Umeyama's [108] solution for weighted graph matching. They employed eigendecompositions of adjacency matrices and the Hungarian method to find a mapping function that minimizes a similarity distance metric between nodes. Since Umeyama's solution requires graphs of the same size and a one-to-one mapping, the authors provided a MILP formulation to improve the joint solution. A greedy algorithm based on bipartite graphs was used to benchmark the eigendecomposition algorithm's performance across various metrics like network property, system load, and request size.

Heuristic Solutions

Given the NP-hard nature of the Service Placement Problem (SPP), exact solutions are often computationally prohibitive for real-world, large-scale Cloud-Fog-Edge environments. Consequently, heuristic and meta-heuristic approaches are widely employed. A heuristic is a computational method that aims to solve an optimization problem faster than exhaustive techniques, typically by iteratively improving a candidate solution. While heuristics may get stuck in local optima, they provide good, near-optimal solutions within a reasonable time. Meta-heuristics, as defined by Blum and Roli [13], are "general algorithmic frameworks which can be applied to different optimization problems with relatively few modifications." They guide the exploration of the search space to find local or global optima more effectively than simple heuristics. This section presents various heuristic and meta-heuristic solutions adapted for the SPP, categorized by their underlying principles. Table 4 in the accompanying PDF provides a comprehensive list of these approaches.

6.1 Fuzzy Logic Algorithms

Fuzzy logic is a computational approach that deals with imprecision and uncertainty in decision-making, mimicking human-like reasoning. It converts input variables into fuzzy sets, applies a set of rules to combine them, and assigns degrees of truth to statements rather than binary true/false values [118]. This makes fuzzy logic suitable for complex environments where exact measurements or clear-cut rules are difficult to establish. Some SPP optimization solutions leverage fuzzy logic system rules to make informed decisions at each step.

- Combined Fuzzy Logic and ACO for VNF Optimization: Shokouhifar et al. [99] proposed a hybrid heuristic-metaheuristic algorithm combining a fuzzy logic system with Ant Colony Optimization (ACO) for VNF placement. This approach aims to leverage the speed of fuzzy-logic-based heuristics and the high quality of metaheuristics. Their multi-objective function includes total power consumption, total path reliability, and total end-to-end latency. The algorithm also adapts to changes in application requirements by readjusting objective function weights. Fuzzy logic is used for both server and link selection: a fuzzy score is calculated for servers based on distance, load, and available resources, and for links based on bandwidth, delay, and congestion. Ants then use these scores to select optimal servers and links, balancing solution quality and convergence speed.

- Fuzzy Logic for QoE Maximization: Mahmud et al. [62] introduced a fuzzy-logic-based algorithm combined with an ILP approach to allocate requested services across nodes. The primary objectives were to maximize user Quality of Experience (QoE) and ensure service access, resource consumption, and service delivery. Their method calculates application rating and Capacity Class Score (CCS) using various parameters mapped to fuzzy sets via

membership functions and fuzzy rules. This fuzzy logic approach considers multiple status parameters to determine the CCS, even for applications with relaxed expectation parameters. Evaluated in the iFogSim Fog-Cloud environment, the method demonstrated effectiveness, though it was based solely on the Fog architecture.

6.2 Greedy Algorithms

A greedy algorithm is a straightforward strategy that makes the locally optimal choice at each step, with the expectation that this will lead to a globally optimal or near-optimal solution. Its main advantages are simplicity of implementation and computational speed. However, greedy algorithms can often get trapped in local optima, failing to find the true global optimum.

- FOGPLAN Framework for QoS and Latency: Yousefpour et al. [116] proposed the FOGPLAN framework for dynamic Fog service provisioning, formulating the optimization problem as an Integer Non-linear Programming (INLP) to support low latency and QoS requirements. They introduced two greedy algorithms to solve the INLP. The first minimizes delay violation for IoT devices by prioritizing Fog nodes with higher incoming traffic and releasing services from low-traffic nodes if it doesn't cause violations. It can also deploy/release services on the Cloud if resources are available. The second greedy algorithm minimizes service deployment cost by sorting incoming traffic rates to Fog nodes and checking if deployment or release reduces costs and increases revenue.

- Resource Provisioning with Branch and Bound and First Fit: Rakshith et al. [89] developed a resource provisioning framework for IoT applications in Fog environments, aiming to reduce latency and resource usage. They combined a Branch and Bound (B&B) algorithm with a first-fit greedy algorithm. The first-fit approach processes items sequentially, assigning each to the first available bin with sufficient space; if no such bin exists, a new one is created. While B&B guarantees overall optimality, the first-fit heuristic ensures timely deployment. Their approach also analyzes provisioning time and required resources, with future work considering service scaling.

- DRACeo Simulator for Energy-Efficient Microservices: Valera et al. [109] introduced DRACeo, a simulator for microservices-based application deployments, designed to optimize energy consumption without degrading QoS. DRACeo schedules microservices based on environmental changes and dependencies. It employs classical heuristic algorithms for centralized and non-centralized adaptive planning. One heuristic determines QoS based on required and obtained hardware resources, while another greedy heuristic selects the microservice with the lowest QoS to limit overall application QoS, without considering broader system impact. The simulator allows users to specify QoS and energy consumption expectations for tracking.

- Dynamic Service Placement for Mobile Micro-clouds: Wang et al. [111] presented a solution based on linear problem modeling to minimize average cost over time. They proposed an offline algorithm to predict future instance placement costs based on user preferences within a time window, solvable by dynamic programming for optimal solutions. For increased complexity with multiple instances, they used an online approximate algorithm with polynomial time complexity to dynamically handle runtime instance placement. This algorithm calculates cost differences for each instance in available slots, selects the instance with the largest minimum difference, and places it in the slot with the smallest difference.

- Priority, Network, and Energy-Aware Placement (MinRE): Hassan et al. [43] proposed MInRE, an efficient policy solution addressing response time, energy consumption, and resource usage. It categorizes services into critical (requiring fast response, optimized by MinRes heuristic) and normal (energy reduction focus, optimized by MinEng heuristic). MinRes sorts critical services and Fog node delays, placing services on the main Fog cluster node with minimum delay. If capacity is insufficient, it seeks a neighbor node. MinEng sorts services by deadline, selecting nodes with less network delay and minimum energy consumption, following a similar neighbor strategy if resources are insufficient. Simulations showed improved delay and energy consumption, but the model did not include applications with multiple dependent services or cost resource nodes.

- QoS-aware Deployment with Fail-first: Brogi and Forti [14] proposed a model to improve QoS, developing placement algorithms for design, simulation, deployment, and execution in Fog environments. They reduce the search space by mapping software components only to feasible fog nodes and cloud data centers that meet all requirements, using a K-key-value map. A greedy fail-first algorithm then selects a single placement node. FogTorch, a Java tool simulator, was used for performance evaluation.

- Container Migration for Energy Reduction: Chhikara et al. [21] suggested an approach for placing and migrating service containers from overused to underused nodes to reduce energy consumption and resource utilization. Their algorithm classifies hosts as overloaded, underloaded, or balanced using k-means and hierarchical clustering to select migration targets. Containers causing overload (based on CPU/RAM) are migrated using a best-fit greedy approach to find the optimal destination host.

6.3 Population-based Heuristics

Population-based metaheuristics are inspired by natural selection and the collective behavior of groups (e.g., flocks of birds, swarms of bees, ant colonies, whale pods). These algorithms maintain a population of candidate solutions that evolve over iterations, exploring the search space to find optimal or near-optimal solutions. While they do not

guarantee global optimality, they are effective for complex problems with large search spaces.

- Genetic Algorithms (GAs): Inspired by natural selection and evolution [67], GAs maintain a population of chromosomes (candidate solutions) that evolve through operations like crossover and mutation. Fitter individuals are stochastically selected, ensuring the next generation is more optimized while maintaining diversity.

  - Service Placement in Fog: Skarlat et al. [101] used a GA for service placement in Fog environments, aiming to maximize services placed on Fog nodes and minimize network communication delays and resource usage. Chromosomes encode placement plans, with genes indicating service placement on specific fog resources and system state (CPU, RAM, storage). Crossovers and mutations combine or modify plans, and a fitness function evaluates constraint adherence (capacity, delay). Compared to greedy and exact methods, the GA showed improved response time.

  - Proactive Data Positioning: Canali and Lancellotti [17] developed a GA-based heuristic for proactively positioning data on Fog nodes, considering current load and sensor communication latency. Genes represent which Fog node receives data from a sensor. Evaluated on smart city scenarios, their approach yielded results equivalent to direct optimization.

  - Business-Driven IT Service Component Placement: Tortonesi et al. [107] proposed a GA optimization technique for business-driven IT service component placement in dynamic, distributed Cloud systems. The GA finds optimal IT service placement on Cloud data centers based on operational costs, calculated from provider fees. Experiments with a realistic two-tier scenario and real-life prices showed improved quality.

  - Multi-objective IoT Service Placement: Natesha and Guddeti [74] provided an elitism-based GA to solve a multi-objective placement problem, minimizing energy consumption, response time, and cost while ensuring QoS. Chromosomes represent service numbers, and genes represent Fog node numbers. Fitness values are calculated based on weighted service time, cost, and energy. Elitism selects best individuals, and crossover/mutation operations produce diverse offspring.

  - Comparison of Evolutionary Algorithms: Guerrero et al. [37] compared three GA-based evolutionary algorithms—Weighted Sum Genetic Algorithm (WSGA), Non-dominated Sorting Genetic Algorithm II (NSGA-II) [70], and Multi-objective Evolutionary Algorithm based on Decomposition (MOEA/D)—for optimizing network latency, service spread, and resource requirements in Fog service placement. WSGA is a single-objective GA using weighted sum transformation. NSGA-II is a multi-objective algorithm ordering solutions by dominance. MOEA/D decomposes the problem into multiple scalar fitness optimizations. Experiments on a random Barabasi-Albert network [10] showed varying performance based on objectives, with NSGA-II achieving highest goal optimizations and diversity, while MOEA/D was faster.

- Particle Swarm Optimization (PSO): PSO [63] is a computational method where a population of candidate solutions (particles) move in the search space, guided by their own best-found position and the global best-found position.

  - Microservices Placement for Resource and Energy Optimization: Yu et al. [117] designed a workload model for microservices placement as a bi-criteria optimization problem. They proposed a three-stage mathematical algorithm to optimize trade-off decisions. First, a multi-objective PSO meta-heuristic creates instances and calculates the Pareto front for optimized MS instances and thread allocation. Second, requests are routed to established microservice instances in a balanced way (similar to a bin packing problem). Finally, a load-balancing algorithm (LEGO) selects the minimum number of servers. Experiments showed better and faster performance than baselines.

  - Energy-Aware Edge Server Placement: Li and Wang [54] devised a PSO energy-aware placement method for Edge server devices to ensure low energy consumption. They adapted PSO for a discrete problem, improving basic PSO equations. The algorithm takes base station datasets, Edge server coverage radius, particle swarm size, and iterations as input, assigning base stations to Edge servers and updating particle positions/velocities to find an optimal placement. Evaluated on a real Shanghai base station dataset, it showed over 10% energy consumption reduction, but lacked dynamic optimization for changing demands.

  - QoS-aware Multi-objective Set-based PSO: Pallewatta et al. [80] provided a QoS-aware Multi-objective Set-based PSO for batch placement of microservice applications in Fog environments, defining QoS by budget, makespan, and throughput. This approach is combined with the Learning Particle Swarm Optimization (S-CLPSO) algorithm to improve convergence. It initializes a particle population, computes fitness, updates positions, and selects the best swarm position over iterations, returning optimal microservice-to-device mapping. Experiments demonstrated improved makespan satisfaction and budget.

  - Discrete Particle Swarm Optimization for IoT Services: Djemai et al. [26] approached service placement as an optimization problem to reduce energy consumption and minimize delays in IoT applications. Their method uses a discrete particle swarm optimization (DPSO) algorithm with real-valued speeds to identify efficient service locations. Physical infrastructure is a graph, and IoT applications are modeled as DAGs.

- Ant Colony Optimization (ACO): ACO algorithms

[27] are inspired by the foraging behavior of ant colonies, where ants find optimal paths by leaving pheromone trails. The optimization problem is transformed into finding the shortest path in a graph.

○ Multi Replicas Pareto ACO (MRPACO): Huang et al. [47] presented MRPACO to manage the geographic distribution of Fog nodes and multi-objective service replicas, including latency time and deployment cost. The algorithm creates multiple pheromone trails and uses single heuristic information. Ants decide the next data flow direction for scheduled source-service pairs. Experiments showed increased convergence speed and enhanced Pareto front accuracy, maximizing objective placement.

○ Modified ACO for VNF Placement and Routing: Farshin et al. [31] modified ACO to propose a chaotic grey-wolf-optimized (GWO) knowledge-based system for VNF placement and path allocation. The GWO algorithm mimics grey wolf hunting behavior [69]. The proposed framework performed well in placement and routing.

● Whale Optimization Algorithm (WOA): WOA [68] mimics the hunting strategy of humpback whales, who communicate and coordinate movements to encircle prey using bubble nets.

○ Cost-Efficient IoT Service Placement: Ghobaei-Arani and Shahidinejad [35] proposed a cost-efficient IoT service placement approach using WOA in a fog computing environment.

○ CREW Heuristic for Reliability and Cost: Martin et al. [83] introduced CREW, a heuristic method addressing conflicting criteria of maximizing service reliability while minimizing monetary cost. Due to the exponential solution space, CREW uses an Eagle strategy based on multi-Whale optimization for placement decisions. The Eagle strategy is a two-stage hybrid method combining random global and local search for stochastic optimization [115]. Simulations showed CREW outperforming existing multi-objective meta-heuristics like NSGA-II and MOWOA. However, the model only considered CPU and memory, not storage or network bandwidth.

○ Self-managing WOA for IoT Services: Ghobaei-Arani and Shahidinejad [35] outlined a self-managing approach using WOA for IoT services deployment across a three-tiered fog architecture, focusing on QoS for enhanced throughput.

● Cuckoo Search (CS): Mortazavi et al. [71] introduced a custom cuckoo search algorithm for service placement (CSA-SP) to optimize service positioning in fog nodes, minimizing energy consumption while considering data transfer constraints and resource availability.

6.4 SPP Specific Heuristics

Beyond the established meta-heuristic families, several heuristics have been specifically designed or tailored for the SPP, addressing unique aspects of the problem in iCFE environments.

● Markov Chain-based Service Placement: Carlini et al. [18] pioneered a Markov chain-based service placement method with point-to-point interactions and discrete cross-step algorithms to reduce energy, time, and cost. Simulations demonstrated convergence in a distributed setting, but the approach did not account for dynamic service arrival/completion.

● Distributed Placement with Markov Approximation: Kayal and Liebeherr [49] proposed a fully distributed placement strategy in Fog/Edge environments based on Markov approximation. This method optimizes communication costs between microservices and energy consumption. Microservices are initially placed randomly on Fog nodes and then moved to neighboring nodes according to Markov Chain transitions that approximate the optimization problem. Experimental results showed comparable solutions to existing centralized methods.

● Decentralized Microservices-based IoT Application Placement: Pallewatta et al. [79] developed a decentralized approach for microservices-based IoT application placement. Their main objective was to place latency- and bandwidth-critical microservices as close as possible to the data source using a heuristic placement algorithm that scales microservices in heterogeneous nodes. Each Fog layer is considered a cluster with a controller node managing microservices placement and load balancing. The controller uses three algorithms: selecting the closest Fog node (for latency/communication cost), selecting the Fog node with the lowest load (for resource balancing), and combining these results for a final placement decision.

Machine Learning Algorithms

The dynamic, complex, and data-intensive nature of Cloud-Fog-Edge environments makes Machine Learning (ML) algorithms particularly well-suited for optimizing service placement. These models can learn patterns from historical execution data to make proactive and adaptive placement decisions. This section categorizes ML approaches used for SPP into supervised learning, reinforcement learning, and neural/deep neural networks. Table 5 in the accompanying PDF summarizes these machine learning solutions.

7.1 Supervised Learning

Supervised learning involves training models on labeled datasets, where both inputs and corresponding desired outputs are provided. The trained model can then predict unknown outputs for new input data.

● Classification and Regression Tree for Task Offloading: Rahbari and Nickray [87] proposed a placement method based on the Classification and Regression Tree (CART) algorithm, named MPCA, for task

offloading in mobile fog computing. They utilized the power consumption of mobile devices for the training phase, incorporating decision parameters such as cost, integrity, availability, speed, and capacity for Fog node selection. They also introduced the MPMCP method to optimize MPCA by analyzing the probability of network resource utilization. The proposed algorithm demonstrated superior performance compared to the First Fit algorithm in terms of consumption, response time, and overall efficiency.

- Gradient Boosting Regression for Serverless Edge/Cloud Optimization: Das et al. [25] presented a framework for performance optimization in serverless Edge/Cloud environments using dynamic task placement, based on Gradient Boosting Regression. Their machine learning model was trained on three real-world application use cases (image resizing, face detection, and speech-to-text) to predict latency and cost. Evaluated using AWS, the framework achieved end-to-end latency prediction with less than 6% error. However, the model was trained on a relatively small sample of use cases, which might limit its generalization.

7.2 Reinforcement Learning

Reinforcement Learning (RL) is a classical machine learning paradigm where an agent learns to make sequential decisions by interacting with an environment through trial and error. The agent receives rewards or punishments as feedback for its actions, aiming to maximize cumulative reward over time [104]. For SPP optimization, various RL techniques, such as Q-Learning (QL) and State-Action-Reward-State-Action (SARSA), have been employed.

- Q-Learning for Dynamic Service Migration: Chen et al. [20] introduced a service migration mechanism in Edge Cognitive Computing to achieve higher energy efficiency and Quality of Experience (QoE). Their migration and placement strategies are based on RL, using a Q-learning algorithm to select optimal nodes for each service. Experimental results showed that this architecture improves QoE, particularly when user needs are not precisely predicted.

- Q-placement for SDN Service Placement: Zhang et al. [120] developed a RL-based algorithm called Q-placement for Software-Defined Networking (SDN) switches service placement. The primary objective was to minimize the average accumulated service costs for end-users while guaranteeing performance. This work demonstrated that the proposed method improves cost savings and outperforms classical service placement algorithms. A key advantage is its on-demand decision-making process, unlike traditional optimization algorithms with fixed optimization levels.

- Dyna-Q RL for Proactive Microservice Placement: Ray et al. [90] developed a RL-based proactive microservice placement and migration mechanism for Edge servers to adapt to user movement and requests. They modeled a linear workflow of microservices as a graph and applied a combination of QL and RL models called Dyna-Q RL [104]. Experiments conducted on a real-world San Francisco taxi dataset [85] showed that their method improved response time and reduced latency compared to existing literature.

- RL for Energy-Efficient Smart City Management: Reddy et al. [91] investigated an approach to optimize energy consumption and service delay in Fog computing for smart city infrastructure management while maintaining QoS. They introduced a reinforcement learning-based duty cycling approach to balance energy usage and QoS, where Fog nodes send state information to a prediction agent that updates its knowledge through rewards or penalties. Additionally, they developed the SCS-GA algorithm for virtual machine allocation, reducing service request failures and minimizing latency. Simulations using iFogSim demonstrated that combining SCS-GA with RL reduced migrations and energy consumption compared to using SCS-GA alone.

- RL for Elastic Container Deployment: Rossi et al. [92] presented a network-aware heuristic for container placement deployed on geo-distributed Clouds. They introduced an RL solution to control the elasticity of containers and then used a network adaptive heuristic to solve the linear programming problem. The approach improved QoS and resource utilization, though the sample used did not include highly diverse Cloud nodes.

7.3 Neural and Deep Neural Networks

Neural networks, and more specifically deep learning, are a subfield of machine learning inspired by the human brain's structure and function. These networks consist of multiple layers (dozens to thousands), where each layer is responsible for interpreting, extracting features, and training based on the output of the previous layer from input data. Through iterative adjustments based on "wrong" answers, the model is trained in a high-dimensional space to describe the input data as precisely as possible.

- Neural Network for Auto-scaling Prediction in 5G Networks: Subramanya et al. [103] proposed a machine learning model based on the Multi-Layer Perceptron (MLP) neural network (classifier and regressor) to enhance auto-scaling prediction for Network Function Virtualization (NFV) demands in 5G networks, based on traffic from a commercial real operator. The authors evaluated the models by examining the number of User Plane Function (UPF) instances to be processed. Both models proved efficient for auto-scaling prediction.

- Random Neural Networks and Cognitive Network Map for Fog Services: Fröhlich and Gelenbe [34] developed a combination of Random Neural Networks (RNN) and Cognitive Network Map (CNM) for optimal fog services placement in SDN IoT networks, aiming to optimize QoS and resource usage. Experimental results showed that RNN provided an effective solution for optimizing

parameters and ensuring good client service. Their combined solution adapted well to changes in both large and small networks, leading to efficient service placement.

- Deep Reinforcement Learning for Proactive Service Placement: Sami et al. [97] provided a deep reinforcement learning (DRL) approach based on scalable Markov Decision Process (MDP) for proactive service placement. The objective was to make placement decisions before user demands, thereby improving the Quality of Experience (QoE). They also presented an end-to-end architecture incorporating a service scheduler and a bootstrapper. Data from connected users, services, and source IPs were collected from applications to train the algorithm. Authors used real-life datasets extracted from the Google Cluster Trace 2011-2 dataset and NASA Server Logs.

- Deep Q-learning for Container Migration: Tang et al. [106] proposed a deep Q-learning algorithm to reduce power consumption cost and delay for container migration. They modeled the container strategy as a multi-dimensional MDP and used deep reinforcement learning to reduce the dimensionality of large MDP spaces, enabling faster decision-making. This study, conducted on a real-world data driver (San Francisco taxi traces [85]) application, demonstrated that the proposed approach enhanced delay, power consumption, and migration cost compared to baseline approaches.

- Reward Sharing Deep Q-Learning for Microservice Deployment: Lv et al. [57] addressed a multi-objective microservice deployment problem in Edge computing, aiming to minimize communications. They represented inter-microservice communication as an undirected, weighted interaction graph and applied a learning-based algorithm called Reward Sharing Deep Q-Learning (RSDQL). They also proposed a dynamic Elastic Scaling algorithm based on heuristics to improve scalability. Experiments conducted using Kubernetes showed shorter response times and better load balancing and scalability.

- Deep Learning for Joint Routing and Placement: Pham et al. [84] introduced a joint routing and placement problem that dynamically allocates resources based on workload demand to reduce long-term operational costs. This framework employs a deep learning module to mimic the Branch and Bound (B&B) algorithm while limiting the search space. The algorithm uses a decision neural network to classify nodes and a dataset of visited nodes for training. It iteratively selects a node, solves its relaxed problem, aggregates the dataset, and uses branching to explore the search space for an optimal solution. Simulation results showed that the method surpassed baselines in convergence and operational cost.

DISCUSSIONS

This section provides a comprehensive analysis of the surveyed works, focusing on the prevalence and effectiveness of different algorithmic approaches, the types of datasets utilized, the prioritization of various optimization criteria, and the evaluation tools employed. This discussion aims to identify current trends, highlight limitations, and pinpoint areas requiring further research in the context of Service Placement Problems (SPP) within the Cloud-Fog-Edge continuum.

8.1 Approaches and Algorithms

The classification and analysis of research works on SPP formulations and resolution approaches reveal distinct trends in algorithmic preferences. As illustrated in Figure 4 of the accompanying PDF, population-based and greedy heuristics are the two most widely used approaches for SPP, followed by graph partitioning, reinforcement learning, and neural network methods.

- Heuristics (Greedy and Population-based): These approaches dominate the literature, which is unsurprising given the NP-hard nature of SPP. Heuristics are favored for their ability to compute suboptimal solutions rapidly, providing good quality results by iteratively improving candidate solutions. Their main advantage lies in their speed and relative simplicity of implementation. However, a significant limitation is their tendency to get stuck in local optima, meaning they do not guarantee a globally optimal solution. Furthermore, explicit approximation ratios—guarantees on how close the solution is to the optimum—are rarely provided, with only one identified approach [89] offering such a guarantee. This highlights a gap in the theoretical understanding and performance bounds of many heuristic solutions.

- Graph-based Approaches: Graph representations are highly effective for modeling network connections and inter-service communications in SPP. This explains the frequent use of graph-based algorithms, particularly those based on community detection, to determine optimized placements. These approaches are often unsupervised, relying on data analysis to discover patterns, create clusters, and identify communities. They are well-suited for problems where network topology and communication patterns are critical factors.

- Machine Learning (ML) Approaches: ML-based approaches, especially reinforcement learning and deep learning, are increasingly recognized for their effectiveness in proactive and dynamic placement strategies. Their ability to learn from historical data allows them to adapt to dynamic environments and optimize resource utilization, leading to more efficient and adaptable placement decisions. When combined with heuristic algorithms, ML solutions can yield quick and accurate results. The emergence of deep and neural networks in SPP is relatively recent, with the first reference to deep learning for SPP dating back to 2019 [9]. As more data becomes available for training and validation, the adoption of these sophisticated models is expected to grow. However, ML-based approaches come with their own set of challenges: they often require

significant computing power for training and inference, which can be a constraint in resource-limited fog environments. Their effectiveness is also highly dependent on the availability and quality of training data; insufficient or biased data can lead to suboptimal or unfair placement decisions. Therefore, balancing the benefits of adaptability and optimization with resource constraints and data quality is essential for successful ML implementation in SPP. A recent survey [81] specifically explores AI in this domain, offering more specific insights.

- Exact Approaches: While exact approaches are the only family guaranteed to provide optimal solutions, their high computational cost and long processing times make them impractical for large-scale SPP instances. The vast search space, even with techniques like Branch and Bound, limits their applicability to very small problem sizes, often guided by the linearity of the objective function and constraints. This explains their limited presence in the literature compared to approximate methods.

8.2 Datasets

A significant challenge in SPP research, particularly concerning microservices, is the scarcity of publicly available open-source projects and datasets that capture microservices patterns, connections, dependencies, and placement details. This lack of standardized datasets often forces researchers to compare microservices placement and chaining results with those of monolithic service functions, which is not entirely relevant. Microservices architectures are inherently more granular and distributed, with distinct communication patterns and dependencies, meaning placement strategies effective for monolithic services (treated as single, cohesive units) do not adequately address the complexities of microservices environments. This also hinders the full exploitation of parallelism and mutualization characteristics offered by microservices.

- Limited Public Datasets: Rahman et al. [88] made a notable contribution by proposing a small dataset of 20 microservices graphs to address this gap. Additionally, Sami et al. [97] utilized real-life datasets extracted from the Google Cluster Trace 2011-2 dataset and NASA Server Logs, providing more realistic scenarios for evaluating placement strategies.

- Synthetic Data Generation: To overcome the scarcity of real-world datasets, many researchers [26, 71, 95] resort to generating random graphs for both network topology and service/microservice dependency graphs. Commonly used generators include Barabasi-Albert networks [10] and growing random networks [50]. While useful for theoretical exploration, these synthetic datasets may not fully capture the complexities and nuances of real-world deployments.

- Mobility Trajectories: To address the dynamic nature of nodes and resources, particularly in mobile edge environments, some researchers incorporate mobility trajectories. For example, the San Francisco Taxi dataset [85] has been used in studies by Ray et al. [90] and Tang et al. [106] to simulate dynamic user movement and its impact on SPP.

The need for more comprehensive, dynamic, and publicly accessible datasets that accurately reflect the Cloud-Fog-Edge continuum remains a critical open challenge for advancing SPP research.

8.3 Prioritized Criteria

Based on the extensive literature review, the top three criteria most frequently optimized by researchers for the SPP are latency, cost, and Quality of Service (QoS). Table 6 in the accompanying PDF provides a detailed list of articles for each criterion. The prioritization of these criteria is partially related to their practical importance in real applications, but also significantly influenced by the ease with which they can be measured, modeled, and optimized.

- Latency: This is the most frequently optimized placement criterion. Its prominence can be attributed to several factors: it directly impacts user experience, especially for real-time applications [5, 43, 96, 110, 114]. Latency is influenced by measurable factors such as distance between user and service, available bandwidth, propagation delay, and processing power. These factors are relatively straightforward to quantify, predict, and optimize, often independently of other criteria. The additive nature of latencies across a path simplifies its calculation and optimization compared to more complex, intertwined parameters.

- Cost: Cost minimization is a primary objective for users and service providers in SPP. The aim is to find solutions that reduce service delivery expenses and minimize overall financial and operational costs [18, 20, 25, 30, 38, 49, 56, 64, 74, 83, 84, 87, 101, 106, 107, 111, 119, 120, 122]. Generally, cost is associated with limits on execution nodes, which are relatively easy to incorporate into SPP models.

- Quality of Service (QoS): User satisfaction is a significant concern for researchers, leading to the frequent inclusion of QoS as an optimization criterion [14, 20, 23, 32, 34, 37, 38, 43, 62, 64, 77, 80, 87, 92, 95, 97, 100, 109, 116]. QoS can encompass various metrics such as availability, reliability, throughput, and response time. It is a critical concern for applications requiring high availability and reliability.

- Energy/CO2 Gas Emissions: Despite growing environmental concerns, energy consumption and CO2 emissions are less frequently optimized compared to latency, cost, and QoS [2, 18, 20, 43, 54, 56, 74, 82, 91, 93, 99, 105, 106, 109, 117]. The existing works that do consider energy typically focus only on hardware consumption (CPU, GPU, RAM) and electrical power of inter/intra-execution nodes, often neglecting the energy

generated by communication links to avoid introducing excessive parameters. Energy optimization in SPP is often modeled as a complex combination of constraints, making it harder to understand and justify. Furthermore, minimizing energy consumption can sometimes conflict with other objectives, such as latency (e.g., selecting a node further away to save energy might increase latency), necessitating a careful trade-off. There is a clear need for more holistic models that account for all energy-related factors across the continuum.

- Resources: Resource utilization is also a key criterion, ensuring efficient use of CPU, memory, storage, and bandwidth [21, 26, 34, 38, 39, 53, 62, 71, 89, 91, 117, 122].

8.4 Evaluation Tools

To evaluate their proposed methodologies, researchers in SPP utilize a diverse set of simulation frameworks, experimental platforms, and analytical tools.

- Simulation Frameworks:

  - YAFS (Yet Another Fog Simulator): This Python-based tool is specifically designed for analyzing Fog Computing architectures, including resource placement, deployment costs, and network design, making it highly suitable for IoT environments.

  - CloudSim: A Java-based simulation framework widely used for modeling and evaluating cloud infrastructures. It supports the simulation of data centers, Virtual Machine (VM) management, resource allocation, and energy consumption analysis.

  - iFogSim: An extension of CloudSim, iFogSim is designed to assess resource management strategies and application architectures in Fog Computing environments, providing specialized support for Fog scenarios.

- Experimental Platforms/Testbeds: Some studies use small-scale physical testbeds (e.g., Raspberry Pi devices [89]) to validate their approaches in more realistic settings, albeit with limited scalability.

- Analytical Tools and Solvers: General-purpose programming languages like Java, C++, and Matlab are frequently used for implementing algorithms and conducting simulations. These are often combined with commercial or open-source optimization solvers such as IBM CPLEX or Gurobi for solving complex mathematical programming models.

- Custom-built Simulators: In some cases, researchers develop custom-built simulators (e.g., DRACeo [109], custom Java EE 8 Platform simulations [93]) to precisely model their specific problem settings and evaluate their algorithms.

The choice of evaluation tool often depends on the complexity of the proposed model, the scale of the desired simulation, and the specific metrics being optimized. While simulations offer flexibility and control, real-world deployments and large-scale testbeds are crucial for validating the practical applicability and scalability of proposed solutions.

Open Challenges and Future Directions

Based on the comprehensive analysis of the existing literature on service placement in the Cloud-Fog-Edge continuum, several significant open challenges and promising future research directions emerge. Addressing these areas will be crucial for unlocking the full potential of distributed computing architectures and for developing truly intelligent, adaptive, and sustainable service placement solutions.

9.1 Application Architecture: Evolving Microservices and Beyond

Microservices-based applications are increasingly favored for IoT deployments due to their inherent modularity, loose coupling, and reusability across various applications. This architectural style enables greater agility, scalability, and resilience. However, they introduce complexities related to data consistency, inter-service communication overhead, and privacy concerns in a highly distributed environment.

- Context-Aware Placement: Future placement strategies should consider the specific operational contexts of applications and the characteristics of data (e.g., sensitivity, volume, velocity) before deploying microservices. This involves understanding not just resource requirements but also data flow patterns, security policies, and compliance regulations.

- Enhanced Security and Privacy: With microservices distributed across a vast continuum, ensuring end-to-end security and data privacy becomes paramount. Research is needed on developing placement algorithms that incorporate security and privacy as first-class optimization objectives, potentially leveraging homomorphic encryption, federated learning, or secure multi-party computation techniques to protect sensitive data and computations.

- Standardized Protocols and Interoperability: While microservices communicate via APIs, the lack of standardized protocols for service discovery, orchestration, and inter-service communication across heterogeneous Cloud-Fog-Edge environments can hinder interoperability. Future advancements may focus on developing unified frameworks and protocols that simplify deployment and management across diverse platforms.

- Serverless Functions and Function-as-a-Service (FaaS): The rise of serverless computing, where developers write and deploy individual functions without managing underlying infrastructure, presents new opportunities and challenges for service placement. Optimizing the placement of these ephemeral, event-driven functions across the continuum requires novel

approaches that consider cold start latencies, resource pooling, and dynamic scaling in highly bursty workloads.

9.2 Collection and Sharing of Comprehensive Datasets

As highlighted in Section 8.2, a significant impediment to progress in SPP research is the notable lack of publicly available, open-source projects and datasets that accurately represent real-world microservices patterns, connections, dependencies, and dynamic placement scenarios.

- Bridging the Gap between Synthetic and Real-world Data: While random graphs [10, 50] and mobility datasets [85] are useful for theoretical exploration, they often fail to capture the intricate complexities and nuances of real-world deployments. This limits the ability to rigorously compare and validate different SPP methods and tools.
- Need for Large-Scale, Dynamic Datasets: The research community urgently needs to address this issue by releasing comprehensive datasets on large-scale microservices applications. These datasets should include:
  - The number and types of services.
  - Detailed inter-service dependencies and communication patterns (e.g., call graphs, data volumes).
  - Resource requirements and utilization profiles for individual microservices.
  - Network topology and characteristics of iCFE nodes (CPU, memory, storage, bandwidth, latency).
  - Sufficiently long runtime traces to replay real-life scenarios, including fluctuating workloads, node failures, and user mobility.
  - Metrics related to energy consumption and carbon footprint, if possible.
- Collaborative Data Collection Initiatives: Fostering collaborative initiatives among industry and academia to collect and anonymize real-world operational data could significantly accelerate research and enable more realistic evaluations of SPP solutions.

9.3 Dynamic Placement Approaches: Beyond Reactive Strategies

Most existing literature proposals often focus on static or reactive placement, which are neither very realistic nor efficient in the highly dynamic Cloud-Fog-Edge continuum. To accommodate object mobility, fluctuating workloads, and rapid system changes, proactive and adaptive placement strategies are essential.

- Predictive Models for Proactive Placement: Future research should focus on developing sophisticated predictive models (e.g., using advanced machine learning, time series analysis) that can forecast resource demands, network conditions, user mobility patterns, and potential node failures. These predictions can then inform proactive placement and migration decisions, preventing performance degradation before it occurs [90, 111].
- Online Learning and Adaptive Control: Developing Reinforcement Learning (RL) agents that can learn and adapt their placement policies in real-time, based on continuous feedback from the environment, is a promising direction. These agents need to handle non-stationary environments and explore optimal strategies without explicit programming.
- Self-Organizing and Autonomous Systems: The ultimate goal is to move towards fully self-organizing and autonomous SPP systems that can dynamically adjust service placements without human intervention, reacting intelligently to unforeseen events and optimizing for multiple objectives simultaneously. This requires robust control loops, intelligent agents, and distributed decision-making mechanisms.
- Mobility-Aware Service Provisioning: With increasing user and device mobility, service placement must explicitly account for movement patterns. This involves strategies for efficient service migration, state transfer, and context awareness to ensure seamless service continuity and minimal disruption as users move between edge nodes [59, 77, 112].

9.4 Microservices Task Scheduling and Offloading: An Integrated View

Efficient task scheduling in microservice architectures is challenging due to variable traffic and dynamic resource availability. Poor scheduling can lead to increased energy consumption, SLA breaches, and reduced user satisfaction. Task offloading, the transfer of computational tasks to remote resources, complements service placement by optimizing processing efficiency and reducing latency for local devices [3].

- Integrated Network Management Pipelines: Currently, many studies tend to focus on isolated aspects of network management (e.g., only placement, only scheduling, or only offloading). A more holistic and integrated view is necessary. Future research should develop comprehensive frameworks that seamlessly integrate service placement, task scheduling, and computation offloading decisions. This involves designing complete pipelines that consider the interdependencies between these processes to ensure cohesive and efficient system operation.
- Joint Optimization of Placement, Scheduling, and Offloading: The goal should be to develop algorithms that jointly optimize these three aspects, considering their combined impact on system performance, energy consumption, and user experience. This would likely involve complex multi-objective optimization problems solved using advanced meta-heuristics or AI techniques.
- Resource Contention and Load Balancing: Effective

scheduling and offloading are critical for managing resource contention in heterogeneous environments. Algorithms need to dynamically balance workloads across the continuum, preventing hotspots and ensuring fair resource allocation among competing services and tasks.

9.5 Better Consideration of the Energy Criterion: Towards Green Computing

Despite the growing awareness of environmental impact, research on energy consumption and resource management in SPP remains somewhat limited, often focusing only on hardware energy. The massive increase in energy and CO2 emissions from digital infrastructure necessitates a more comprehensive approach to green computing in SPP.

- Holistic Energy Modeling: The majority of existing research articles consider only the energy consumption of hardware components (CPU, GPU, RAM). However, distributed services significantly increase the energy consumption of communication links. Future models must account for the energy cost of data transfer across the network, including wired and wireless communication.

- Expanded Placement Criteria for Sustainability: Beyond basic energy consumption, SPP criteria should explicitly include:

  - Gray Energy and Primary Energy Emissions: Measured in kWh, accounting for the energy embedded in the manufacturing and disposal of devices.

  - Relative Carbon Footprint Emissions: Considering the carbon intensity of different energy sources used by computing nodes (e.g., renewable vs. fossil fuels).

  - Life Cycle Analysis (LCA): A comprehensive assessment of the environmental impacts of all terminal devices throughout their entire life cycle (manufacturing, use, and end-of-life).

  - Utilization of Renewable Energy Sources: Placement decisions could prioritize nodes powered by renewable energy sources like wind turbines or solar panels.

- Interdisciplinary Collaboration: Incorporating such a wide array of criteria makes the SPP significantly more complex. This necessitates interdisciplinary collaboration between computer scientists, operational researchers, environmental scientists, and economists to develop robust models that accurately capture the interplay of these factors. The scientific community needs to work on establishing standardized models and metrics for evaluating the environmental footprint of service deployments.

- Trade-offs with Performance Metrics: Minimizing energy consumption can sometimes conflict with other critical objectives like QoS or latency (e.g., placing a service further away to leverage a greener data center might increase latency). Future research must explore sophisticated multi-objective optimization techniques that effectively manage these complex trade-offs, finding solutions that are both environmentally sustainable and performant.

CONCLUSION

In this comprehensive survey, we have presented a detailed classification and analysis of the algorithmic solutions employed to address the Service Placement Problem (SPP) within the integrated Cloud-Fog-Edge (iCFE) computing environments. Our study has focused on the algorithmic approaches, considering various influencing parameters such as the infrastructure environment, the type of application components (e.g., monolithic, inter-dependent, microservices), and the chosen placement mode (e.g., centralized, decentralized, static, dynamic, reactive, proactive). We have thoroughly examined key optimization objectives, including latency, quality of service, cost, energy efficiency, and resource utilization, recognizing that the quest for optimal placement in such complex, multi-criteria scenarios constitutes a challenging combinatorial problem.

The survey categorized existing solutions into four main families: exact optimization-based approaches, graph-based approaches, heuristic and meta-heuristic solutions, and machine learning-driven algorithms. For each category, we have provided an in-depth review of specific algorithms, their working principles, and their application to SPP, supported by extensive citations to the relevant literature.

Our analysis and discussion have highlighted several critical aspects of the current state-of-the-art. We observed the prevalence of heuristic and meta-heuristic methods due to the NP-hard nature of the problem, alongside the growing adoption of graph-based and machine learning techniques for their ability to handle dynamic environments and complex dependencies. A significant limitation identified is the scarcity of comprehensive, real-world datasets for microservices, which hinders robust comparative evaluations. Furthermore, while latency, cost, and QoS are frequently prioritized, a more holistic consideration of energy consumption and environmental impact remains an underexplored area.

Building upon these insights, we have proposed several crucial avenues for future research. These include developing more sophisticated context-aware and dynamic placement strategies for evolving application architectures (especially serverless and advanced microservices), fostering collaborative initiatives for collecting and sharing large-scale, dynamic datasets, and creating integrated network management pipelines that jointly optimize placement, scheduling, and offloading. Most importantly, there is a pressing need for a more comprehensive and holistic approach to green computing in SPP, incorporating detailed energy modeling and a

broader range of environmental criteria throughout the entire life cycle of digital infrastructure. By addressing these challenges, the research community can pave the way for truly ubiquitous, responsive, and environmentally sustainable computing in the Cloud-Fog-Edge continuum.

REFERENCES

Bernardetta Addis, Giuliana Carello, and Meihui Gao. 2020. On a virtual network functions placement and routing problem: Some properties and a comparison of two formulations. Networks 75, 2 (2020), 158–182.

Ehsan Ahvar, Shohreh Ahvar, Zoltán Ádám Mann, Noel Crespi, Roch Glitho, and Joaquin Garcia-Alfaro. 2021. DECA: A dynamic energy cost and carbon emission-efficient application placement method for edge clouds. IEEE Access 9 (2021), 70192–70213.

Mohammad Yahya Akhlaqi and Zurina Binti Mohd Hanapi. 2023. Task offloading paradigm in mobile edge computing-current issues, adopted approaches, and future directions. J. Netw. Comput. Appl. 212 (2023), 103568.

Mahmoud A. M. Albreem, Ayman A. El-Saleh, Muzamir Isa, Wael Salah, M. Jusoh, M. M. Azizan, and A. Ali. 2017. Green internet of things (IoT): An overview. In Proceedings of the IEEE 4th International Conference on Smart Instrumentation, Measurement and Application (ICSIMA'17). IEEE, Putrajaya, Malaysia, 1–6. https://doi.org/10.1109/ICSIMA. 2017.8312021

Yasser Aldwyan and Richard O. Sinnott. 2019. Latency-aware failover strategies for containerized web applications in distributed clouds. Future Gen. Comput. Syst. 101 (2019), 1081–1095.

Mohammad Reza Alizadeh, Vahid Khajehvand, Amir Masoud Rahmani, and Ebrahim Akbari. 2020. Task scheduling approaches in fog computing: A systematic review. Int. J. Commun. Syst. 33, 16 (2020), e4583.

Gabriel Araújo, Vandirleya Barbosa, Luiz Nelson Lima, Arthur Sabino, Carlos Brito, Iure Fé, Paulo Rego, Eunmi Choi, Dugki Min, Tuan Anh Nguyen et al. 2024. Energy consumption in microservices architectures: A systematic literature review. IEEE Access 12 (2024), 186710–186729.

Onur Ascigil, Truong Khoa Phan, Argyrios G Tasiopoulos, Vasilis Sourlas, Ioannis Psaras, and George Pavlou. 2017. On uncoordinated service placement in edge-clouds. In Proceedings of the IEEE International Conference on Cloud Computing Technology and Science (CloudCom'17). IEEE, Hong Kong, Hong Kong, 41–48.

Yixin Bao, Yanghua Peng, and Chuan Wu. 2019. Deep learning-based job placement in distributed machine learning clusters. In Proceedings of the Conference on Computer Communications (INFOCOM'19). IEEE, Paris, France, 505–513.

Albert-László Barabási and Réka Albert. 1999. Emergence of scaling in random networks. Science 286, 5439 (1999), 509–512.

Kay Bierzynski, Antonio Escobar, and Matthias Eberl. 2017. Cloud, fog and edge: Cooperation for the future?. In Proceedings of the 2nd International Conference on Fog and Mobile Edge Computing (FMEC'17). IEEE, Valencia, Spain, 62–67.

Vincent D. Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. 2008. Fast unfolding of communities in large networks. J. Stat. Mech.: Theory Exper. 2008, 10 (2008), P10008.

Christian Blum and Andrea Roli. 2001. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. ACM Comput. Surv. 35 (01 2001), 268–308. https://doi.org/10.1145/937503.937505

Antonio Brogi and Stefano Forti. 2017. QoS-aware deployment of IoT applications through the Fog. IEEE Internet Things J. 4 (2017), 1185–1192.