

Review

Optimizing Kubernetes with Multi-Objective Scheduling Algorithms: A 5G Perspective

Mazen Farid ^{1,2,*}, Heng Siong Lim ^{3,*}, Chin Poo Lee ⁴, Charilaos C. Zarakovitis ⁵ and Su Fong Chien ⁵

¹ Faculty of Information Science and Technology (FIST), Multimedia University, Melaka 75450, Malaysia

² Centre for Intelligent Cloud Computing, COE for Advanced Cloud, Multimedia University, Melaka 75450, Malaysia

³ Faculty of Engineering and Technology, Multimedia University, Melaka 75450, Malaysia

⁴ School of Computer Science, University of Nottingham Ningbo China, 199 Taikang East Road, Yinzhou District, Ningbo 315100, China; chin-poo.lee@nottingham.edu.cn

⁵ ICT Department, axon logic IKE, 14122 Athens, Greece; c.zarakovitis@axonlogic.gr (C.C.Z.); sufong.chien@axonlogic.gr (S.F.C.)

* Correspondence: ramadhan.mazen@mmu.edu.my (M.F.); hslim@mmu.edu.my (H.S.L.)

Abstract

This review provides an in-depth examination of multi-objective scheduling algorithms within 5G networks, with a particular focus on Kubernetes-based container orchestration. As 5G systems evolve, efficient resource allocation and the optimization of Quality-of-Service (QoS) metrics, including response time, energy efficiency, scalability, and resource utilization, have become increasingly critical. Given the scheduler's central role in orchestrating containerized workloads, this study analyzes diverse scheduling strategies designed to address these competing objectives. A novel taxonomy is introduced to categorize existing approaches, offering a structured view of deterministic, heuristic, and learning-based methods. Furthermore, the review identifies key research challenges, highlights open issues, such as QoS-aware orchestration and resilience in distributed environments, and outlines prospective directions to advance multi-objective scheduling in Kubernetes for next-generation networks. By synthesizing current knowledge and mapping research gaps, this work aims to provide both a foundation for newcomers and a practical reference for advancing scholarly and industrial efforts in the field.

Keywords: 5G network; multi-objective scheduling algorithms; Kubernetes; QoS



Academic Editor: Paolo Bellavista

Received: 11 July 2025

Revised: 8 September 2025

Accepted: 11 September 2025

Published: 15 September 2025

Citation: Farid, M.; Lim, H.S.; Lee, C.P.; Zarakovitis, C.C.; Chien, S.F. Optimizing Kubernetes with Multi-Objective Scheduling Algorithms: A 5G Perspective. *Computers* **2025**, *14*, 390. <https://doi.org/10.3390/computers14090390>

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Cloud service providers (CSPs) employ various networking devices to deliver services to their clientele. One technology instrumental in enhancing service efficiency is Network Function Virtualization (NFV), which involves the virtualization of hardware equipment to streamline service provision. Within the NFV framework, Virtual Network Functions (VNFs) represent the virtual counterparts of traditional network functions, running atop commercial-off-the-shelf (COTS) servers in the form of virtual machines. The adoption of NFV yields several advantages, including cost reduction associated with hardware and maintenance, simplified network updates, space savings for new device installations, and decreased power consumption. In parallel, software-defined networking (SDN) emerges as a key solution, efficiently managing VNFs centrally through the widely used OpenFlow Application Programming Interface (API). This approach effectively decouples the data plane from the control plane, contributing to enhanced network robustness in terms of

security, power consumption, and performance optimization. The subsequent section delves into the application of NFV in the context of 5G.

In [1], NFV stands as a foundational element in the architecture of 5G networks, playing a pivotal role in the virtualization of hardware devices within the network. Beyond hardware virtualization, NFV equips cloud-based systems with an agile network architecture, fostering the development of flexible and programmable networks tailored to meet the specific requirements of applications, clients, or operators. This strategic integration of NFV within the 5G ecosystem marks a significant leap toward achieving a more adaptive and responsive networking infrastructure [2].

Kubernetes is a leading open-source platform that automates the deployment, scaling, and management of containerized applications. Operating on a declarative model, it allows users to define the desired state of their applications while the system ensures its maintenance, simplifying operations and enhancing consistency. It further integrates robust monitoring and self-healing mechanisms that automatically detect and recover from failures, ensuring resilience and high availability. With its comprehensive toolset, Kubernetes has become a powerful and flexible solution for managing containerized applications, particularly in large-scale production environments.

In [3], Kubernetes finds the optimal utility in the realm of microservice-based web applications, where the modularity of each application component is encapsulated within its own container. This microservice architecture aligns seamlessly with Kubernetes, capitalizing on its capabilities to orchestrate and manage diverse containerized components cohesively. In essence, Kubernetes emerges as a cornerstone in modern application development, providing a versatile and efficient platform for the orchestration and maintenance of containerized applications in dynamic production environments.

As evidenced by the wide range of applications and issues linked with Kubernetes, it is critical to investigate proposed algorithms in the related domain in order to find state-of-the-art and future research directions. Numerous studies have focused on the creation of new Kubernetes algorithms. The primary goal of this survey is to provide a complete assessment of the current state of the art in the topic of Kubernetes-scheduling algorithms in 5G. We hope to give a critical evaluation of the strengths and limitations of existing methodologies by evaluating the existing literature and identifying the major theories, methods, and findings from past studies. We also intend to identify gaps in the existing literature and unanswered topics, as well as make recommendations for future study directions.

Our overall objectives are to advance the subject and offer a helpful tool for practitioners and scholars that deal with Kubernetes-scheduling methods. While several surveys exist on container orchestration, including those by [4–7], these works primarily provide broad overviews of orchestration frameworks and container technologies without a focused analysis of scheduling in 5G contexts. Other surveys, such as those in [8–10], target clouds, fog, or big-data environments, while Wang et al. [11] emphasize scheduling in data center networks. More recently, [12,13] have offered valuable and structured surveys on Kubernetes scheduling. However, these contributions mainly classify scheduling strategies at a general level and do not extensively examine their implications for multi-objective optimization in 5G. Building on these foundations, our survey distinguishes itself by providing a state-of-the-art review of Kubernetes scheduling, specifically for 5G network functions and applications. We introduce a taxonomy that links scheduling methods with key 5G performance metrics, analyze diverse approaches across cloud-native network functions (CNFs), and highlight research gaps that must be addressed to advance multi-objective scheduling in the emerging 5G system. Container technologies provide notable efficiency

gains over VMs by reducing resource consumption, improving deployment flexibility, and lowering operational costs [4–7].

This survey thus serves as a valuable reference for researchers aiming to address the limitations and challenges of Kubernetes scheduling in 5G contexts. The main contributions of this paper are as follows:

1. **Comprehensive Literature Review:** A thorough examination of existing work on Kubernetes scheduling, with a focus on 5G use cases and networking. The review is organized into two sub-categories: generic scheduling and multi-objective optimization-based scheduling;
2. **Critical Analysis:** A meticulous evaluation of the strengths and weaknesses of current Kubernetes schedulers for 5G, offering insights into their effectiveness and limitations;
3. **Gap Identification and Open Questions:** Identification of key research gaps and open questions in the literature, highlighting areas that require further investigation and innovation;
4. **Multi-objective Scheduling Approaches:** A focused discussion on bi-objective, tri-objective, and multi-objective scheduling strategies identified in the existing research.

Table 1 presents a comparative overview of related work on container-based scheduling algorithms.

Table 1. Related work on container-based scheduling algorithms.

Ref.	Survey
A Survey on Common Problems and Solution Approaches [12]	The article reviews advances in Kubernetes scheduling, explaining that the default scheduler handles general tasks well but struggles with modern demands, like machine learning, deep learning, and edge computing. It classifies research by objectives, workload types, and environments and highlights optimization strategies.
A Survey on Scheduling Techniques in Computing and Network Convergence [14] (2024)	The survey comprehensively reviews literature on scheduling across various scenarios, including computing and network convergence from heterogeneous resources, multiple-objective optimization, and diverse task environments. It explores potential explanations and implications arising from these studies and identifies important challenges for future research.
Container Scheduling Techniques: A Survey and Assessment [15] (2022)	The survey categorizes scheduling techniques into four groups based on the optimization algorithm used: mathematical modeling, heuristics, meta-heuristics, and machine learning. For each category, it analyses the strengths, weaknesses, and challenges of the techniques, focusing on their performance metrics. The paper concludes by outlining promising avenues for future research aimed at maximizing the benefits of container technology.
Deep Reinforcement Learning for Resource Management on Network Slicing: A Survey [16] (2022)	This survey identifies key stages in network-slicing resource management and examines how reinforcement-learning (RL) and deep reinforcement-learning (DRL) algorithms can independently handle each stage. It evaluates these approaches based on their optimization goals, state and action spaces, algorithmic choices, deep neural network structures, exploration–exploitation strategies, and specific use cases (vertical applications). Additionally, the survey outlines future research paths concerning RL/DRL-based resource management for network slicing.
Kubernetes Scheduling: Taxonomy, Ongoing Issues, and Challenges [13] (2022)	This survey specifically focuses on the scheduler, a critical orchestrator responsible for allocating physical resources to containers. Scheduling strategies are tailored around various quality-of-service (QoS) parameters. The paper seeks to provide comprehensive insights into Kubernetes scheduling, identify key gaps, and offer guidance for future research directions. It conducts an empirical study on Kubernetes-scheduling techniques and introduces a novel taxonomy for categorizing them. The survey also addresses challenges, outlines future directions, and explores research opportunities in this field.
VNF and CNF Placement in 5G: Recent Advances and Future Trends [17] (2022)	This study categorizes optimization techniques for VNF placement, organizing papers by metrics, methods, algorithms, and deployment contexts. Virtualization spans beyond VMs and VNFs to encompass microservices, containers, and cloud-native systems. The article’s second part focuses on CNF placement in edge/fog computing, addressing challenges like traffic congestion, resource use, energy efficiency, and performance degradation.
A Survey on the Placement of Virtual Resources and Virtual Network Functions [18] (2019)	This survey extensively investigates existing VNF placement strategies and algorithms, categorizing them into network function chain placement, VNF-forwarding graphs, and VNF replications. It covers generic VM-based NFV frameworks and specific VNF placement strategies. The survey reviews relevant protocols, heuristics, algorithms, and architectures with the aim of proposing efficient strategies for network slicing in future work, ensuring satisfaction for end-users and verticals while respecting various constraints.
This work	This paper offers a comprehensive review of Kubernetes advancements specific to orchestrating 5G network functions and use cases. It focuses on key metrics studied in this domain and explores various methods applicable to Kubernetes scheduling in 5G scenarios. The taxonomy categorizes different aspects of Kubernetes resource scheduling, extending beyond algorithms and methods.

1.1. Motivation

The challenge highlighted above serves as the driving force behind this review, which delves into multi-objective scheduling algorithms within the 5G framework utilizing Kubernetes. These strategies are categorized based on their objectives and characteristics.

1.2. Search Strategy

The creation of search phrases encompassed the following steps [19]: (1) selecting keywords based on the research questions, (2) identifying synonyms and variations in spelling to expand the keyword range, (3) assessing the identified keywords in the pertinent literature, (4) combining synonyms and alternative spellings using 'Boolean OR', and (5) connecting the main terms and conditions with 'Boolean AND'. Following these steps, we derived comprehensive terms and conditions associated with QoS customization. The resulting search string utilized in digital library engines is as follows: “(“Kubernetes” OR “K8s” OR “containers”) AND (“Scheduling” OR “placement” OR “network slicing”) AND (“5G” OR “Fifth Generation”) AND (“multi objectives” OR “multi-objectives”)).

1.3. Eligibility Criteria

Publications considered included peer-reviewed journal articles, conference papers, and surveys published between 2005 and 2024. Studies were included if they explicitly addressed Kubernetes scheduling, multi-objective optimization, or resource management in 5G contexts. Works were excluded if they lacked sufficient methodological detail, fell outside the scope of Kubernetes or 5G, or provided only high-level overviews without technical relevance.

1.4. Data Extraction

From 128 collected papers, 60 were selected after applying the criteria. For each, information was extracted regarding algorithm type (e.g., heuristic, metaheuristic, and ML based), optimization goals (latency, energy, QoS, and scalability), validation approach (simulation, testbed, and real deployment), and key strengths/limitations. Findings were then synthesized and presented in structured tables or comparative analysis.

1.5. Sources of Data

Research papers addressing scheduling strategies incorporating multi-objective algorithms in 5G with Kubernetes techniques were rigorously retrieved from electronic databases, which included resources such as IEEE Xplore, ResearchGate, Google Scholar, the Wiley Library, and more. The subsequent online electronic databases were utilized for the purpose of this research:

- (1) IEEE Xplore (<www.ieeexplore.ieee.org>);
- (2) Science Direct (<www.scidirect.com>);
- (3) Springer (<www.springer.com>);
- (4) ResearchGate (<www.researchgate.net>);
- (5) Google Scholar (<scholar.google.com>);
- (6) Scopus (<www.scopus.com>);
- (7) Taylor & Francis (<taylorandfrancis.com>);
- (8) The Wiley Library (<www.onlinelibrary.wiley.com>).

2. Background

2.1. Microservices

Microservices, a paradigm within software architecture, entails the decomposition of an application into small, autonomous modules known as microservices, as depicted in

Figure 1. Each microservice assumes responsibility for a specific functionality of the application, fostering modularity and encapsulation. Communication between these microservices is typically facilitated through a message bus, enabling seamless interaction while maintaining autonomy. This architectural approach yields various advantages, including the automation of deployment, scaling, and management processes.

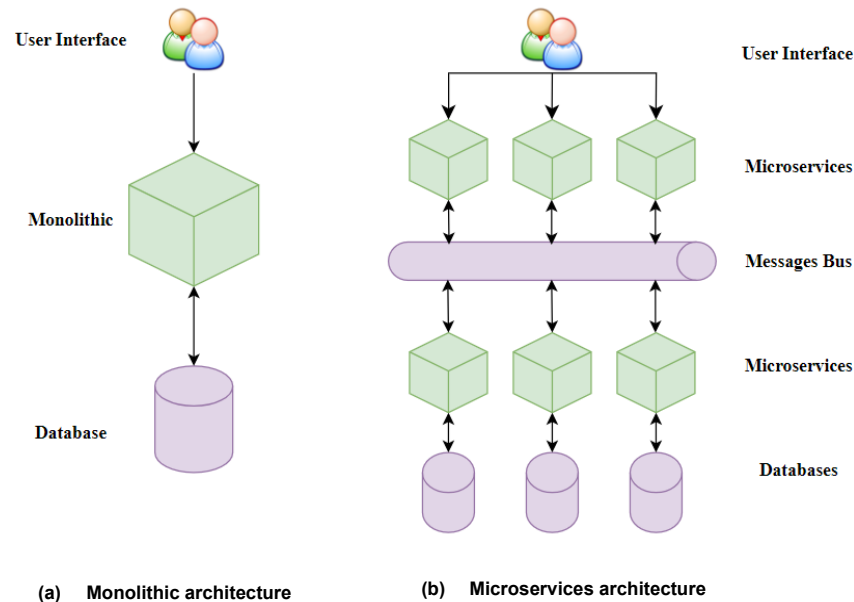


Figure 1. Comparison between different application architectures.

The intrinsic independence of each microservice proves to be instrumental in facilitating agility and flexibility in software development. As each microservice operates autonomously and can be managed and updated independently, implementing changes becomes more straightforward without causing disruptions to the entire system. Furthermore, microservices are not confined to a single programming language or server; they can be developed in diverse languages and executed on different servers, providing developers with unparalleled flexibility in the development process.

Kubernetes emerges as a key orchestration tool in the realm of microservices, seamlessly adapting to fluctuating demand intensities [3]. For instance, in the contexts of real-time analytics, machine learning, and deep learning applications, Kubernetes can dynamically scale the number of pods based on the current demand. During periods of low visitor traffic, the application can be efficiently scaled down to a few pods, conserving resources and reducing costs. Conversely, if the application experiences a surge in popularity with a substantial influx of visitors, Kubernetes can rapidly scale up the deployment, ensuring the application can robustly handle varying levels of demand. This dynamic scalability, coupled with the versatility of microservices, exemplifies the power of contemporary software architectures in meeting the challenges of a dynamic and ever-evolving digital landscape.

2.2. Containers

The advent of containerization has revolutionized the landscape of software deployment and resource utilization. Containers, known for their lightweight nature, present a significant advancement over traditional virtual machines, as depicted in Figure 2. Their lightweight design allows for swift creation and destruction, resulting in faster and more efficient utilization of computing resources. Unlike virtual machines, containers encapsulate applications and their dependencies without the need for a separate operating system instance, streamlining deployment processes and minimizing overhead.

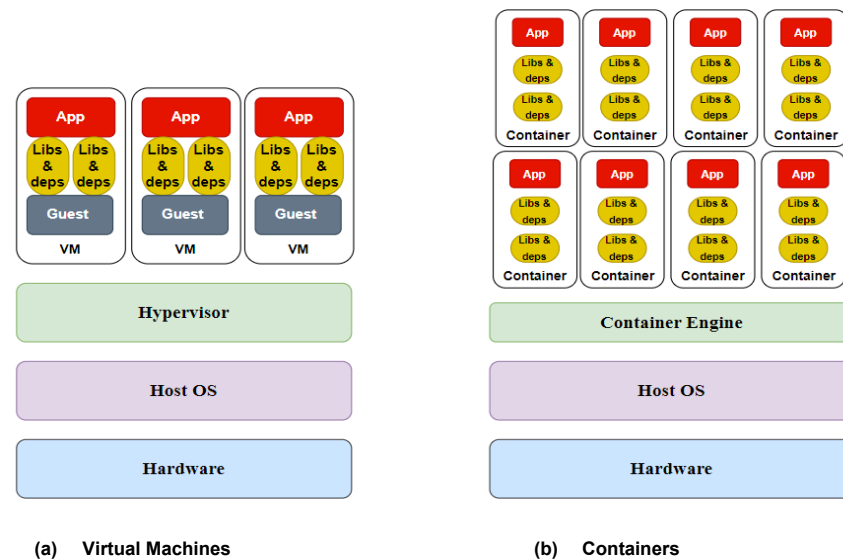


Figure 2. Comparison between different types of application development.

In this context, Kubernetes emerges as a powerful orchestration platform that automates the deployment, scaling, and management of containers across a cluster of machines. This orchestration capability addresses the complexities associated with managing large-scale containerized applications. By automating various aspects of container lifecycle management, Kubernetes enhances resource utilization efficiency and provides flexibility in handling diverse workloads. The automated orchestration offered by Kubernetes simplifies the development and maintenance of complex applications, allowing developers to focus on building robust and scalable solutions without the burden of intricate deployment logistics. In essence, the combination of lightweight containers and the automation prowess of Kubernetes represents a transformative approach to application deployment, fostering efficiency, flexibility, and ease of management in contemporary software development practices.

While containers revolutionize deployment efficiency, their lightweight and dynamic nature creates new challenges for scheduling in telecommunication workloads. In large-scale 5G deployments, where ultralow latency and high reliability are critical, container orchestration requires intelligent scheduling strategies that optimize not only resource use but also QoS parameters, such as availability, resilience, and response time [12,13,20]. This direct link between containerization benefits and scheduling complexity motivates the focus of this survey.

2.3. Kubernetes Architecture

Kubernetes [20] (K8s), an open-source system, streamlines the deployment, scaling, and management of containerized applications. It organizes containers into pods within a cluster, where a master node assigns container workloads from user pods to worker nodes. Containers in a pod share resources, like filesystems, kernel namespaces, and an IP address. Deployments can be paired with a service tier to facilitate horizontal scaling or ensure availability. While Docker Swarm [21] and Kubernetes are both options for container orchestration [22], Docker Swarm's ease of deployment is offset by its lack of scalability for enterprise solutions, making it less suitable for large-scale pod deployments. One limitation of Kubernetes is its inability to automatically configure multiple master nodes in the open-source version. K8s has a scalability ceiling of 5000 nodes per cluster and 150,000 pods [23].

Kubernetes architecture is not merely a framework for container orchestration but also the foundation for multi-objective scheduling in 5G. For example, the kube scheduler governs workload placement across nodes, directly shaping performance indicators, like latency, throughput, and fault tolerance [12,13,24]. Similarly, the controller manager ensures reliability and resilience through state consistency, while etcd underpins scalability by maintaining cluster-wide configurations. These architectural functions map directly onto scheduling research: Heuristic algorithms (e.g., DRAP [25]) emphasize cost and resilience; optimization models (e.g., ILP [25]) aim for precision but face scalability limits, and AI/ML-based schedulers [26,27] enable predictive scaling for improved efficiency. Thus, Kubernetes components form the operational backbone that scheduling algorithms exploit to achieve 5G objectives of ultra-reliability, low latency, and energy efficiency [28], [26].

Despite this constraint, Kubernetes remains the dominant choice in the market and is widely acknowledged as the standard [7]. Figure 3 illustrates the Kubernetes architecture:

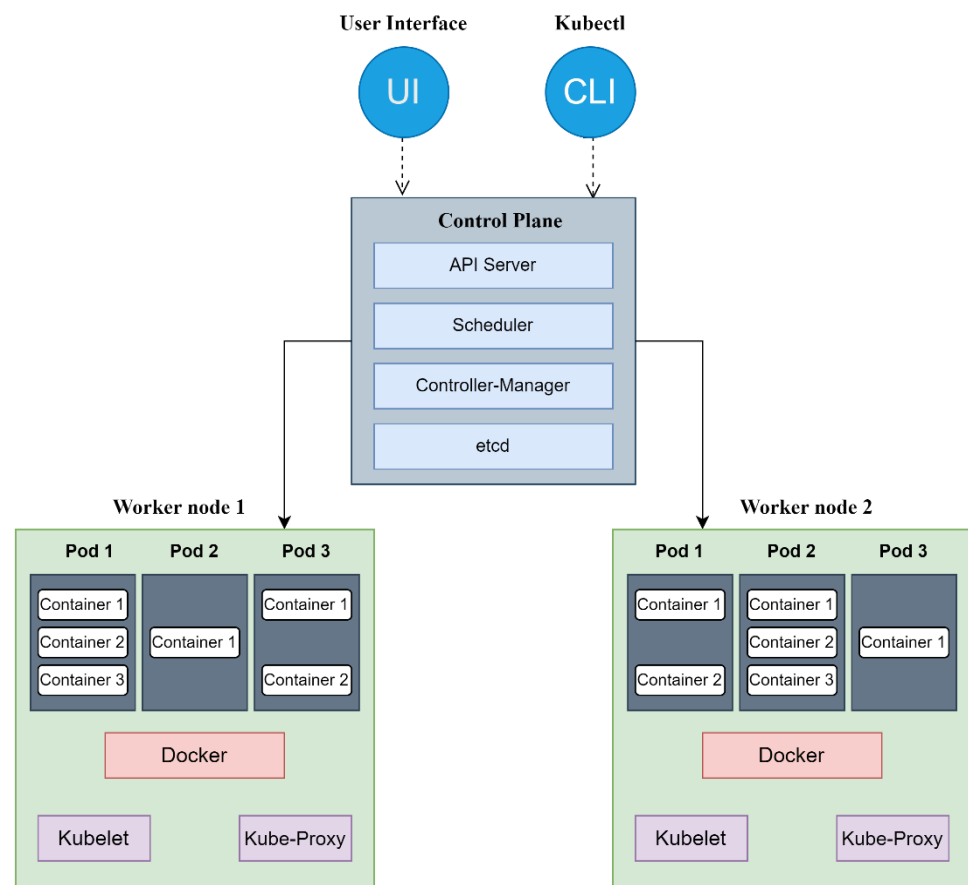


Figure 3. Kubernetes architecture.

User Interface: This refers to the Kubernetes Dashboard. The Kubernetes Dashboard is a web-based graphical user interface that allows users to manage and monitor their Kubernetes clusters. It provides an overview of the clusters' resource usage; allows users to view and manage deployments, pods, services, and other Kubernetes objects; and offers various monitoring and troubleshooting features.

Kubectl: This is a command-line interface (CLI) tool used for interacting with Kubernetes clusters. It allows users to perform various operations, such as deploying applications, managing clusters, inspecting resources, and executing commands, on Kubernetes clusters. With kubectl, users can control and manage their Kubernetes clusters efficiently through simple and intuitive commands.

Control Plane: This refers to the master node responsible for overseeing all the resources within the cluster. It determines overarching decisions regarding the cluster, such as deploying new pods, identifying and removing inactive pods, and monitoring all the cluster activities for any state changes. It consists of several components dedicated to managing these operations.

Etc: This serves as a distributed key-value store specifically designed for the cluster. It maintains comprehensive information about the cluster's status at any given moment. This includes details about object specifications, the status of each object, the distribution of objects across nodes in the cluster, exposed services for various objects, and more. As a critical database for the cluster, it is recommended to store etcd outside of the server hosting the master node to enhance security. Essentially, etcd functions as the backend infrastructure for the cluster.

Kube Scheduler: This component is tasked with determining the optimal node for running the objects. It scans for unassigned objects and identifies the most suitable node to assign them to.

Controller Manager: Responsible for overseeing controller processes, this component continuously monitors the cluster's state to ensure it aligns with the desired configuration. It verifies the stability of the cluster and the operational status of all the resources.

API Server: Serving as the gateway for interacting with Kubernetes, this component processes commands that specify the desired state of the Kubernetes environment. It acts as the interface between users and the control plane or master node.

Worker Node: This machine hosts all the resources and objects managed by Kubernetes. Users have the option to select specific machines for deploying workloads. Similar to the master node, the worker node also comprises two key components.

Pod: A pod represents a group of one or more containers with similar configurations. Containers within a pod share resources, such as storage and network, and can communicate with each other via localhost.

Kubelet: Acting as an intermediary between the worker node and the control plane, this agent is responsible for provisioning resources required to execute workloads.

Kube Proxy: This networking component facilitates the exposure of pods to various services within the Kubernetes environment.

The Kubernetes architecture is not only a framework for container orchestration but also a fundamental enabler of multi-objective scheduling in 5G environments. Core components, such as the kube scheduler, controller manager, and etcd, directly influence how containerized network functions (CNFs) are deployed, balanced, and optimized across clusters. For instance, kube scheduler decisions govern workload distribution and thereby affect QoS metrics, such as latency, throughput, and reliability [12,13,24]. Recent heuristic approaches, like DRAP [25], emphasize resilience and cost reduction, while mathematical models, such as ILP [24], target optimal placement but face scalability issues. Similarly, AI- and ML-driven schedulers [27,29] enhance proactive scaling and resource prediction, contributing to efficiency and reduced response time. These architecture-scheduling relationships are essential in meeting 5G performance goals, including ultralow latency, energy efficiency, fault tolerance, and high availability. As Tables 8 and 9 illustrate, each scheduling strategy optimizes different parameters—from bandwidth and computation time to flexibility and resilience—highlighting the tradeoffs that must be addressed for effective orchestration. By linking Kubernetes' architectural layers with these optimization metrics, the survey underscores how scheduling choices directly impact the ability to deliver reliable, scalable, and high-performance 5G services.

However, Kubernetes scheduling is not without limitations, including scalability challenges, scheduling delays, and overhead in large-scale 5G deployments [12,13,24].

2.4. Network Slicing

A pivotal feature of 5G networks that distinguishes them from their predecessors is the concept of network slicing, a fundamental aspect illustrated in Figure 4. Network slicing represents a paradigm shift in network architecture, enabling the creation of multiple virtual networks on shared physical infrastructure. This innovation is particularly crucial for 5G, given its diverse use cases and services. By partitioning a physical network into distinct virtual networks, 5G virtualization can efficiently cater to various radio access networks (RANs) or services tailored for specific customer segments. An essential characteristic of network slicing is the complete separation of the control plane and user plane within the network slices. This separation ensures that users experience the network as if it were physically distinct, enhancing flexibility and adaptability. A prominent use case for network slicing in the 5G context is mobile broadband, offering high throughput and low latency. In [30], the authors demonstrate that advantages of 5G network slicing encompass increased bandwidth, improved mobility, heightened security, and enhanced availability.

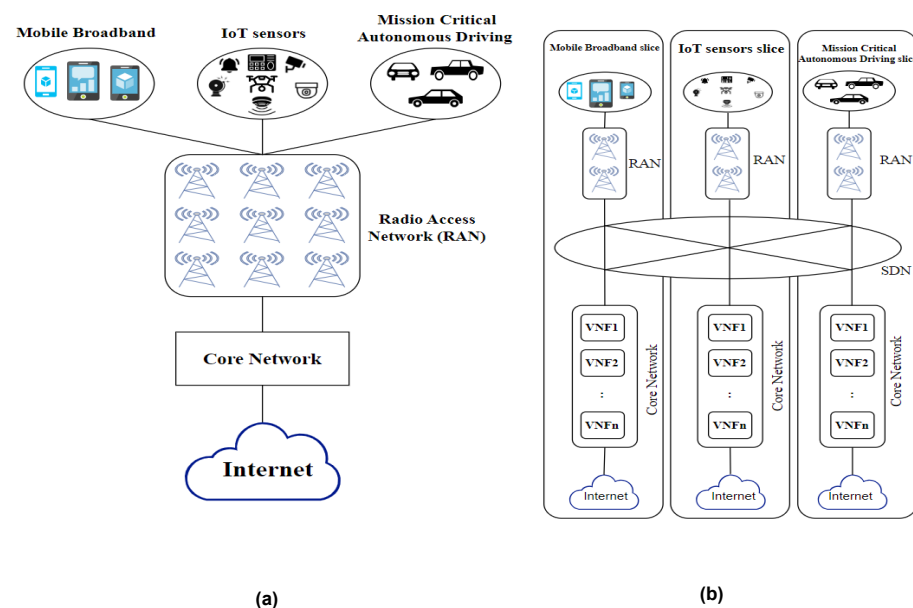


Figure 4. The comparison between (a) traditional network (b) network-slicing implementations.

In the context of 5G networks, software-defined networking (SDN) and NFV play pivotal roles in enabling programmable management and control of network resources [29]. These technologies empower dynamic orchestration and optimization of network elements, contributing to the efficiency and adaptability of 5G infrastructure.

The shift from virtual machines (VMs) and VNFs to containers and cloud-native network functions (CNFs) signifies a significant evolution in virtualization methodologies. Initially, VMs and VNFs facilitated hardware resource virtualization, allowing the deployment of network functions independently of physical infrastructure. However, containers have emerged as a lighter and more portable alternative, boasting faster startup times and improved resource efficiency. Embracing cloud-native principles, CNFs further bolster agility and scalability through microservice architecture, automation, and orchestration, paving the way for a more efficient, scalable, and flexible deployment of network services. Kubernetes plays crucial roles in orchestrating container deployment, scaling, and management across clusters, streamlining the complexities associated with large-scale applications. The adoption of the microservice architecture fosters modularity and encapsulation, enhancing agility and flexibility in software development. In the context of 5G networks, network slicing

transforms network architecture by enabling the creation of multiple virtual networks on shared physical infrastructure, effectively catering to diverse use cases and services.

Network slicing, supported by CNFs and Kubernetes orchestration, introduces diverse scheduling requirements. For example, enhanced mobile broadband slices demand high throughput, while URLLC slices require ultralow latency, and IoT slices emphasize energy efficiency [31,32]. This diversity underscores the need for multi-objective scheduling frameworks capable of balancing competing goals across heterogeneous slices, aligning orchestration mechanisms with the broader performance metrics presented in Tables 8 and 9.

3. Literature Review

Our attention is directed toward investigating multi-objective scheduling algorithms within the Kubernetes framework, particularly in the context of 5G. The scarcity of articles retrieved from the search outcomes is attributed to the novelty of the technology employed, as elucidated in the subsequent sections.

This review has been organized into three sections for better clarity. The first section covers containerized 5G network functions, the second addresses containerized 5G applications, and the third focuses on the Kubernetes scheduler's performance metrics in 5G. This structure enhances the clearness and comprehensiveness of the review.

Our literature review and background provide insights into the predominant terms and metrics associated with multi-objective Kubernetes-scheduling algorithms in the 5G context, differentiating strategies in scheduling, identifying primary simulation tools used, and presenting significant insights and key findings derived from exploring these algorithms.

3.1. Containerized 5G Network Functions

In the coming years, network operators are anticipated to adopt cloud architectures for both edge and core networks [1], with the aim of enhancing efficiency, reliability, and scalability [2].

3.1.1. CNF Orchestration with Kubernetes-Based Approaches

The authors in [28] aimed to validate and evaluate the deployment of an open-source 5G mobile core network, using both virtualized and cloud-native approaches via open-source 5G mobile core network OSM. Initially, each network function (NF) was deployed on a separate virtualized network function (VNF). Subsequently, the transition to the cloud-native approach was made, commencing with the containerization of each NF. Manifest files for the Kubernetes resources involved in the deployment were then created, and, ultimately, the NFs were packaged into a Helm chart.

The authors in [31] propose CIM, a trusted measurement scheme for container integrity measurement. CIM extends the chain of trust to both bare metal containers and virtual machine containers, effectively ensuring the integrity protection of containerized VNFs.

The study in [24] utilizes ILP (integer linear programming) to address the placement of service and network functions in the edge–cloud continuum, recognizing the inherent complexity that requires approximation and heuristic methods. It highlights the potential to manage the problem size effectively in practical settings, allowing for the application of optimal solvers. The approach prioritizes maximizing placement efficiency as a consolidated metric, streamlining the problem by condensing multiple criteria into a singular value. Moreover, it introduces continuity constraints alongside precedence constraints, facilitating cohesive function placement that accounts for interdependencies among nodes.

The authors in [24] propose a dynamic resource allocation and placement algorithm (DRAP) aiming to design and place a simple cloud-native network service. Their approach can help service providers to reduce their infrastructure costs. The proposed DRAP heuristic

algorithm aims to reduce resource utilization while ensuring service availability. It focuses on minimizing the number of nodes needed to place CNF pods (i.e., Kubernetes pods) by adapting the vCPU allocation to each pod. The scalable vCPU allocation permits the algorithm to scale up or down the number of pods based on service availability.

The authors in [32] investigated the use of single-root input/output virtualization (SR-IOV) in cloud-native functions (CNFs) to meet the stringent network demands of 5G applications. Motivated by the success of SR-IOV in (NFV) on virtual machines (VMs), they applied SR-IOV to the charging function (CHF) in a cloud-native 5G architecture. This approach led to a 30% increase in network throughput compared to that of a Kubernetes deployment using Calico, demonstrating the potential of SR-IOV and CPU-pinning in optimizing the performance of cloud-native container-based applications.

While ILP-based solvers, DRAP heuristics, and SR-IOV-based CNF placements demonstrate promising performance improvements, they also introduce significant tradeoffs. ILP-based methods face scalability issues under large-scale 5G workloads; DRAP heuristics can incur high computational costs, and SR-IOV placements are limited by hardware dependencies. These limitations restrict their applicability in highly dynamic and heterogeneous 5G environments.

The work in [33] utilizes Kubernetes infrastructure for hosting various network services, aiming to address the diversity of 5G use cases with maximum flexibility and cost effectiveness. However, limited information is available on the experimental setup and specific results obtained.

Compared to traditional VNFs, CNFs deployed through Kubernetes significantly reduce downtime, enhance responsiveness, and improve scalability, making them highly suitable for 5G network services [32].

The aim of [34] is to demonstrate and validate a proof of concept for a standalone OSM-isolated Kubernetes (K8s) environment for CNF deployment. This paper contributes in three major ways: first, by providing detailed insights into enabling native CNFs within this NFV architecture, addressing gaps, and evaluating its advantages, limitations, and drawbacks; second, by discussing encountered issues and proposing solutions to enhance OSM support in standalone K8s deployments; and third, by conducting performance benchmarking of CNFs using an OSM-K8s testbed and comparing it with an OSM–OpenStack environment deploying VNFs. Adapting the ETSI NFV reference architecture with Open-Source MANO, the study employs Kubernetes to achieve cloud-native objectives in 5G systems, supporting microservice-based architectures. The challenges include integrating Kubernetes deployments within existing VNF-based solutions, like OpenStack.

The authors in [35] evaluate the feasibility of a hardware-accelerated 5G cloud-native virtualized RAN (vRAN), specifically focusing on the distributed unit (DU). The setup uses a Kubernetes cluster, where intensive decoding tasks are offloaded to an FPGA. The evaluation compares the cost, latency, and power consumption of this hardware-accelerated approach to those of a full software stack provided by OAI. The results show potential improvements in offloading latency and reductions in CPU usage and power consumption for large RAN deployments, where multiple radio units (RUs) are pooled on a single DU platform.

The proposed solution in [36] introduces a scalable cloud-native solution for LTE MME (CNS-MME) using microservices to improve autoscaling, resiliency, and resource optimization. It separates the MME's state from its processing tasks and uses VNFs for lightweight, scalable MME functions. Kubernetes, Docker, and Prometheus are employed for orchestration and monitoring, ensuring minimal operational costs. The CNS-MME architecture is evaluated against a monolithic MME, demonstrating better throughput, load balancing, autoscaling, and resource efficiency with an L7 load balancer.

The lessons learned from these studies highlight the potential of Kubernetes and cloud-native deployments for enhancing scalability, efficiency, and resource optimization in 5G networks. Approaches such as containerization, dynamic resource allocation, and optimized function placement proved to be beneficial in reducing infrastructure costs and improving network function performance. Key limitations include the complexity of managing network functions across hybrid environments, especially when integrating with existing virtualized solutions, like OpenStack, and the need for trusted integrity measurement schemes for container security. Additionally, while hardware acceleration and SR-IOV show performance gains, they introduce dependencies on specialized infrastructure that may limit deployment flexibility and increase operational costs.

Table 2 provides a summary of CNF orchestration using Kubernetes-based approaches, focusing on each approach's method, dataset type, testing tools, advantages, and disadvantages.

Table 2. Summary of CNF orchestration with Kubernetes-based approaches.

Reference, Year	Algorithm/Method	Type of Algorithm	Type of Dataset	Testing Tool	Advantage	Disadvantage
[28] (2022)	Kubernetes-based network function orchestration with open-source MANO	KNF-based containerized approach	Service instances	Open-Source MANO (OSM), including Kubernetes	Kubernetes-based 5G core deployment offers a low latency, high availability, and flexible business model, reducing downtime fivefold compared to virtualization.	The cloud-native Kubernetes approach increases complexity, resource demands, and operational costs.
[31] (2021)	Container integrity measurement (CIM)	Static integrity measurement	Comprises experimental data from deploying OAI-based 4G core network functions in bare-metal and VM containers.	NFV MANO and Docker or Kubernetes	The proposed scheme secures containerized VNFs, enhancing trust in 5G SBA while ensuring flexibility, scalability, and efficient resource management.	The CIM scheme increases average CPU usage by 26% and slightly raises memory usage, balancing security with performance at an acceptable cost.
[25] (2022)	ILP (integer linear programming)	Mathematical programming	Various scenarios provide structured datasets for investigating service and function placement optimization in edge-cloud environments.	IBM ILOG CPLEX Studio	ILP solvers excel with small inputs, producing optimal solutions for edge-cloud placement, while continuity constraints enhance realism and modeling accuracy.	IBM ILOG CPLEX Studio incurs high licensing costs, a steep learning curve, and high computational demands for large-scale problems.
[24] (2021)	Dynamic resource allocation and placement of (DRAP) algorithm	Heuristic	Service instances	Kubernetes	The cloud-native approach enhances resilience and reduces costs, using multiple replicas and dynamic vCPU allocation.	The algorithm has only been tested with simple services; complex network services may present unaddressed challenges.
[32] (2022)	SR-IOV	Hardware virtualization technology	Voice and messaging services	Kubernetes	Offloading packet processing to user space with SR-IOV and DPDK reduces overhead on the operating system's kernel, leading to faster processing times and improved responsiveness.	Implementing SR-IOV and DPDK requires expertise and careful configuration, potentially increasing deployment complexity.
[33] (2022)	Modular design for network functions.	Statistical approach for modeling and resolution of resource allocation	Uses Kubernetes infrastructure hosting different network services	Kubernetes	Addresses the diversity of 5G use cases with maximum flexibility and cost effectiveness; improves network functions' availability and resilience	Limited information provided on experimental setup and specific results obtained

Table 2. Cont.

Reference, Year	Algorithm/Method	Type of Algorithm	Type of Dataset	Testing Tool	Advantage	Disadvantage
[34] (2021)	Adapting the ETSI NFV reference architecture and leveraging open-source MANO initiatives, specifically OSM	Full container technology deployment—via Kubernetes—in an NFV architecture	Service instances	OSM-K8s' testbed	Kubernetes, as a key component of the CN approach in 5G systems, drives operational and capital expenditure savings, promoting its adoption.	Early Kubernetes deployments in 5G systems are non-standalone and are often integrated with VNF-based solutions, like OpenStack.
[35] (2021)	Automatic activation of hardware-programmable acceleration resources for Cloud RAN	Resource allocation/optimization	Data related to radio physical layer processes	Kubernetes-based orchestration	Improves performance and energy efficiency by offloading demanding functions to hardware accelerators.	May not fully address the variability in computing demands across different network scenarios.
[36] (2017)	Cloud-native solution for LTE MME (CNS-MME)	Microservice-based cloud-native architecture	Mobile network data	Kubernetes, Docker, Prometheus, and L7 load balancer	Provides 7% higher MME throughput and reduces processing resource consumption by 26%. Offers scalability, autoscaling, and load-balancing features.	Additional complexity of implementing microservices and managing cloud-native infrastructure.

3.1.2. CNF Orchestration with Artificial Intelligence and Machine Learning

The authors in [37] present an artificial-intelligence-based resource-aware orchestration (AIRO) framework in cloud-native environments (CNEs). The AIRO framework leverages the zero-touch service management (ZSM) concept, cloud-native approach, and machine-learning (ML) techniques to efficiently manage network and computation resources. Key contributions include a unified AIRO framework aligned with the ETSI ZSM vision for enabling closed-loop automation and autonomous CNEs, a monitoring and management agent deployed alongside the master node in each cluster to create a single management domain receiving high-level controls from the end-to-end (E2E) management domain, and a simulation platform that mimics Kubernetes (K8s) microservices clouds to evaluate the scalability of the AIRO framework and similar future frameworks.

In [38], the authors propose an edge intelligence method to analyze and predict network data traffic at edge devices, aligning with the 6G vision of softwarized networks. Utilizing the cloud-native enabled OpenFlow over the Trans-Eurasia Information Network (OF@TIEN++) Playground, which connects ten sites across nine Asian countries, they introduce a deep learning method for predicting statistical characteristics of data traffic inflow at edge devices. They collect raw traffic flow data, send it to a visibility center for processing, and use Kubeflow at the orchestration center to develop and train the deep learning model. This model is trained on time-series data to predict traffic characteristics, and its performance is evaluated using root-mean-square error (RMSE) and coefficient-of-determination (R^2) metrics.

The authors in [27] have highlighted the roles of machine learning (ML) and deep reinforcement learning (DRL) in advancing network and service coordination by optimizing resource allocation, scaling, placement, and scheduling, particularly in wireless networks. Their work indicates that these techniques, often facilitated by tools like Kubernetes, significantly enhance network efficiency and enable autonomous decision making, leading to reduced resource waste. However, the authors also point out that some DRL methods rely on a priori knowledge or partial observations, which can limit the model's adaptability to dynamic, real-world conditions.

Recent works, such as DRL-based orchestrators and AIRO, show high adaptability but introduce new challenges. These include model-training overhead, algorithmic instability,

and sample inefficiency, particularly in dynamic environments with partial observability. As such, their deployment remains constrained in practical 5G infrastructures.

The research in [39] aims to enhance the network slice selection process by proposing a framework that incorporates the network slice selection function (NSSF) service within both vertical and horizontal models. This strategy allows handover decisions to be collaboratively managed by the network edge and the user equipment (UE) during network operations. While the network slice selection function decision-aid framework (NSSF DAF) provides a distributed solution for 5G slice selection, it necessitates the integration of various decision-making methods and machine-learning algorithms, potentially increasing complexity and resource demands.

The authors in [40] aim to create a real-time classification model on the network operator's side to identify applications running on the network, predict future traffic patterns, and optimize slice allocation based on anticipated needs. It involves selecting the best supervised ML model for forecasting demand and evaluating the approach in real-world scenarios with actual devices and traffic.

The article in [41] aims to leverage the Kubernetes framework to extract metrics that accurately reflect network utilization for both the RAN and core network, utilize machine learning for predicting future resource requirements based on observed traffic patterns, and establish a proactive scaling mechanism for RAN and core network components utilizing the framework's vertical and horizontal scaling capabilities.

The insights gained from these studies highlight the effectiveness of embedding artificial intelligence and machine learning in cloud-native frameworks to optimize resource allocation, predict traffic, and automate network operations. AI-driven orchestration, deep learning for traffic insights, and reinforcement learning for resource distribution have all shown potential to boost network efficiency, scalability, and responsiveness. However, significant challenges arise, such as the complexity added by machine-learning models, which can intensify computational needs and complicate real-time applications. Furthermore, many AI and machine-learning approaches rely heavily on extensive data and prior insights, which may limit adaptability in dynamic environments. Additionally, integrating multiple machine-learning models within frameworks poses difficulties in maintaining coordination and handling increased resource consumption. Future evaluations should employ standardized benchmarking environments, such as OpenAirInterface [39], OSM-K8s [34], and MEC [42] testbeds, to ensure consistent, fair, and reproducible performance assessments across different schedulers.

Table 3 provides a summary of CNF orchestration approaches using artificial intelligence (AI) and machine learning (ML).

Table 3. Summary of CNF orchestration with artificial intelligence and machine learning.

Reference, Year	Algorithm/Method	Type of Algorithm	Type of Dataset	Testing Tool	Advantage	Disadvantage
[37] (2021)	AI-based resource-aware orchestration (AIRO) framework in CNE	AIRO framework leverages ZSM concept, cloud-native approach, and ML	Service instances	AIRO framework for training ML agents within Kubernetes	The AIRO framework offers robust resource management and service enforcement for telco-grade systems.	The simulation platform introduces uncertainties due to stochastic behaviors, potentially affecting the accuracy of performance predictions in real deployments.
[38] (2021)	Edge intelligence method	Deep learning model	Real network traffic dataset	Kubernetes-based edge cluster with GIST playground control center and packet tracing	Enhances 6G network management with AI and SDN/NFV; accurately predicts edge device data traffic properties	Complex implementation and data-preprocessing challenges; potential for future automation and scalability improvements

Table 3. Cont.

Reference, Year	Algorithm/Method	Type of Algorithm	Type of Dataset	Testing Tool	Advantage	Disadvantage
[27] (2023)	ML and deep reinforcement learning (DRL) for network and service coordination	ML and DRL	Wireless network data	Kubernetes	Enhances resource allocation; optimizes scaling, placement, and scheduling; reduces wasted resources; and allows autonomous decision making	Some existing DRL methods rely on a priori knowledge or partial observations, which can reduce the model's adaptability to real-world scenarios.
[39] (2022)	Network slice selection function decision-aid framework (NSSF DAF)	Dynamic and nonlinear	Data collected from testbed experiments conducted to validate the proposed framework	Kubernetes based on OpenAirInterface (OpenAir-K8s)	The study shows the proposed 5G slice selection solution is standard-compliant and suits IoT and multimedia applications.	Implementing NSSF DAF requires integrating multicriteria decision-making methods and ML algorithms, increasing complexity and resource demands.
[40] (2024)	ML models for traffic classification and prediction in cellular networks	ML algorithm for traffic prediction and slice reallocation in network management	Cellular network traffic flow data used for training and testing the models	Kubernetes	Enables proactive slice reallocation, resulting in up to four times lower latency (jitter) and nearly four times higher throughput compared to those of standard methods while optimizing resource allocation.	The framework's efficiency may depend on accurate real-time predictions, which could be challenging under highly dynamic and unpredictable network conditions.
[41] (2021)	LSTM RNN model to predict network load	ML	Real-world traffic datasets	OAI Kubernetes	Enables automatic scaling based on monitored metrics.	Requires large datasets for training and validation

3.1.3. CNF Orchestration with Service Function Chains (SFCs)

A service function chain (SFC) is an ordered list of VNFs to traverse the traffic, and the classifier decides which traffic to pass through them. In [43], the authors present an SFC controller to optimize the placement of container-based service chains in fog-computing environments, addressing the gap left by multi-access edge computing (MEC)-focused studies. Unlike MEC, which aims to reduce latency by deploying services near end-users, fog computing requires bidirectional communication between edges and clouds due to its hierarchical structure. The proposed SFC controller, implemented as an extension to Kubernetes, ensures necessary bandwidth for these interactions. Evaluations, particularly for Smart City applications, demonstrate that the controller effectively allocates container-based SFCs, conserving bandwidth and reducing end-to-end latency.

The authors in [44] present an optimized network-aware load-balancing algorithm designed for 5G traffic steering in a cloud-native microservice environment, utilizing network traffic data and infrastructure states. Experiments conducted on a Kubernetes-based platform demonstrate that this algorithm enhances traffic management efficiency while reducing capital and operational expenditures (CAPEX and OPEX). However, the complexity introduced by numerous microservices can make the SFC process error prone within the (NFV) ecosystem.

The authors in [45] focused on developing a performance model using non-product-form queueing networks to analyze latency in the packet data unit (PDU) session establishment across interconnected 5G nodes, modeled as G/G/m queues. Additionally, a hierarchical availability model employs reliability block diagrams (RBDs) and stochastic reward networks (SRNs) to probabilistically characterize the failure and repair behaviors of 5G nodes, differentiating between proactive and reactive recovery methods. Using a testbed based on Open5GS, the study implements software routines for estimating service times and evaluating repair times through fault injection techniques.

The ultimate goal is to identify optimal resilient 5G configurations that balance performance and availability requirements, ensuring high availability with minimal downtime.

However, challenges include the complexity of accurately modeling interconnected nodes and the need for extensive data for reliable predictions.

These studies highlight the benefits of optimizing service function chaining and resource allocation in cloud-native environments to achieve reliable, low-latency performance, though challenges remain in managing complex microservices and high computational demands for 5G.

Table 4 summarizes the approaches for CNF orchestration using service function chaining (SFC).

Table 4. Summary of CNF orchestration with service function chains (SFCs).

Reference, Year	Algorithm/Method	Type of Algorithm	Type of Dataset	Testing Tool	Advantage	Disadvantage
[43] (2020)	SFC controller	Network service management	Container-based service chains	Kubernetes	The proposed SFC controller reduces network latency and conserves bandwidth in fog–cloud environments, optimizing resource provisioning for Smart City use cases.	The scheduling decision time is increased by 6 ms per pod, potentially affecting the overall system performance.
[44] (2020)	Optimized network-aware load-balancing algorithm	Load-balancing algorithm	Traffic data and infrastructure states	Kubernetes-based platform	Efficient traffic steering and load balancing in cloud-native SFCs, reducing CAPEX and OPEX while ensuring network performance	Error proneness in service function chaining due to many microservices
[45] (2024)	Non-product-form queueing networks and RBDs/SRN	Proactive scaling using machine learning	Real-world traffic datasets	Kubernetes	Enables efficient management and automatic scaling of network functions	Dependency on effective fault injection setup and analysis

3.1.4. CNF Orchestration with Slicing

In [46], a 5G-ready architecture model and NFV-based network slicing are proposed to provide scalable VNFs and deliver customized 5G slices tailored to customer requirements. Similarly, [47] presents a novel architecture for open-cloud-based 5G communications, treating network slicing as a fundamental component in C-RAN, aimed at enhancing the scalability of existing RAN systems. In the realm of network slicing, each slice possesses unique parameters tailored to its intended purpose, exemplified by its ability to cater to diverse services, such as IoT connectivity or augmented reality, within a single 5G system, providing connectivity with specific requirements related to availability, latency, throughput, and security levels.

In [48], the authors proposed a multi-level scheduling system for network slicing, where a global scheduler sets up slice schedulers based on slice requirements. This approach was implemented and assessed for both single- and multiple-slice deployments. The two-level scheduler was developed using Kubernetes and container technology and deployed on an OpenStack cluster.

In [49], the Setpod scheduler presents an innovative approach to resource scheduling within Kubernetes, aimed at enhancing energy efficiency and maximizing slice deployment. While the Setpod scheduler demonstrates superior performance compared to that of the Kube scheduler, future research is required to explore its adaptability in larger clusters and to include constraints, like bandwidth and latency, for improved quality of service.

In [50], the authors addressed the need for low-latency orchestration in the B5G/6G network–cloud continuum, proposing a novel architectural and scheduling approach. The results show that the new orchestrator reduces scheduling latency by nearly 50% compared to those of Kubernetes and Docker Swarm, with improved self-healing capabilities. However, opportunities remain for further enhancement, including faster container monitoring for failure detection, advanced scheduling criteria (e.g., load balancing and node

affinity), and the introduction of automatic scaling features. Future work could also explore advanced monitoring tools to track resource utilization and trigger scaling based on custom metrics. In the context of ultra-reliable low-latency communication (URLLC), schedulers must achieve deterministic responsiveness and constrained computation times while supporting multi-objective optimization. However, most existing approaches inadequately address these requirements, leaving a gap in achieving predictable real-time service delivery.

In [51], the authors' proposed monitoring framework for 5G network slicing features a scalable architecture with lifecycle management, utilizing Kubernetes for distributed monitoring of network slice performance metrics. This framework incorporates dynamic slice data collectors that are instantiated and deleted with each network slice, allowing for the monitoring of diverse technological domains through a novel, technology-agnostic data collection protocol. It aggregates performance data at the slice level, providing actionable insights to slice owners while ensuring low CPU and memory consumption, even with numerous slices deployed. However, it requires continuous testing to maintain the optimal performance and manage complexities associated with dynamic collectors and standardized metric formats.

Table 5 summarizes CNF orchestration methods focused on network slicing, detailing scheduling and resource management algorithms, testing tools, advantages, and challenges.

Table 5. Summary of CNF orchestration with slicing.

Reference, Year	Algorithm/Method	Type of Algorithm	Type of Dataset	Testing Tool	Advantage	Disadvantage
[46] (2015)	Network store with SDN/NFV-based network slicing	Architecture/platform design	Real-time LTEaaS testbed	OpenStack (with Heat and LXC), OAI, and USRP B210	Enables programmable 5G network slices; supports service-specific customization; leverages NFV/SDN for flexibility	Limited real-world deployment; challenges in meeting real-time deadlines without proper hardware provisioning
[47] (2016)	5G architectural design patterns (distributed shared memory, dedicated data plane, shared control plane, agent-based VNF/VNA software, and SLA-driven RAN)	Architectural design methodology	Not applicable (design focused)	Theoretical framework, SDN/NFV orchestration, and cloud simulation references	Modular, scalable, supports SLA-driven RAN, stateless VNFs, improved fault tolerance, and flexible slicing	High complexity in implementation, dependency on SDN/NFV maturity, and lacks empirical performance metrics
[48] (2019)	Multi-level (global slice) scheduling for network slicing	Heuristics	Measurement data from an in-house developed testbed	Kubernetes and OpenStack cluster	Multi-level orchestration	Limited optimization
[49] (2024)	Setpod scheduler	Scheduling algorithm for network slicing within Kubernetes (K8s)	Virtualized resource data related to network slices within a Kubernetes environment	Kubernetes (K8s) as the platform for virtualization and resource management	Maximizes acceptance ratio for deploying slices while minimizing energy consumption; shows better performance in energy efficiency and average deployment time compared to those of Kube scheduler	Tested only with a small cluster of three machines; does not account for additional constraints, like bandwidth and latency between pods; requires further scalability and adaptability for larger clusters and varying use cases
[50] (2024)	Low-latency resource scheduler (LLRS)	Scheduling algorithm	Resource allocation and scheduling data within a cloud continuum environment	Docker Swarm and Kubernetes as orchestration platforms	Reduces container-scheduling time by nearly 50% in comparison to those of default schedulers in Docker Swarm and Kubernetes, improving the overall system responsiveness and performance.	While it reduces scheduling delay, it may not fully address the stringent QoS demands of URLLC in 5G/6G networks without further optimization or enhancements.
[51] (2022)	Scalable architecture with lifecycle management	Distributed monitoring	Network slice performance metrics	Kubernetes	Supports high scalability for numerous network slices	Needs continuous testing to maintain performance under varying loads

The studies above highlight the effectiveness of network slicing and advanced scheduling for scalability, low latency, and efficient resource management in 5G-ready and beyond-5G architectures. Approaches such as customized slice parameters, multi-level scheduling, and novel schedulers demonstrate substantial improvements in service delivery and energy efficiency and reduced latency. However, limitations remain, including the need for enhanced adaptability in larger clusters, faster failure detection, advanced scheduling criteria, and continuous testing to manage dynamic monitoring and maintain standardized metrics. These challenges underscore the complexity of ensuring consistent, high-performance network slicing across diverse applications and resource demands.

Figure 5 illustrates a taxonomy of CNF orchestration approaches in 5G networks, classified into four primary domains: Kubernetes-based methods, AI/ML-driven solutions, service function chaining, and network slicing. Each category encompasses representative techniques, such as ILP-based scheduling, reinforcement learning, heuristic optimization, and AI-driven adaptive slicing. This structured view highlights the diversity of strategies employed to optimize containerized network functions, reflecting the evolving demands of scalability, efficiency, and responsiveness in 5G environments.

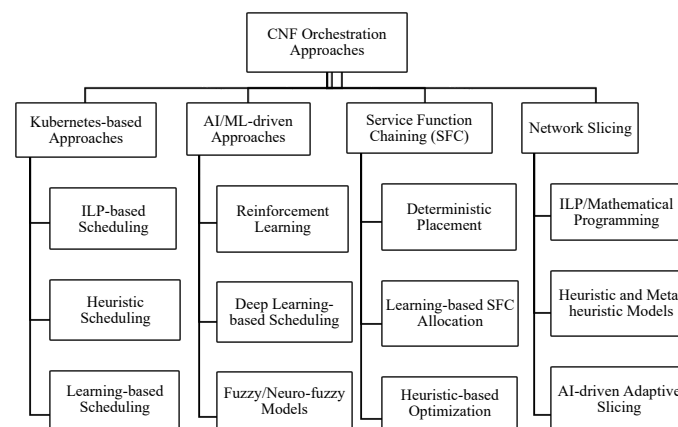


Figure 5. Taxonomy of CNF orchestration approaches for 5G networks.

3.2. Containerized 5G Applications

The transition to cloud-native applications involves breaking down software into smaller components known as microservices, which can then be individually packaged using containerization technology. Containers, a form of virtualization, segregate system instances from user space within a single operating system kernel, unlike traditional virtual machines (VMs), which virtualize entire machines. While this approach offers benefits in communication performance between containers, it also introduces challenges, as all the containers compete for the same system resources, potentially leading to inefficiencies. Containers operate as unique processes within the operating system, each with limited access to resources, such as the file system tree, network interfaces, memory allocation, and disk I/O throughput. Present container service frameworks lack intelligent resource scheduling, often resulting in suboptimal application execution times, underutilized container instances, and a constrained number of deployable apps. The authors in [52] introduce a novel container orchestrator leveraging an integer linear-programming optimization model to address the limitations of traditional orchestration and achieve optimal placement with minimal downtime in hybrid computing environments. Due to the optimization model's time complexity, they also propose an accurate heuristic model that provides near-optimal results in significantly less time, enabling real-time orchestration. The solutions are activated upon detecting or anticipating a failure. Initially, the host takes a snapshot of all the hosted containers and compiles a list of their allocated resources, dependencies,

and constraints. A new placement is then determined, either through migration or re-instantiation, to minimize downtime and access delay latency. This approach enhances QoS by reducing container downtime and meeting the carrier-grade requirements of availability and performance.

In [53], the authors introduce NetMARKS, an extension to the Kubernetes scheduler, designed to optimize scheduling for latency-sensitive workloads. NetMARKS utilizes data from the Istio Service Mesh to inform pod placement decisions based on real-time network metrics, complementing existing Kubernetes scheduler functionalities without conflicts. Through comprehensive testing, the study demonstrates significant improvements in application response time and savings in inter-node bandwidth, highlighting the effectiveness of leveraging service mesh data for enhanced workload scheduling within Kubernetes environments. Although schedulers such as NetMARKS and Setpod report substantial improvements in latency and energy efficiency, these results were validated in specific environments (e.g., OSM-K8s and MEC testbeds). This context dependency highlights that such enhancements cannot be generalized without careful benchmarking. In [54], the authors aim to devise a deployment strategy for applications operating on interconnected computing resources, particularly focusing on the edge cloud. Their primary contributions include the development of a lightweight tool for capturing cluster context data, the creation of a versatile metric server capable of dynamically retrieving various sets of measurements, an enhanced design for a Kubernetes-compatible scheduler that integrates real-time data on edge infrastructure components for more dynamic and sophisticated orchestration and management, and the enhancement of fault tolerance capabilities to offer a more robust platform for critical IoT edge applications. This solution competes effectively with existing standards, and the inclusion of extensive information in the decision-making process potentially reduces the likelihood of rescheduling. In [55], the authors explore optimal edge container selection in distributed devices for virtual aggregation edge computing. They establish an architecture for container selection, construct a task graph, and model collaborative edge-node interactions. Quality metrics guide multi-objective optimization for optimal container combinations using an improved ant colony algorithm. This strategy enhances service quality by reducing the response delay and capacity loss and improving sensitivity and the overall QoS of virtual aggregation edge computing services in heterogeneous environments.

In [56], the authors introduce the Polaris Scheduler2, which is a scheduler designed specifically for edge environments, with a focus on service-level objectives (SLOs). Their contributions include developing a scheduling framework that prioritizes SLOs and takes into account network topology. They also introduce the use of a service graph and a cluster topology graph to represent application SLOs and edge network structures, respectively. Additionally, they provide a set of scheduling plugins that leverage these frameworks to ensure that network SLOs are upheld during the scheduling process. In [57], the authors propose a method to address the joint issues of selecting mobile edge hosts (MEHs) and computing traffic paths. They introduce a graph framework capable of incorporating both network-layer and application-layer performance considerations, effectively consolidating these problems into a single optimization task. To tackle this task, they propose the multi-objective Dijkstra algorithm (MDA) to compute the Pareto front of the multi-objective shortest path (MOSP) model, which allows for accommodating applications with diverse requirements. They validate the performance of the MDA through a hybrid testbed comprising simulative, emulative, and experimental components. Additionally, they demonstrate the integration of the MDA with a 5G-MEC system through the implementation of a controller that manages input retrieval and output application for both the network and application layers. This serves the purposes of MEC host selection and traffic path deter-

mination, migrating applications with minimal impacts on the network performance and user experience. The authors in [42] introduce the ContMEC architecture to enable the user equipment (UE) to offload portions of applications to the MEC infrastructure. The architecture addresses two main requirements by creating a computing cluster at each edge station, implementing hierarchical resource management, and establishing overlapping computing clusters. ContMEC aims to utilize an existing container orchestration system without modifications for easy deployment, despite these systems not originally considering MEC infrastructure. The feasibility of the ContMEC architecture is demonstrated through a proof-of-concept (PoC) implementation using Kubernetes as the unmodified container orchestration system. The study in [58] aims to investigate Kubernetes features for resource scheduling to improve the availability and resilience of NFV networks. The authors specifically target features that address resource scheduling to mitigate failure risks in deployed network services. Additionally, they propose a statistical method to calculate parameters for each CNF, considering future demands and the criticality of the hosted network function. The authors in [59] propose a distributed virtual-network-embedding mechanism for IoT applications on distributed edge infrastructure. The key contributions include a heuristic for optimally mapping VNFs within edge-computing infrastructure to minimize resource utilization, a shortest-path-based algorithm to find the distributed virtual-network-embedding DVNE solution that minimizes round-trip delay with different forward and backward paths for a virtual network (VN), and numerical results demonstrating that the proposed algorithm computes optimal or near-optimal solutions quickly compared to existing studies. The authors in [60] developed a scheduling algorithm for fog computing called the IPerf Algorithm (IPA) to address the need for the efficient use of fog resources in IoT applications with time constraints. They designed IPA to provide quality of service (QoS) by executing applications within their deadlines, taking into account the network status and estimating job completion times. They implemented IPA as an extension to the Kubernetes default scheduler and tested various configurations of IPA against other proposals. They found that one configuration, IPA-E, significantly improved job execution within deadlines by at least 30% compared to other methods, demonstrating the critical role of scheduling algorithms in QoS provision and the importance of considering network status and job completion estimates in scheduling decisions. Sophisticated schedulers, such as Polaris2 and IPA-E, while claiming latency reductions and improved SLO awareness, operate under stochastic or approximation policies that may undermine predictability. A comparative review of the convergence behavior, resource utilization overhead, and scheduling delays indicates that while these approaches offer significant benefits, they must also be balanced against operational drawbacks.

Table 6 provides a comparative overview of prominent multi-objective scheduling algorithms applied in Kubernetes for 5G environments. The table highlights how each method addresses critical factors, such as scalability, latency, energy efficiency, operational overhead, and real-time responsiveness. By contrasting ILP-based solvers, heuristic approaches, like DRAP, hardware-assisted placements (SR-IOV), and advanced AI/ML-based orchestrators, such as DRL, AIRO, NetMARKS, Polaris2, and IPA-E, the table emphasizes the tradeoffs among computational complexity, adaptability, and performance in dynamic 5G deployments.

The default Kubernetes scheduler is effective for general workloads but demonstrates clear limitations in multi-objective optimization and responsiveness. In 5G scenarios, this leads to suboptimal performance in latency-sensitive or resource-constrained contexts, underscoring the need for custom scheduling enhancements.

Table 7 provides a summary of various Kubernetes-scheduling algorithms tailored for 5G applications/use cases.

Table 6. Comparative overview of multi-objective scheduling algorithms in Kubernetes for 5G environments.

Algorithm/Scheduler	Scalability	Latency	Energy Efficiency	Operational Overhead	Real-Time Responsiveness
ILP-based solvers [24]	Low (NP, hard; impractical at scale)	Moderate	Neutral	High (computational cost)	Poor (slow convergence)
DRAP heuristics [25]	Moderate	Improved	Neutral	Low–moderate	Limited (heuristic approximations)
SR-IOV CNF placements [32]	High (hardware assisted)	Very low (fast throughput)	Moderate	Hardware dependent	Good (if hardware is available)
DRL-based orchestrators [27]	High (learning adaptation)	Improved	Varies (depending on training)	High (training cost and instability)	Moderate (depending on observability)
AIRO [37]	Moderate	Improved	Improved	High (complex orchestration)	Limited under dynamic conditions
NetMARKS [53]	Moderate	Improved	Improved	Low–Moderate	Context dependent (testbed specific)
Setpod scheduler [49]	Moderate	Improved	Improved	Moderate	Context dependent
Polaris2 [56]	High	30% latency reduction	Neutral	Moderate (approximation methods)	Limited predictability
IPA-E [60]	High	Reduced latency	Neutral	Moderate–high (network aware)	Possible scheduling delays

Table 7. Summary of Kubernetes schedulers for containerized 5G applications.

Reference, Year	Algorithm	Type of Algorithm	Type of Dataset	Testing Tool	Advantage	Disadvantage
[52] (2022)	Edge-computing-enabled container migration/re-instantiation (EC2-MRI)	Real-time heuristic	IOT applications	Testbed based	By optimizing container placement, applications can execute faster, leading to better performance and shorter execution times.	Underutilized container instances lead to inefficient use of resources.
[53] (2021)	NetMARKS represents an innovative approach to Kubernetes pod scheduling, utilizing advanced techniques.	Dynamically collected network metrics through the Istio Service Mesh	Different workloads and processing layouts	Kubernetes	NetMARKS optimizes Kubernetes scheduling for 5G workflows, saving up to 50% of the inter-node bandwidth and reducing response times by 37%.	Its drawback is the lack of a detailed experimental setup and specific results, hindering evaluation and reproducibility.
[54] (2019)	Dynamic orchestration of edge-native applications on 5G	The algorithm dynamically coordinates edge-native 5G applications, considering physical, operational, network, and software parameters.	Real-time information about edge infrastructure components	Kubernetes	The strategy broadens node capabilities but needs improvement in scheduling speed and efficiency for demanding deployments.	Enhancements could include empowering users to fine-tune scheduling strictness by specifying the degree of influence that assigned parameters have on the final score.
[55] (2023)	Two-population ant colony algorithm to solve the container resource allocation problem	Dynamic multi-objective algorithm	Real-time sensing data	Kubernetes and Docker	The goal is to attain the best allocation of edge container resources for distributed devices in edge-computing scenarios, ensuring effective utilization.	The proposed model and algorithm introduce complexity, requiring advanced implementation and oversight.
[56] (2022)	Polaris Scheduler is designed to be SLO aware.	Heuristic algorithm	Real-time collected data	Kubernetes	The scheduler models edge topology as a cluster topology graph, capturing network quality for accurate environment representation and improved microservice placement.	The added network QoS and dependency considerations increase resource requirements, impacting scheduling efficiency.
[57] (2023)	Multi-objective Dijkstra algorithm (MDA)	Multi-objective Dijkstra algorithm (MDA)	VideoLan application between multiple multi-access edge-computing (MEC) hosts (MEHs)	Testbed based on AdvantEDGE and Kubernetes	The results show that MDA can migrate applications with minimal impacts on network performance and user experience, addressing the challenges of mobility and resource allocation in MEC systems.	Implementing MEC entails managing edge infrastructure, incurring high costs, and raising security and privacy concerns due to processing sensitive data at the edge.
[42] (2022)	Container orchestration system for the PoC implementation of ContMEC	ContMEC's algorithm involves creating computing clusters per edge station, implementing hierarchical resource management, and deploying overlapped computing clusters.	The control traffic in Kubernetes for container deployment and health checks occurs between the cluster master and worker nodes, as well as between the cluster master and UEs when joining the computing cluster in ContMEC.	Kubernetes	The ContMEC architecture efficiently offloads applications to MEC servers, promoting scalability and resource sharing, as demonstrated in a Kubernetes-based PoC.	Implementing applications as container clusters can introduce management and maintenance complexity, potentially increasing overhead in orchestration and monitoring.
[58] (2022)	KRS: Kubernetes resource scheduler for resilient NFV networks	Kubernetes resource scheduler	10 to 20 CNFs deployed at each node	Kubernetes	The KRS algorithm allocates memory and CPUs for critical CNFs, leverages Kubernetes customization, and optimizes CNF coexistence to improve stability post deployment.	Optimizing KRS involves mastering statistical methods, GAs, parameter tuning, continuous adaptation for resilience.
[59] (2023)	sV-VNM algorithm	Round-trip delay minimization	IoT-based applications	Kubernetes and OpenStack	The proposed algorithm offers time-efficient DVNE solutions for IoT applications with strict delay requirements, optimizing resource usage and ensuring timely service delivery, cost savings, and improved network performance.	The worst-case complexity of the proposed algorithm is $O(kn^3)$, which may limit scalability or numerous VNE requests, necessitating efficient complexity management.
[60] (2021)	IPerf algorithm (IPA)	Network-aware scheduling algorithm	200 jobs	Kubernetes	The IPA improves QoS by executing applications within deadlines, considering the network status, and estimating job completion times.	The current limitations and complexity of IPA necessitate improvements in prediction calculations, advance scheduling, and the inclusion of additional network metrics.

3.3. Performance Metrics of the Kubernetes Scheduler in 5G

The outlined metrics provide a comprehensive framework for the evaluation of Kubernetes scheduling in the dynamic landscape of 5G environments. The availability underscores the importance of consistent accessibility and reliability, while the performance strives for equilibrium through effective load management, resource optimization, and adherence to affinity rules tailored for the specific demands of 5G applications. The save bandwidth parameter focuses on efficiency by employing data compression and streamlined communication strategies. The response time places a premium on swift system responsiveness, achieved through the minimization of delays and optimized resource management. Flexibility introduces dynamic resource allocation and scaling, ensuring the adept handling of diverse workloads. The deployment time represents the duration required for efficient orchestration and container deployment. Cost considerations take into account economic factors associated with resource utilization and infrastructure. Resilience tackles the system's recovery from failures, employing mechanisms like fault tolerance. The time of scheduling designates the allocated duration for executing scheduling processes. The metaphorical description of the cluster temperature encompasses the overall load and resource utilization, including strategies like load balancing and dynamic scaling. Energy consumption pertains to optimized power usage through strategies such as dynamic resource allocation. Fault tolerance guarantees system reliability, especially in the face of failures, while load balancing involves the even distribution of workloads to sustain the optimal performance. Ultimately, QoS (quality of service) ensures the fulfilment of performance expectations, assuring users a reliable and satisfactory experience in the ever-evolving 5G landscape.

The key metrics in 5G networks encompass various aspects crucial for evaluating performance and user experience. Latency measures the delay in communication, while the served traffic quantifies the data volume exchanged between users and devices. Throughput denotes the rate of data transfer, while computation time indicates processing durations for tasks. QoE (quality of experience) reflects the overall user satisfaction with network services. Execution time gauges the task completion duration, while security ensures data protection. QoV (quality of video) assesses visual content fidelity, and the MOS (mean opinion score) subjectively evaluates communication quality. These metrics collectively provide insights into the effectiveness, reliability, and usability of 5G networks and services.

Table 8 displays the various parameters that govern the functionality of scheduling algorithms in the context of 5G with Kubernetes.

Table 8. Performance metrics of multi-objective Kubernetes scheduling algorithms in 5G.

Metric	Description
Availability	Availability refers to the system's ability to ensure consistent accessibility of applications [12,29,35].
Reliability	Reliability refers to the consistent and dependable performance of the network in providing connectivity, data transmission, and other services to users [40].
Performance	Performance is achieved by balancing loads, optimizing resources, and applying affinity rules to ensure effective allocation for meeting the low-latency and high-throughput demands of 5G applications, enhancing the overall system responsiveness and scalability [12].
Bandwidth	The bandwidth requirement involves efficient data compression, minimizing unnecessary transfers and optimizing communication patterns [55,56].
Response Time	The response time refers to the duration taken for the system to process and respond to a request. It is achieved by minimizing workload allocation delays, optimizing resource management, and prioritizing low-latency communication [29,55].

Table 8. Cont.

Metric	Description
Flexibility	Flexibility involves dynamically allocating resources, adapting scaling, and supporting diverse workloads, enabling rapid adjustment to changing demands and ensuring versatility in handling various applications within the 5G environment [33].
Deployment Time	Deployment time refers to the duration required for efficient orchestration, streamlined container deployment processes, and optimized resource allocation, ensuring swift and timely deployment of applications within the dynamic environment [48].
Cost	Cost refers to the economic considerations associated with resource utilization, infrastructure expenses, and efficiency in managing computing resources [33].
Resilience	Resilience refers to the system's capability to withstand and recover from failures or disruptions by employing mechanisms such as fault tolerance, automatic workload recovery, and dynamic resource reallocation [33].
Scheduling Duration	Scheduling duration refers to the duration taken for the execution of scheduling processes, including workload allocation decisions, resource assignments, and real-time decision making [54].
Cluster Temperature	Cluster temperature metaphorically describes the overall load and resource utilization within the cluster. It refers to the strategies, such as load balancing and dynamic scaling, employed to maintain optimal temperatures by preventing resource bottlenecks and ensuring efficient performance [54].
Energy Consumption	Energy consumption refers to the amount of power utilized by the system, which is optimized through strategies like dynamic resource allocation and load-aware scheduling [61].
Fault Tolerance	Fault tolerance the system's ability to withstand and recover from failures by employing mechanisms such as pod replication and dynamic workload rescheduling, ensuring uninterrupted operation and reliability in the face of node or pod failures within the dynamic 5G environment [12].
Load Balancing	Load balancing involves distributing workloads evenly across cluster nodes, optimizing resource usage, and preventing bottlenecks, ensuring efficient allocation of computing resources and maintaining optimal performance in the dynamic 5G environment [55].
Quality of Service (QoS)	QoS (quality of service) refers to the system's ability to meet the performance expectations of applications, ensuring optimal resource allocation, low latency, and high throughput to deliver a reliable and responsive user experience [55].
Latency	It represents the delay experienced during communication and is typically measured in milliseconds (ms) [57,59].
Served Traffic	This refers to the volume of data transmitted over the network to and from users or devices within a specific period. It encompasses all the data packets sent and received, including voice calls, video streams, text messages, and internet browsing, among others [44,62].
Throughput	Refers to the rate at which data can be transferred over the network within a given period, typically measured in bits per second (bps) or megabits per second (Mbps) [57].
Computation time	This refers to the duration it takes for network devices or systems to process and analyze data, execute tasks, and generate responses. It encompasses various processes, such as data transmission, reception, signal processing, protocol handling, and application execution [42].
Quality of Experience (QoE)	This refers to the overall satisfaction and perception of users regarding the performance, reliability, and usability of services and applications delivered over 5G networks [27].
Execution Time	This refers to the duration taken to complete a specific task or operation within the network infrastructure or on end-user devices. It encompasses the time required for processing user requests, executing network functions, and delivering services or applications over the 5G network [59].
Security	This refers to the measures and protocols implemented to safeguard the integrity, confidentiality, and availability of data and communications within the 5G network [31].
QoV	Quality of video (QoV) refers to the level of excellence or fidelity in visual content delivered over 5G networks. It encompasses factors such as resolution, frame rate, compression efficiency, latency, and reliability of the video stream [39].
MOS Mean Opinion Score	This refers to a subjective measure used to assess the perceived quality of communication services, including voice and video, experienced by users [39].

Table 8. Cont.

Metric	Description
Downtime	Ensuring minimal downtime is crucial for maintaining high availability and performance levels, especially in situations where uninterrupted service is essential [52].
Makespan	This is the period between the starting time of the execution and the completion time of a set of tasks [24,26].
Efficiency	Efficiency refers to the optimal orchestration and management of network resources to achieve high performance and service quality [58].

Table 9 outlines the parameters associated with the multi-objective algorithms reviewed in the context of 5G with Kubernetes.

Table 9. The parameters in the reviewed multi-objective algorithms in 5G with Kubernetes.

Scheduling Strategy	Dynamic	Dynamic	Dynamic	Dynamic	Dynamic	Dynamic
Single/Multi-Objective	Multi	Multi	Multi	Multi	Multi	Multi
Mean Opinion Score						✓
Reliability					✓	
Scalability				✓		
Resource Utilization	✓	✓	✓	✓		✓
QoV						
Security						
Execution Time						
QoE						
Computation Time						
Throughput	✓			✓	✓	
Latency	✓				✓	
Served Traffic						✓
Efficiency		✓	✓		✓	
Makespan						
QoS						
Load Balancing				✓		
Fault Tolerance						
Energy Consumption			✓			
Cluster Temperature						
Scheduling Duration						
Resilience						
Downtime						
Cost			✓			
Deployment Time						
Flexibility		✓				
Response Time						
Bandwidth						
Performance						✓
Availability						
Reference	[32]	[33]	[35]	[36]	[37]	[38]

Table 9. Cont.

Scheduling Strategy	Dynamic	Dynamic	Dynamic	Dynamic	Dynamic	Dynamic	Dynamic
Single/Multi-Objective	Single	Multi	Multi	Multi	Multi	Single	Multi
Mean Opinion Score							
Reliability							
Scalability							
Resource Utilization		✓	✓	✓			
QoV							
Security							
Execution Time			✓				
QoE							
Computation Time				✓			
Throughput							
Latency	✓	✓	✓				
Served Traffic							
Efficiency				✓			
Makespan							
QoS							
Load Balancing							✓
Fault Tolerance							✓
Energy Consumption							
Cluster Temperature						✓	✓
Scheduling Duration							
Resilience							
Downtime							
Cost							
Deployment Time					✓		
Flexibility							
Response Time					✓		✓
Bandwidth					✓		
Performance							
Availability							
Reference	[50]	[51]	[59]	[60]	[53]	[48]	[54]
							[55]
							[56]

Table 9. Cont.

Scheduling Strategy	Dynamic	Dynamic	Static	Dynamic	Dynamic	Dynamic	Dynamic
Single/Multi-Objective	Multi	Multi	Multi	Multi	Multi	Multi	Multi
Mean Opinion Score							
Reliability							
Scalability							
Resource Utilization							
QoV							
Security							✓
Execution Time						✓	
QoE							
Computation Time					✓	✓	
Throughput					✓		
Latency					✓		
Served Traffic	✓						
Efficiency						✓	
Makespan							
QoS	✓						
Load Balancing				✓			
Fault Tolerance							
Energy Consumption		✓					
Cluster Temperature							
Scheduling Duration		✓					
Resilience							
Downtime						✓	
Cost	✓						
Deployment Time			✓				
Flexibility							
Response Time				✓			
Bandwidth							
Performance						✓	✓
Availability						✓	
Reference	[63]	[61]	[28]	[34]	[57]	[42]	[52]
			✓				[59]
							[31]

Table 9. Cont.

Scheduling Strategy	Dynamic	Dynamic	Dynamic	Dynamic
Single/Multi-Objective	Multi	Single	Multi	Multi
Mean Opinion Score	✓			
Reliability				
Scalability				
Resource Utilization	✓			
QoS	✓			
Execution Time				
QoE				
Computation Time				
Throughput				
Latency				
Served Traffic				
Efficiency				
Makespan		✓	✓	
QoS	✓			
Load Balancing				
Fault Tolerance				
Energy Consumption				
Cluster Temperature				
Scheduling Duration				
Resilience				
Downtime				✓
Cost				
Deployment Time				
Flexibility				
Response Time				
Bandwidth				
Performance				
Availability				
Reference	[39]	[26]	[25]	[58] ✓

3.4. Limitations of the Current Literature

Although this review systematically examines multi-objective Kubernetes scheduling for 5G, several limitations remain. The study synthesizes findings from peer-reviewed publications indexed in digital libraries, such as IEEE, Springer, and ScienceDirect, which excludes gray literature (e.g., GitHub implementations, CNCF white papers, and OSM requirements) that may provide practical insights. Despite applying structured inclusion and exclusion criteria, some subjectivity persists in categorizing algorithms across overlapping typologies, particularly among heuristic, hybrid, and ML-driven approaches. Furthermore, while numerous studies address critical objectives, such as latency, energy efficiency, or throughput, most focus on isolated aspects without proposing unified frameworks capable of balancing these conflicting goals in real-world deployments. Much of the evidence relies on simulation or testbed studies, raising concerns about applicability to large-scale 5G networks. Limited attention has also been given to predictive AI-based scheduling, interoperability with standards like Open RAN, and validation across heterogeneous infrastructures. These limitations highlight opportunities for future surveys to broaden source coverage, apply more formal evaluation frameworks, and validate findings through empirical 5G testbeds, leading to more comprehensive, adaptive, and practically validated solutions.

4. Advantages and Disadvantages of Integrating 5G Technology with Kubernetes

Kubernetes provides a robust framework for managing and orchestrating containerized applications, making it well suited for the dynamic and distributed nature of 5G networks. However, while Kubernetes brings several benefits, such as scalability, flexibility, and automation, it also poses challenges in terms of complexity, security, and integration with legacy systems. In this discussion, we explore the advantages and disadvantages of Kubernetes in the context of 5G technology deployments.

Advantages of Kubernetes in 5G

1. **Dynamic Scalability for 5G Traffic:** Kubernetes enables operators to dynamically add/remove nodes to manage fluctuating 5G traffic, such as ultra-reliable low-latency communication (URLLC) and enhanced mobile broadband (eMBB) services [64];
2. **Automation for CNFs:** Kubernetes automates CNF deployment, scaling, and recovery across distributed 5G infrastructure, reducing manual intervention in handling core and RAN functions [65];
3. **High Availability for 5G Services:** Ensures uninterrupted 5G services by restarting failed pods, reallocating workloads, and distributing slices to maintain ultralow latency requirements [66];
4. **Flexibility Across 5G Deployments:** Supports hybrid and edge–cloud environments, enabling orchestration of 5G functions across on-premises sites, cloud, and edge nodes for network slicing [44];
5. **Security for Telecom-Grade Workloads:** Role-based access control, encrypted node communication, and fine-grained policies help to secure 5G CNF deployments against evolving threats [67].

Disadvantages in 5G Integration

1. **Complexity for Telecom Operators:** Running Kubernetes in multi-domain 5G networks requires deep expertise in telecom protocols and distributed systems, creating adoption barriers [68];
2. **Resource Overhead in 5G Core/RAN:** Kubernetes' control plane consumes significant CPU/memory, which may reduce efficiency in dense 5G core deployments [69];
3. **Operational Complexity in Large 5G Networks:** Maintaining Kubernetes clusters across heterogeneous RAN, edge, and core locations adds monitoring and troubleshooting burdens [5];
4. **Security Risks in 5G Environments:** Expanded attack surfaces, including insecure container images or poorly configured policies, pose risks to critical 5G workloads [67];
5. **Vendor Lock-in for Telecom Clouds:** Reliance on cloud-managed Kubernetes services risks limiting portability of 5G network functions across multi-cloud environments [23].

Challenges for 5G Service Providers

1. **Policy Control Across Protocols:** Difficulty in applying consistent scheduling and policy control across mixed traffic (4G protocols like SIP and SCTP vs. 5G NR slicing) [64];
2. **Security Enforcement Across Layers:** Ensuring end-to-end security across multi-layer 5G CNF architectures remains a challenge [67];
3. **Limited Visibility of Network Flows:** Difficulty in monitoring intra-slice traffic, making optimization of 5G CNF scheduling less transparent [38];
4. **Revenue Model Limitations:** Dependence on legacy 4G billing systems while deploying standalone 5G cores delays ROI for operators [70].

5. Comparing Container Orchestration Solutions for Modern Deployment Needs

Choosing the right container orchestration solution is critical to meet the demands of modern deployment environments, particularly in the context of 5G's high-throughput, low-latency, and edge-oriented requirements. While Kubernetes has emerged as the most widely adopted platform due to its scalability, flexibility, and ecosystem support, alternatives, such as Docker Swarm and Nomad, offer simplicity and ease of use for smaller or less complex deployments, whereas Mesos and Apache Aurora provide stronger support for large-scale, enterprise-grade workloads. Platforms, like OpenShift and Rancher, and managed services, such as Amazon ECS, Google Kubernetes Engine (GKE), Azure Kubernetes Service (AKS), and IBM Cloud Kubernetes Service, deliver enhanced capabilities by integrating orchestration with additional tools for automation, monitoring, and security. As shown in Table 10, the choice among these solutions depends largely on the deployment context: For instance, 5G-native scenarios require orchestration frameworks that can balance scalability and responsiveness at the edge while minimizing latency and resource overhead [62].

Table 10. Comparing alternative container orchestration solutions to Kubernetes.

Feature	Development	Application Suitability	Infrastructure	Setup
Docker Swarm	Simple setup and user-friendly interface	Smaller deployments and teams with less experience	Docker	Simple and easy to set up
Nomad	Easy to use, lightweight, and flexible	Smaller teams; does not require complex setup/configuration	Any infrastructure (public/private clouds and bare metal)	Straightforward setup: no complex configuration needed
Mesos	Modular architecture integrated with different runtimes	Teams with advanced needs and expertise to manage complexity	Any infrastructure (flexible)	Requires expertise for setup and configuration
OpenShift	Additional features/tools for simplified management	Teams looking for a more complete container platform	Kubernetes-based platform can run on any infrastructure	May require more setup and configuration effort
Rancher	Complete container management platform that is easy to use	Teams needing a complete container platform and preferring managed service	Kubernetes and other container orchestration tools as a service	Easy to set up and user-friendly interface
Amazon ECS	Fully managed container service that is simple and cost effective	Teams already using AWS and looking for fully managed solution	AWS cloud	Fully managed by AWS; simple setup and management
Google Kubernetes Engine (GKE)	Fully managed Kubernetes service that is scalable and secure	Teams already using GCP and looking for fully managed Kubernetes	Google Cloud Platform (GCP)	Fully managed by Google; easy deployment and management
Microsoft Azure Kubernetes Service (AKS)	Fully managed Kubernetes service; simple deployment on Azure	Teams already using Azure and looking for fully managed Kubernetes	Microsoft Azure cloud	Fully managed by Microsoft; easy deployment and management
IBM Cloud Kubernetes Service	Fully managed Kubernetes service; scalable and secure	Teams already using IBM Cloud and looking for fully managed Kubernetes	IBM Cloud	Fully managed by IBM; easy deployment and management
Apache Aurora	Advanced and customizable option for container orchestration	Teams with advanced needs and expertise	Flexible: runs on any infrastructure	Requires expertise for setup and configuration

Table 10 presents alternative container orchestration solutions to Kubernetes.

Figure 6 presents a taxonomy of Kubernetes scheduling for 5G. It maps core components, scheduling techniques, and optimization goals, showing how Kubernetes's architecture connects with algorithmic strategies to address latency, energy, and resource efficiency in 5G environments.

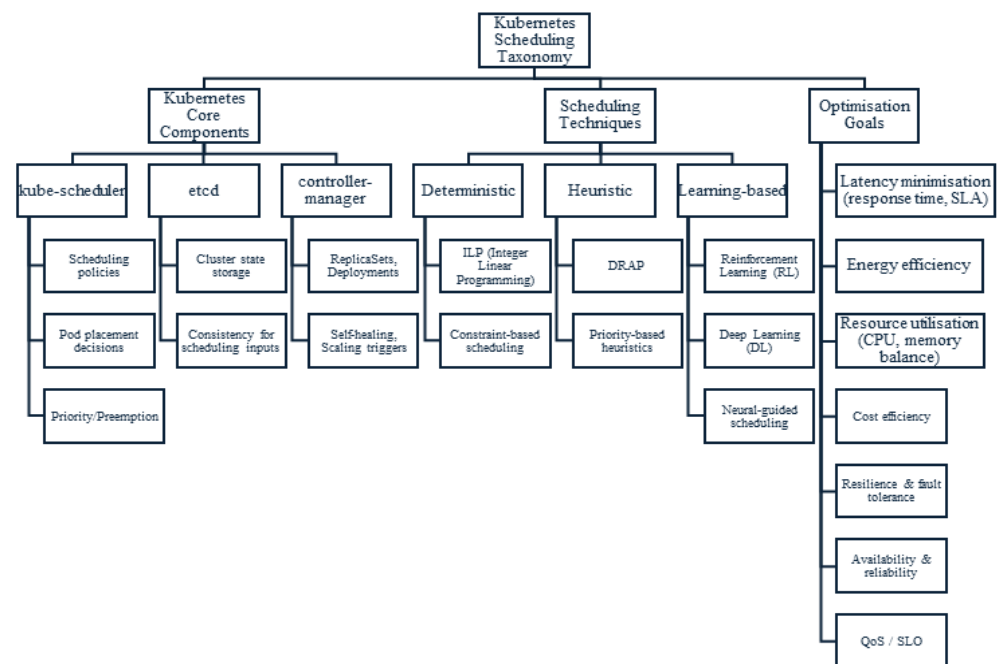


Figure 6. Taxonomy of Kubernetes-scheduling approaches for 5G networks.

6. Applications Enhanced by Kubernetes in 5G Environments

In the context of 5G, Kubernetes offers several advantages for applications that require high availability, scalability, and efficient resource management [64]. Here are some types of applications that benefit from using Kubernetes in a 5G environment:

1. **Service Orchestration and Automation:** 5G networks require complex service orchestration to efficiently manage resources and deliver services to end-users. Kubernetes can orchestrate and automate various network functions, such as (VNFs) and containerized network functions (CNFs), ensuring smooth service delivery and dynamic scaling to meet changing demands;
2. **Network Slicing:** 5G enables network slicing, where a single physical network is partitioned into multiple virtual networks to support different use cases with varying requirements. Kubernetes can manage the deployment and scaling of applications within each network slice, ensuring isolation and efficient resource utilization;
3. **Edge Computing:** Kubernetes is well suited for deploying and managing applications at the network edge, where low latency and high performance are critical. Edge applications in 5G networks, such as IoT services, video analytics, and augmented reality (AR) applications, benefit from Kubernetes's ability to manage containers close to end-users, reducing latency and improving responsiveness;
4. **Workflow Schedulers:** Workflow schedulers, such as Airflow or Argo, are essential for orchestrating complex data-processing pipelines and workflows in 5G environments. Kubernetes can manage the execution of workflow tasks across distributed environments, ensuring reliability, scalability, and efficient resource utilization;
5. **Real-time Analytics and Monitoring:** 5G networks generate large amounts of data that require real-time analytics and monitoring to optimize network performance and ensure quality of service (QoS). Kubernetes can deploy and manage analytics and monitoring applications, such as Prometheus and Grafana, to collect, analyze, and visualize network data in real time;
6. **Content Delivery Networks (CDNs):** CDNs in 5G networks benefit from Kubernetes's ability to dynamically scale and distribute content delivery nodes closer to end-users.

Kubernetes can deploy and manage CDN nodes across distributed edge locations, optimizing content delivery and reducing latency for multimedia and streaming services.

However, to properly evaluate these orchestration tools in the context of 5G, it is necessary to move beyond generic deployment features and assess them against stringent requirements, such as scalability, low latency, and edge-native orchestration. Kubernetes has been validated in telecom-grade deployments for containerized network functions (CNFs) and edge computing [20,28,35,57], offering support for ultralow latency, resilience, and multi-objective scheduling strategies [12,13,24]. For instance, Kubernetes-based frameworks enable network slicing [47,50] and proactive resource allocation in mobile-edge-computing (MEC) scenarios, directly addressing performance goals, such as throughput and response time. In contrast, lighter solutions, like Docker Swarm or Nomad, may offer simplicity but lack the maturity to handle large-scale, latency-sensitive 5G applications. More complex options, such as Mesos and Apache Aurora, can manage distributed systems, yet they introduce higher operational overhead without the integrated CNF support found in Kubernetes. Managed services and extended platforms, like OpenShift, GKE, or AKS, enhance Kubernetes' capabilities but raise tradeoffs in cost, interoperability, and vendor lock-in [25,69]. By situating alternatives within 5G-specific KPIs, such as scalability, latency, and energy efficiency, this survey highlights why Kubernetes remains the dominant orchestrator for next-generation telecom networks.

7. Research Gaps and Future Directions

The future landscape of research in Kubernetes resource management holds significant promise, particularly in the enhancement and proposition of innovative resource management algorithms. At present, Kubernetes resource management relies predominantly on optimization-modeling frameworks and heuristic-based algorithms. Future research endeavors can explore untapped potential in this realm, aiming to propose more effective and efficient algorithms that align with the evolving demands of distributed, heterogeneous environments. The challenges posed by managing complex, dynamic workloads provide a focal point for future investigations, necessitating the development of sophisticated algorithms and techniques. Key areas of exploration may include refining workload placement, resource allocation, and load-balancing strategies, as well as venturing into novel approaches to containerization and virtualization. This section elucidates the primary challenges encountered and proposes future recommendations within the domain of multi-objective scheduling algorithms using Kubernetes in the context of 5G.

7.1. Key Challenges in Multi-Objective Scheduling for 5G with Kubernetes

Kubernetes resource management optimization approaches are typically grouped into five key objective functions: dynamic workload variation, heterogeneous infrastructure, quality-of-service (QoS) requirements, real-time responsiveness, and energy efficiency. However, the challenge lies in simultaneously optimizing these often-conflicting objectives, particularly given the varying environments, schemes, and constraints involved. The context and scope of virtual resource placement management play crucial roles when selecting the appropriate problem setting for Kubernetes resource optimization.

1. **Dynamic Workload Variation:** 5G networks exhibit diverse and dynamic workloads. Adapting scheduling algorithms to handle these variations efficiently is a significant challenge. Inconsistent workload handling may lead to suboptimal resource utilization and compromise performance objectives [20,35];
2. **Heterogeneous Infrastructure:** The presence of diverse hardware in 5G networks requires scheduling algorithms to account for heterogeneity for optimal resource

- allocation. Failure to address hardware diversity can result in inefficient resource utilization and hinder the achievement of scheduling objectives [28,38];
3. Quality-of-Service (QoS) Requirements: Multi-objective scheduling must balance conflicting QoS requirements for different applications, including latency, throughput, and reliability. Failure to meet QoS targets may result in degraded application performance and user dissatisfaction [12,27,29];
 4. Real-time Responsiveness: The need for real-time responsiveness in 5G applications requires scheduling algorithms to make rapid decisions based on changing conditions. Delays in decision making can lead to performance bottlenecks, especially for applications with stringent latency requirements [25,54];
 5. Energy Efficiency: Efficient energy consumption is crucial in 5G networks, necessitating scheduling algorithms to consider the energy impacts of resource allocation decisions. Inefficient energy usage can contribute to increased operational costs and environmental concerns [68].

While significant strides have been made in addressing these challenges, several gaps remain in the research landscape. A primary issue is the lack of unified frameworks that can simultaneously optimize all five objectives in a scalable and efficient manner. Existing studies often focus on isolated objectives or simplified models but fail to account for the complex interactions between these objectives in real-world 5G deployments. Additionally, there is a need for more adaptive and resilient scheduling algorithms that can dynamically respond to unforeseen network conditions while maintaining optimal performance across all the parameters.

Although Kubernetes scheduling for 5G networks has seen significant advancements, several challenges remain that hinder its full potential. One major gap is the lack of a unified scheduling solution that effectively handles dynamic workloads, heterogeneous infrastructure, and real-time responsiveness. Most existing studies focus on optimizing a single aspect, such as response times or resource utilization, rather than developing a multi-objective approach that balances performance, efficiency, and scalability. Additionally, while Kubernetes is widely used for managing microservices, there is still limited research on integrating AI and machine learning to enhance predictive scheduling, which could significantly improve decision making and adaptability under fluctuating network conditions. Another key challenge lies in interoperability, particularly in the integration of Kubernetes with emerging technologies, like Open RAN and 5G slicing. Without seamless compatibility, these advancements cannot fully leverage the flexibility and efficiency that Kubernetes offers. Security and fault tolerance also remain critical concerns, as container orchestration introduces new vulnerabilities that need stronger mitigation strategies. Furthermore, many of the proposed scheduling strategies are based on simulations, with limited validation using real-world 5G deployments. This gap raises concerns about their practical effectiveness in real network environments. Addressing these interconnected challenges will pave the way for a more efficient, scalable, and secure Kubernetes-scheduling framework for 5G networks, ensuring optimal performance and resilience in future telecommunication infrastructure.

7.2. Future Directions in Multi-Objective Scheduling for 5G with Kubernetes

The 5G mobile broadband network operators face significant challenges in meeting the large-scale, dynamic, and highly distributed infrastructure demands. They must deploy and manage thousands of radio antennae and networks while also handling software applications across multiple layers—from the access layer to the aggregation layer and central data centers. Moreover, they are tasked with meeting strict latency and network performance standards for both applications and infrastructure. Additionally, operators need

the agility to dynamically relocate services to optimize network performance, minimize latency, and reduce operating costs.

Consequently, 5G architectures must adopt a service-based approach, relying on hundreds or thousands of network services in the form of VNFs or CNFs spread across geographically dispersed environments. While Kubernetes partially addresses this challenge by orchestrating and managing CNFs, it still grapples with limitations when it comes to handling 5G services at distributed locations with stringent latency and performance requirements.

A multi-objective optimization strategy is necessary to solve this problem, as it involves balancing multiple, often competing goals. A number of approaches, such as heuristic and meta-heuristic algorithms, have been proposed to solve this problem. Additionally, machine-learning methods are recommended for distributed systems and large data centers to identify optimal solutions for complex problems, analyze optimization goals, and assess the effectiveness of scheduling algorithms based on KPIs. As container technology evolves, new scheduling, orchestration, placement, and resource management solutions will be needed. Emerging technologies, like edge/fog computing and microservices, offer new opportunities for developing real-time schedulers that prioritize responsiveness while also considering factors such as energy consumption, dynamic workload changes, heterogeneous infrastructure, and QoS.

Future directions include the following:

1. **Machine-Learning Integration:** Integrating machine-learning techniques for dynamic workload prediction and optimization can significantly improve the adaptability of scheduling algorithms. Machine-learning models offer valuable insights into workload patterns, empowering the system to make proactive and intelligent scheduling decisions. By discerning patterns and trends, the system can anticipate fluctuations and efficiently scale Kubernetes clusters in preparation for expected increases or decreases in demand [26,71];
2. **Continuous Monitoring and Adaptation:** Continuous monitoring enables dynamic adjustment of scheduling algorithms in real time to optimize performance in dynamic environments. This adaptability enhances responsiveness by assessing performance and resource utilization across heterogeneous infrastructure components. Leveraging real-time data, Kubernetes can allocate resources dynamically, optimizing the overall system performance based on the specific characteristics and capabilities of each component [72,73];
3. **Quantum Computing Exploration:** The exploration of quantum computing's potential impacts on complex scheduling scenarios in 5G networks offers new possibilities. Quantum computing's parallel processing capabilities may help to address multi-objective scheduling challenges more efficiently than classical computers. Quantum algorithms could enhance QoS optimization in resource allocation, scheduling, and routing, thereby improving the overall efficiency in 5G networks [74,75];
4. **Standardization and Interoperability:** Establishing industry standards is essential for ensuring seamless integration between 5G components and Kubernetes, fostering interoperability for efficient scheduling. Standardized interfaces enable dynamic service orchestration within Kubernetes, allowing for automatic resource allocation and configurations to meet real-time responsiveness requirements. This reduces compatibility issues and promotes widespread adoption by providing a common foundation for deployment and management [76,77];
5. **Autonomic Computing Principles:** Applying autonomic computing principles to scheduling algorithms in 5G with Kubernetes enables self-configuring and self-optimizing systems. These systems enhance adaptability and reduce the need for manual intervention. By continuously monitoring workloads in a 5G environment,

autonomic systems dynamically scale resources based on real-time conditions, optimizing energy consumption while meeting service-level objectives [74,78].

Emerging technologies, such as edge computing and 5G networks, present additional avenues for future research in Kubernetes resource management. There is an opportunity to leverage these technologies to usher in more efficient and scalable resource management practices within the Kubernetes ecosystem. As edge computing and 5G networks continue to evolve, their integration holds potential for optimizing resource allocation strategies, thereby addressing the unique challenges posed by dynamic computing environments [13].

Shifting focus to the virtualization network layer, findings from evaluated studies underscore the crucial roles played by network topology in fog/edge-computing environments, particularly for IoT applications. The optimization objective is to minimize latency and maximize throughput for client requests, showcasing the importance of network design in enhancing performance. Additionally, some research papers have delved into the scheduling of flows in NFV environments, specifically designed to accommodate the diverse use cases envisioned by 5G technology. Future research in this domain can further explore and refine network layer virtualization strategies, aligning them with the evolving requirements of IoT and 5G technologies. This multifaceted approach, encompassing algorithmic advancements, technological integration, and network layer optimizations, outlines the expansive and promising future directions for research in Kubernetes resource management [13].

Future directions could also involve the integration of Kubernetes with Open-RAN for end-to-end network slicing and assessing the scalability of the implemented 5G core network, emphasizing ongoing efforts to enhance practical applications and optimize deployment, in line with evolving standards and requirements [28].

8. Conclusions

The integration of Kubernetes with 5G networks presents a transformative approach to container orchestration, enhancing scalability, resource efficiency, and automation in next-generation network environments. This study has provided a comprehensive review of multi-objective scheduling algorithms, highlighting their impacts on optimizing quality-of-service (QoS) parameters, such as latency, resource utilization, and energy efficiency. Kubernetes's ability to dynamically manage containerized workloads aligns well with the evolving demands of 5G, enabling flexible and resilient service delivery.

At the same time, the study identified several key gaps that hinder the full realization of Kubernetes's potential in 5G deployments. Most existing solutions remain focused on single objectives rather than offering unified frameworks that address dynamic workloads, heterogeneous infrastructures, and real-time responsiveness simultaneously. Limited adoption of predictive intelligence, challenges in interoperability with emerging technologies, and the lack of validation in real-world 5G environments further highlight the need for continued research.

By integrating recent advancements and offering a structured overview, this study contributes not only a review of the current landscape but also a clearer picture of where future work must focus. Specifically, progress is needed in developing adaptive and resilient scheduling algorithms, integrating AI-driven orchestration, and strengthening security and interoperability. In doing so, this work provides a practical roadmap for researchers and practitioners aiming to advance Kubernetes scheduling and ensure efficient, scalable, and secure orchestration for future telecommunication networks.

Author Contributions: Conceptualization, M.F. and H.S.L.; Methodology, M.F. and H.S.L.; Investigation, C.P.L., C.C.Z. and S.F.C.; Writing-Original Draft Preparation, M.F. and H.S.L.; Review & Editing, C.P.L., C.C.Z. and S.F.C.; Supervision, H.S.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research is supported by the projects “Programming Platform for Intelligent Collaborative Deployments over Heterogeneous Edge-IoT Environments (P2CODE)” funded by the European Union’s Horizon 2020 Research and Innovation Programme (Grant Number: 101093069), and “Integrated Software Toolbox for Secure IoT-to-Cloud Computing (INTACT)”, funded by the European Commission Horizon Europe Programme (Grant Number: 101168438).

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

Acknowledgments: The authors gratefully acknowledge the support of the European Union’s Horizon 2020 Research and Innovation Programme through the project Programming Platform for Intelligent Collaborative Deployments over Heterogeneous Edge-IoT Environments (P2CODE) (Grant No. 101093069), as well as the European Commission Horizon Europe Programme through the project Integrated Software Toolbox for Secure IoT-to-Cloud Computing (INTACT) (Grant No. 101168438).

Conflicts of Interest: Authors Charilaos C. Zarakovitis and Su Fong Chien are employed by ICT Department, axon logic IKE. The remaining author declares that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

References

1. Wijethilaka, S.; Liyanage, M. Survey on Network Slicing for Internet of Things Realization in 5G Networks. *IEEE Commun. Surv. Tutor.* **2021**, *23*, 957–994. [\[CrossRef\]](#)
2. Kaur, K.; Mangat, V.; Kumar, K. A review on Virtualized Infrastructure Managers with management and orchestration features in NFV architecture. *Comput. Netw.* **2022**, *217*, 109281. [\[CrossRef\]](#)
3. Senjab, K.; Abbas, S.; Ahmed, N.; Khan, A. ur R. A survey of Kubernetes scheduling algorithms. *J. Cloud Comput.* **2023**, *12*, 87. [\[CrossRef\]](#)
4. Casalicchio, E. Container Orchestration: A Survey. In *Systems Modeling: Methodologies and Tools*; EAI/Springer Innovations in Communication and Computing; Springer: Cham, Switzerland, 2019; Volume 91, pp. 221–235. [\[CrossRef\]](#)
5. Pahl, C.; Brogi, A.; Soldani, J.; Jamshidi, P. Cloud container technologies: A state-of-the-art review. *IEEE Trans. Cloud Comput.* **2019**, *7*, 677–692. [\[CrossRef\]](#)
6. Rodriguez, M.A.; Buyya, R. Container-based cluster orchestration systems: A taxonomy and future directions. *Softw. Pract. Exp.* **2019**, *49*, 698–719. [\[CrossRef\]](#)
7. Truyen, E.; Van Landuyt, D.; Preuveneers, D.; Lagaisse, B.; Joosen, W. A comprehensive feature comparison study of open-source container orchestration frameworks. *Appl. Sci.* **2019**, *9*, 931. [\[CrossRef\]](#)
8. Arunarani, A.R.; Manjula, D.; Sugumaran, V. Task scheduling techniques in cloud computing: A literature survey. *Future Gener. Comput. Syst.* **2019**, *91*, 407–415. [\[CrossRef\]](#)
9. Vijindra; Shenai, S. Survey on scheduling issues in cloud computing. *Procedia Eng.* **2012**, *38*, 2881–2888. [\[CrossRef\]](#)
10. Hosseinioun, P.; Kheirabadi, M.; Kamel Tabbakh, S.R.; Ghaemi, R. aTask scheduling approaches in fog computing: A survey. *Trans. Emerg. Telecommun. Technol.* **2022**, *33*, e3792. [\[CrossRef\]](#)
11. Wang, K.; Zhou, Q.; Guo, S.; Luo, J. Cluster frameworks for efficient scheduling and resource allocation in data center networks: A survey. *IEEE Commun. Surv. Tutor.* **2018**, *20*, 3560–3580. [\[CrossRef\]](#)
12. Rejiba, Z.; Chamanara, J. Custom Scheduling in Kubernetes: A Survey on Common Problems and Solution Approaches. *ACM Comput. Surv.* **2022**, *55*, 151. [\[CrossRef\]](#)
13. Carrión, C. Kubernetes Scheduling: Taxonomy, Ongoing Issues and Challenges. *ACM Comput. Surv.* **2022**, *55*, 138. [\[CrossRef\]](#)
14. Tang, S.; Yu, Y.; Wang, H.; Wang, G.; Chen, W.; Xu, Z.; Guo, S.; Gao, W. A Survey on Scheduling Techniques in Computing and Network Convergence. *IEEE Commun. Surv. Tutor.* **2024**, *26*, 160–195. [\[CrossRef\]](#)
15. Ahmad, I.; AlFailakawi, M.G.; AlMutawa, A.; Alsalman, L. Container scheduling techniques: A Survey and assessment. *J. King Saud. Univ. Comput. Inf. Sci.* **2022**, *34*, 3934–3947. [\[CrossRef\]](#)
16. Sánchez, J.A.H.; Casilimas, K.; Rendon, O.M.C. Deep Reinforcement Learning for Resource Management on Network Slicing: A Survey. *Sensors* **2022**, *22*, 3031. [\[CrossRef\]](#)

17. Attaoui, W.; Sabir, E.; Elbiaze, H.; Guizani, M. VNF and CNF Placement in 5G: Recent Advances and Future Trends. *IEEE Trans. Netw. Serv. Manag.* **2023**, *20*, 4698–4733. [\[CrossRef\]](#)
18. Laghrissi, A.; Taleb, T. A Survey on the Placement of Virtual Resources and Virtual Network Functions. *IEEE Commun. Surv. Tutor.* **2019**, *21*, 1409–1434. [\[CrossRef\]](#)
19. Ali, A.Q.; Sultan, A.B.M.; Ghani, A.A.A.; Zulzalil, H. A Systematic Mapping Study on the Customization Solutions of Software as a Service Applications. *IEEE Access* **2019**, *7*, 88196–88217. [\[CrossRef\]](#)
20. Botez, R.; Pasca, A.G.; Dobrota, V. Kubernetes-Based Network Functions Orchestration for 5G Core Networks with Open Source MANO. In Proceedings of the 2022 International Symposium on Electronics and Telecommunications (ISETC), Timisoara, Romania, 10–11 November 2022; pp. 1–4. [\[CrossRef\]](#)
21. Delnat, W.; Truyen, E.; Rafique, A.; Van Landuyt, D.; Joosen, W. K8-scalar: A workbench to compare autoscalers for container-orchestrated database clusters. In Proceedings of the 40th International Conference on Software Engineering (ICSE), Gothenburg, Sweden, 27 May–3 June 2018; pp. 33–39. [\[CrossRef\]](#)
22. Cérin, C.; Menouer, T.; Saad, W.; Abdallah, W. Ben A New Docker Swarm Scheduling Strategy. In Proceedings of the 2017 IEEE 7th International Symposium on Cloud and Service Computing (SC2), Kanazawa, Japan, 22–25 November 2017; Volume 2018, pp. 112–117. [\[CrossRef\]](#)
23. Bernstein, D. Containers and cloud: From LXC to docker to kubernetes. *IEEE Cloud Comput.* **2014**, *1*, 81–84. [\[CrossRef\]](#)
24. Ogbuachi, M.C.; Gore, C.; Reale, A.; Suskovich, P.; Kovacs, B. Context-aware K8S scheduler for real time distributed 5G edge computing applications. In Proceedings of the 2019 International Conference on Software, Telecommunications and Computer Networks (SoftCOM), Split, Croatia, 19–21 September 2019; pp. 1–6. [\[CrossRef\]](#)
25. Mondal, S.K.; Pan, R.; Kabir, H.M.D.; Tian, T.; Dai, H.N. Kubernetes in IT administration and serverless computing: An empirical study and research challenges. *J. Supercomput.* **2022**, *78*, 2937–2987. [\[CrossRef\]](#)
26. Boudi, A.; Bagaa, M.; Poyhonen, P.; Taleb, T.; Flinck, H. AI-Based Resource Management in beyond 5G Cloud Native Environment. *IEEE Netw.* **2021**, *35*, 128–135. [\[CrossRef\]](#)
27. Schneider, S. Conventional and Machine Learning Approaches for Network and Service Coordination. In Proceedings of the 2023 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), Dresden, Germany, 7–9 November 2023; pp. 201–206. [\[CrossRef\]](#)
28. Tsolkas, D.; Charismiadi, A.S.; Xenakis, D.; Merakos, L. Service and network function placement in the edge-cloud continuum. In Proceedings of the 2022 IEEE Conference on Standards for Communications and Networking (CSCN), Thessaloniki, Greece, 28–30 November 2022; pp. 188–193. [\[CrossRef\]](#)
29. Chen, Y. Optimal Configuration of Distributed Device Container Resources considering Virtual Aggregation Rules and Edge Node Constraints. In Proceedings of the 2023 8th Asia Conference on Power and Electrical Engineering (ACPEE), Tianjin, China, 14–16 April 2023; pp. 1844–1851. [\[CrossRef\]](#)
30. Charismiadi, A.S.; Tsolkas, D.; Passas, N.; Xenakis, D.; Merakos, L. Metaheuristics as Enablers for VNF Scheduling in the Network Slice Set Up Process. *J. Commun. Netw.* **2022**, *24*, 742–753. [\[CrossRef\]](#)
31. Lee, P.-H.; Lin, F.J. Tackling IoT Scalability with 5G NFV-Enabled Network Slicing. *Adv. Internet Things* **2021**, *11*, 123–139. [\[CrossRef\]](#)
32. Alemany, P.; Roman, A.; Vilalta, R.; Pol, A.; Bonnet, J.; Kapassa, E.; Touloupou, M.; Kyriazis, D.; Karkazis, P.; Trakadas, P.; et al. A KPI-Enabled NFV MANO Architecture for Network Slicing with QoS. *IEEE Commun. Mag.* **2021**, *59*, 44–50. [\[CrossRef\]](#)
33. Qian, D.; Guo, S.; Sun, L.; Hao, Q.; Song, Y.; Wang, M. An Integrity Measurement Scheme for Containerized Virtual Network Function. In Proceedings of the 2021 5th International Conference on Electrical, Mechanical and Computer Engineering (ICEMCE 2021), Xi'an, China, 29–31 October 2021; Volume 2137. [\[CrossRef\]](#)
34. Nguyen, D.T.; Dao, N.L.; Tran, V.T.; Lang, K.T. Enhancing CNF performance for 5G core network using SR-IOV in Kubernetes. In Proceedings of the 2022 24th International Conference on Advanced Communication Technology (ICACT), PyeongChang, Republic of Korea, 13–16 February 2022; pp. 501–506. [\[CrossRef\]](#)
35. Vu, D.D.; Tran, M.N.; Kim, Y. Predictive Hybrid Autoscaling for Containerized Applications. *IEEE Access* **2022**, *10*, 109768–109778. [\[CrossRef\]](#)
36. Pino, A.; Khodashenas, P.; Hesselbach, X.; Coronado, E.; Siddiqui, S. Validation and Benchmarking of CNFs in OSM for pure Cloud Native applications in 5G and beyond. In Proceedings of the 2021 International Conference on Computer Communications and Networks (ICCCN), Athens, Greece, 19–22 July 2021; Volume 2021, pp. 1–9. [\[CrossRef\]](#)
37. Arora, S.; Ksentini, A. Dynamic Resource Allocation and Placement of Cloud Native Network Services. In Proceedings of the ICC 2021-IEEE International Conference on Communications, Montreal, QC, Canada, 14–23 June 2021; pp. 1–6. [\[CrossRef\]](#)
38. Dion, J.; Lallet, J.; Beaulieu, L.; Savelli, P.; Bertin, P. Cloud Native Hardware Accelerated 5G virtualized Radio Access Network. In Proceedings of the 2021 IEEE 32nd Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC), Helsinki, Finland, 13–16 September 2021; Volume 2021, pp. 1061–1066. [\[CrossRef\]](#)

39. Amogh, P.C.; Veeramachaneni, G.; Rangiseti, A.K.; Tamma, B.R.; Antony, F.A. A cloud native solution for dynamic auto scaling of MME in LTE. In Proceedings of the 2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC), Montreal, QC, Canada, 8–13 October 2017; Volume 2017, pp. 1–7. [\[CrossRef\]](#)
40. Zeb, S.; Rathore, M.A.; Mahmood, A.; Hassan, S.A.; Kim, J.; Gidlund, M. Edge Intelligence in Softwarized 6G: Deep Learning-enabled Network Traffic Predictions. In Proceedings of the 2021 IEEE Globecom Workshops (GC Wkshps), Madrid, Spain, 7–11 December 2021.
41. Da Silva, D.C.; Batista, J.O.R.; de Sousa, M.A.F.; Mostaço, G.M.; Monteiro, C.d.C.; Bressan, G.; Cugnasca, C.E.; Silveira, R.M. A Novel Approach to Multi-Provider Network Slice Selector for 5G and Future Communication Systems. *Sensors* **2022**, *22*, 6066. [\[CrossRef\]](#)
42. Tsourdinis, T.; Chatzistefanidis, I.; Makris, N.; Korakis, T.; Nikaein, N.; Fdida, S. Service-aware real-time slicing for virtualized beyond 5G networks. *Comput. Netw.* **2024**, *247*, 110445. [\[CrossRef\]](#)
43. Mudvari, A.; Makris, N.; Tassiulas, L. ML-driven scaling of 5G Cloud-Native RANs. In Proceedings of the GLOBECOM 2021-2021 IEEE Global Communications Conference, Madrid, Spain, 7–11 December 2021; pp. 1–6. [\[CrossRef\]](#)
44. Watanabe, H.; Yasumori, R.; Kondo, T.; Kumakura, K.; Maesako, K.; Zhang, L.; Inagaki, Y.; Teraoka, F. ContMEC: An Architecture of Multi-access Edge Computing for Offloading Container-Based Mobile Applications. In Proceedings of the ICC 2022-IEEE International Conference on Communications, Seoul, Republic of Korea, 16–20 May 2022; Volume 2022, pp. 3647–3653. [\[CrossRef\]](#)
45. Wauters, T.; Volckaert, B.; Turck, F. De Towards delay-aware container-based Service Function Chaining in Fog Computing. In Proceedings of the 2020 IEEE/IFIP Network Operations and Management Symposium (NOMS 2020), Budapest, Hungary, 20–24 April 2020. [\[CrossRef\]](#)
46. Dab, B.; Fajjari, I.; Rohon, M.; Auboin, C.; Diquelou, A. An Efficient Traffic Steering for Cloud-Native Service Function Chaining. In Proceedings of the 2020 23rd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN), Paris, France, 24–27 February 2020; pp. 71–78. [\[CrossRef\]](#)
47. De Simone, L.; Di Mauro, M.; Natella, R.; Postiglione, F. Performance and Availability Challenges in Designing Resilient 5G Architectures. *IEEE Trans. Netw. Serv. Manag.* **2024**, *21*, 5291–5303. [\[CrossRef\]](#)
48. Nikaein, N.; Schiller, E.; Favraud, R.; Katsalis, K.; Stavropoulos, D.; Alyafawi, I.; Zhao, Z.; Braun, T.; Korakis, T. Network store: Exploring slicing in future 5G networks. In Proceedings of the MobiCom'15: The 21th Annual International Conference on Mobile Computing and Networking, Paris, France, 7–11 September 2015; pp. 8–13. [\[CrossRef\]](#)
49. Katsalis, K.; Nikaein, N.; Schiller, E.; Favraud, R.; Braun, T.I. 5G Architectural Design Patterns. In Proceedings of the 2016 ICC-2016 IEEE International Conference on Communications Workshops (ICC), Kuala Lumpur, Malaysia, 23–27 May 2016; pp. 32–37. [\[CrossRef\]](#)
50. Luong, D.H.; Outtagarts, A.; Ghamri-Doudane, Y. Multi-level resource scheduling for network slicing toward 5G. In Proceedings of the 2019 10th International Conference on Networks of the Future (NoF), Rome, Italy, 1–3 October 2019; pp. 25–31. [\[CrossRef\]](#)
51. Aba, M.A.; Kassis, M.; Elkael, M.; Araldo, A.; Al Khansa, A.; Castel-Taleb, H.; Jouaber, B. Efficient Network Slicing Orchestrator for 5G Networks using a Genetic Algorithm-based Scheduler with Kubernetes: Experimental Insights. In Proceedings of the 2024 IEEE 10th International Conference on Network Softwarization (NetSoft), Saint Louis, MO, USA, 24–28 June 2024; pp. 82–90. [\[CrossRef\]](#)
52. Mukuhi, D.K.; Outtagarts, A. LLRS: A Low Latency Resource Scheduling in Cloud Continuum Orchestration. In Proceedings of the 2024 27th Conference on Innovation in Clouds, Internet and Networks (ICIN), Paris, France, 11–14 March 2024; pp. 81–87. [\[CrossRef\]](#)
53. Mekki, M.; Arora, S.; Ksentini, A. A Scalable Monitoring Framework for Network Slicing in 5G and beyond Mobile Networks. *IEEE Trans. Netw. Serv. Manag.* **2022**, *19*, 413–423. [\[CrossRef\]](#)
54. Aleyadeh, S.; Moubayed, A.; Heidari, P.; Shami, A. Optimal Container Migration/Re-Instantiation in Hybrid Computing Environments. *IEEE Open J. Commun. Soc.* **2022**, *3*, 15–30. [\[CrossRef\]](#)
55. Wojciechowski, L.; Opasiak, K.; Latusek, J.; Wereski, M.; Morales, V.; Kim, T.; Hong, M. NetMARKS: Network metrics-AwaRe kubernetes scheduler powered by service mesh. In Proceedings of the IEEE INFOCOM 2021-IEEE Conference on Computer Communications, Vancouver, BC, Canada, 10–13 May 2021; Volume 2021. [\[CrossRef\]](#)
56. Pusztai, T.; Nastic, S.; Morichetta, A.; Pujol, V.C.; Raith, P.; Dustdar, S.; Vij, D.; Xiong, Y.; Zhang, Z. Polaris Scheduler: SLO- and Topology-aware Microservices Scheduling at the Edge. In Proceedings of the 2022 IEEE/ACM 15th International Conference on Utility and Cloud Computing (UCC), Portland, OR, USA, 6–9 December 2022; pp. 61–70. [\[CrossRef\]](#)
57. Wadatkar, P.V.; Garroppo, R.G.; Nencioni, G.; Volpi, M. Joint multi-objective MEH selection and traffic path computation in 5G-MEC systems. *Comput. Netw.* **2024**, *240*, 110168. [\[CrossRef\]](#)
58. Rahali, M.; Phan, C.; Rubino, G.; Rahali, M.; Phan, C.; Rubino, G.; Kubernetes, K.R.S.; Scheduler, R.; Rahali, M.; Phan, C.; et al. KRS: Kubernetes Resource Scheduler for resilient NFV networks. In Proceedings of the GLOBECOM 2021—2021 IEEE Global Communications Conference, Madrid, Spain, 7–11 December 2021.

59. Dimolitsas, I.; Dechouniotis, D.; Papavassiliou, S. Time-efficient distributed virtual network embedding for round-trip delay minimization. *J. Netw. Comput. Appl.* **2023**, *217*, 103691. [\[CrossRef\]](#)
60. Caminero, A.C. Quality of Service Provision in Fog Computing: Network-Aware Scheduling of Containers. *Sensors* **2021**, *21*, 3978. [\[CrossRef\]](#)
61. Botez, R.; Costa-Requena, J.; Ivanciu, I.A.; Strautiu, V.; Dobrota, V. Sdn-based network slicing mechanism for a scalable 4g/5g core network: A kubernetes approach. *Sensors* **2021**, *21*, 3773. [\[CrossRef\]](#)
62. Ojaghi, B.; Adelantado, F.; Verikoukis, C. SO-RAN: Dynamic RAN Slicing via Joint Functional Splitting and MEC Placement. *IEEE Trans. Veh. Technol.* **2023**, *72*, 1925–1939. [\[CrossRef\]](#)
63. Srirama, S.N. A decade of research in fog computing: Relevance, challenges, and future directions. *Softw. Pract. Exp.* **2023**, *54*, 3–23. [\[CrossRef\]](#)
64. Biran, Y.; Pasricha, S.; Collins, G.; Dubow, J. Enabling green content distribution network by cloud orchestration. In Proceedings of the 2016 3rd Smart Cloud Networks & Systems (SCNS), Dubai, United Arab Emirates, 19–21 December 2016. [\[CrossRef\]](#)
65. Xu, Z.; Gong, Y.; Zhou, Y.; Bao, Q.; Qian, W. Enhancing Kubernetes Automated Scheduling with Deep Learning and Reinforcement Techniques for Large-Scale Cloud Computing Optimization. In Proceedings of the Ninth International Symposium on Advances in Electrical, Electronics, and Computer Engineering (ISAEECE 2024), Changchun, China, 15–17 March 2024.
66. Oluebube, B.; Egbuna, P.; Member, R.T.; Org, K. Machine Learning Applications in Kubernetes for Autonomous Container Management. *J. Artif. Intell. Res.* **2024**, *4*, 196–219.
67. Mao, Y.; Fu, Y.; Zheng, W.; Cheng, L.; Liu, Q.; Tao, D. Speculative Container Scheduling for Deep Learning Applications in a Kubernetes Cluster. *IEEE Syst. J.* **2021**, *16*, 3770–3781. [\[CrossRef\]](#)
68. F5, “5G Makes a Cloud-Native Application Architecture Vital,” F5 Resources. [Online]. Available online: https://www.f5.com/pt_br/resources/articles/5g-makes-a-cloud-native-application-architecture-vital (accessed on 12 September 2025).
69. SID Global Solutions. Exploring the top 10 Kubernetes Alternatives: Which One Is Right for You? Medium, Mar. 7, 2023. [Online]. Available online: <https://sidglobalsolutions.medium.com/exploring-the-top-10-kubernetes-alternatives-which-one-is-right-for-you-7249faf724b> (accessed on 12 September 2025).
70. Kaur, K.; Garg, S.; Kaddoum, G.; Ahmed, S.H.; Atiquzzaman, M. KEIDS: Kubernetes-Based Energy and Interference Driven Scheduler for Industrial IoT in Edge-Cloud Ecosystem. *IEEE Internet Things J.* **2020**, *7*, 4228–4237. [\[CrossRef\]](#)
71. Chen, Z.; Hu, J.; Min, G.; Zomaya, A.Y.; El-Ghazawi, T. Towards Accurate Prediction for High-Dimensional and Highly-Variable Cloud Workloads with Deep Learning. *IEEE Trans. Parallel Distrib. Syst.* **2020**, *31*, 923–934. [\[CrossRef\]](#)
72. Ogbuachi, M.C.; Reale, A.; Suskovics, P.; Kovács, B. Context-Aware kubernetes scheduler for edge-native applications on 5g. *J. Commun. Softw. Syst.* **2020**, *16*, 85–94. [\[CrossRef\]](#)
73. Perez, R.; Benedetti, P.; Pergolesi, M.; Garcia-Reinoso, J.; Zabala, A.; Serrano, P.; Femminella, M.; Reali, G.; Steenhaut, K.; Banchs, A. Monitoring Platform Evolution Toward Serverless Computing for 5G and Beyond Systems. *IEEE Trans. Netw. Serv. Manag.* **2022**, *19*, 1489–1504. [\[CrossRef\]](#)
74. Gill, S.S.; Xu, M.; Ottaviani, C.; Patros, P.; Bahsoon, R.; Shaghaghi, A.; Golec, M.; Stankovski, V.; Wu, H.; Abraham, A.; et al. AI for next generation computing: Emerging trends and future directions. *Internet Things* **2022**, *19*, 100514. [\[CrossRef\]](#)
75. Zeb, S.; Rathore, M.A.; Hassan, S.A.; Raza, S.; Dev, K.; Fortino, G. Toward AI-Enabled NextG Networks with Edge Intelligence-Assisted Microservice Orchestration. *IEEE Wirel. Commun.* **2023**, *30*, 148–156. [\[CrossRef\]](#)
76. Apostolakis, K.C.; Margetis, G.; Stephanidis, C.; Duquerois, J.M.; Drouglazet, L.; Lallet, A.; Delmas, S.; Cordeiro, L.; Gomes, A.; Amor, M.; et al. Cloud-native 5G infrastructure and network applications (NetApps) for public protection and disaster relief: The 5G-EPICENTRE Project. In Proceedings of the 2021 Joint European Conference on Networks and Communications & 6G Summit (EuCNC/6G Summit), Porto, Portugal, 8–11 June 2021; pp. 235–240. [\[CrossRef\]](#)
77. Varga, P.; Peto, J.; Franko, A.; Balla, D.; Haja, D.; Janky, F.; Soos, G.; Ficzer, D.; Maliosz, M.; Toka, L. 5G Support for Industrial Iot Applications—Challenges, Solutions, and Research Gaps. *Sensors* **2020**, *20*, 828. [\[CrossRef\]](#) [\[PubMed\]](#)
78. Huang, S.Y.; Chen, C.Y.; Chen, J.Y.; Chao, H.C. A Survey on Resource Management for Cloud Native Mobile Computing: Opportunities and Challenges. *Symmetry* **2023**, *15*, 538. [\[CrossRef\]](#)

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.