Autonomy and Intelligence in the Computing Continuum: Challenges, Enablers, and Future Directions for Orchestration

Henna Kokkonen*††, Lauri Lovén*‡‡††, Naser Hossein Motlagh‖, Juha Partala†,
Alfonso González-Gil**, Ester Sola**, Iñigo Angulo**, Madhusanka Liyanage§¶,
Teemu Leppänen*, Tri Nguyen*, Panos Kostakos*,
Mehdi Bennis§, Sasu Tarkoma‖*, Schahram Dustdar‡‡, Susanna Pirttikangas*, Jukka Riekki*
*Center for Ubiquitous Computing, University of Oulu, Finland
†Center for Machine Vision and Signal Analysis, University of Oulu, Finland
§Center for Wireless Communications, University of Oulu, Finland
¶School of Computer Science, University College Dublin, Ireland
‖Department of Computer Science, University of Helsinki, Finland
‡‡Distributed Systems Group, TU Wien, Austria
**Zylk, Spain
Email: *†§{firstname.lastname}@oulu.fi, ¶madhusanka@ucd.ie

These authors contributed equally: ††

***Abstract*—Future AI applications require performance, reliability, and privacy that the existing, cloud-dependant system architectures cannot provide. In this article, we study orchestration in the device-edge-cloud continuum, and focus on AI for edge, that is, the AI methods used in resource orchestration. We claim that to support the constantly growing requirements of intelligent applications in the device-edge-cloud computing continuum, resource orchestration needs to embrace edge AI and emphasize local autonomy and intelligence. To justify the claim, we provide a general definition for continuum orchestration, and look at how current and emerging orchestration paradigms are suitable for the computing continuum. We describe certain major emerging research themes that may affect future orchestration, and provide an early vision of an orchestration paradigm that embraces those research themes. Finally, we survey current key edge AI methods and look at how they may contribute into fulfilling the vision of future continuum orchestration.**

***Index Terms*—edge AI, edge intelligence, computing continuum, orchestration**

I. Introduction

Bringing intelligence *to* the edge requires intelligence *on* the edge. We argue that in the future device-edge-cloud computing continuum, the continuum cannot be seen as a static environment, and orchestration there cannot rely on the traditional best-effort, reactive, threshold based methods of the current centralized orchestration paradigms developed for the cloud. Instead, we must embrace autonomy and distributed intelligence.

In more detail, future AI applications require performance, reliability and privacy that the existing, cloud-dependant system architectures cannot provide. In these systems, sensor data is transported to the cloud for AI analytics, overburdening the networks, consuming a lot of energy, increasing the latency of task processing, and raising privacy concerns. These issues have caused a paradigm shift: instead of having all AI processing in the cloud, the intelligence is brought to the edge, closer to the data generating devices, users, and controlled devices.

However, while edge-based processing increases data privacy and shortens latencies, deploying AI applications on the edge introduces significant challenges. Edge is a distributed platform of heterogeneous nodes, characterized by fluctuating and intermittent communication, opportunistic, heterogeneous and distributed computational resources, and siloed, distributed, and non-IID data (Fig. 1). Transferring centralized AI with high resource requirements to the distributed, resource-constrained environment, while providing guarantees for performance, reliability, and security, requires new architectures and algorithms for training, inference and decision making over wireless and fixed links.

On the other hand, edge/fog computing is based on the deployment of a wide variety of applications on heterogeneous hardware components (e.g., CPU, GPU, or FPGA), each with its own software tools and dependencies. To mitigate the heterogeneity of the platforms, virtualization techniques are widely used to extend the physical limitations of the

This research has been supported by Academy of Finland 6Genesis Flagship program (grant 346208); the ECSEL JU FRACTAL (grant 877056), receiving support from the EU Horizon 2020 programme and Spain, Italy, Austria, Germany, France, Finland, Switzerland; and the Infotech Oulu research institute.

infrastructure, as they provide isolated software environments and full abstraction of the software or hardware running behind the environment. This includes IT approaches, such as virtual machines (VM) and containers [1], which share a pool of resources that can be scaled dynamically, and network approaches, such as Network Function Virtualization (NFV), Software Defined Networks (SDN), and Network Slicing [2, 3], which permit decoupling the data and control planes from the constraints of the physical network. This allows multi-tenant environments to adapt to different edge/fog computing scenarios.

These enabling technologies lead towards a wide range of new and innovative services [2]. Many of them, such as computation offloading, multi-site collaboration and context awareness, are presented as disruptive approaches that will change current computing paradigms, catering, for example, for the requirements of distributed and cooperative AI applications [4, 5]. As the computational workloads of these services become more demanding, automating their deployment and management over the available computational resources is a requirement for efficient use of resources and optimal performance.

Furthermore, to reach the device-edge-cloud computing continuum envisioned as the future direction in 5G and beyond architectures [2, 6], telecommunication and IT services must convergence and be jointly optimized. Existing physical infrastructure, comprising IT elements (e.g., machine clusters in the cloud or edge/fog servers) and network components (access points and transport networks), has to cater for the needs of an opportunistic computing continuum, provisioning services flexibly and scalably [7].

Increasing the intelligence of the edge platform itself, promoting next generation services, and bringing about a coherent computing continuum are therefore necessities for supporting the AI workflows on the edge. To this end, 5G technology has taken initiative in improving the reliability and scalability of wireless networks by introducing Ultra-reliable low-latency communication (URLLC), Enhanced Mobile Broadband (eMBB), and Massive Machine Type Communication (mMTC). Furthermore, reaching the 6G vision of ubiquitous wireless intelligence and use cases in, e.g., extended reality (XR) and telepresence [8] requires autonomous, context-aware, self-learning and self-optimising computational, communicational and data storage services that will fulfill the growing requirements of security, efficiency, and reliability.

Subsequently, the device-edge-cloud computing continuum, as a crucial element of the next generation mobile and fixed networks, requires AI solutions for its orchestration, that is, for managing, allocating and distributing its resources in a dynamic, efficient, and context-aware manner.

Designing AI methods for the edge environment and deploying AI applications there, and developing AI solutions that intelligently configure, control and manage the edge environment to optimize its resource usage, have respectively been coined as *AI on edge* and *AI for edge* [9, 10]. Together, these aspects form *edge AI* or *edge intelligence* (Fig. 1).

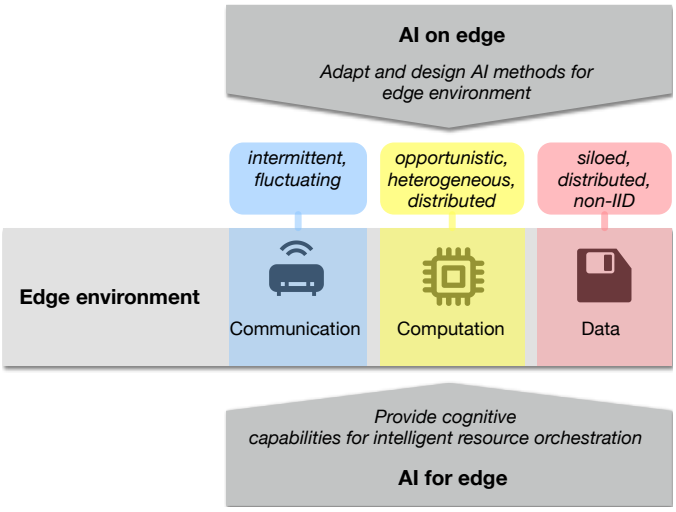

Fig. 1: AI on edge vs. AI for edge.

This confluence of edge computing and AI with the next generation of communication and computation in the device-edge-cloud continuum has been envisioned as a fundamental key factor in the creation of the *Intelligent Internet of Intelligent Things* [11]. The culmination of this confluence is distributed, edge-native AI, which will enable novel, smart applications in domains such as urban computing, smart environments, personalized services and context-aware mobile technologies [10], supporting sustainability and resilience through efficient resource usage and dynamic and context-aware operation.

A. Contributions

Taking a very different point of view compared to related work (Section II), we focus on holistically bringing together distributed AI and orchestration in the device-edge-cloud computing continuum. We adopt the Multi-Agent System (MAS) paradigm to regard the edge platform as a dynamic network of AI nodes. Each node is a highly adaptive, self-interested, autonomous and context-aware agent that is able to monitor the state of the surrounding dynamic and complex edge environment, forecast internal and external events, and react proactively to changes based on the forecasts. We build the survey around a vision for AI for edge, that is, by developing AI solutions for the orchestration of the network, the edge environment will eventually evolve into a coherent computing continuum that is able to function in an autonomous, decentralized and decoupled manner, while optimizing and balancing multiple objectives with regard to, e.g., efficiency, reliability and security. The network will be able to orchestrate its limited computational, network, energy and memory resources in a globally optimized manner while being aware of and ready to adapt to the dynamic environment.

The main aim of this survey is to bring strong emphasis to and justify a paradigm shift in edge orchestration. We see it imperative that orchestration paradigms move from centralized approaches focusing on specific domains such as container

or network orchestration towards an autonomous, intelligent and decentralized approaches that take a holistic view on the resources residing in the device-edge-cloud computing continuum.

To achieve this, a more comprehensive view on the edge orchestration is required along with the development of distributed AI methods for the orchestration. Hence, in this survey, we will provide a unified view on resources and orchestration in the computing continuum by combining several different taxonomies found in the literature. We provide an overview of the current state of the art in different orchestration paradigms, identifying what they are lacking in terms of moving towards autonomous and decentralized orchestration that caters to the inherent characteristics of the computing continuum environment (e.g., geographical distribution, heterogeneity, non-IID data and intermittent connectivity).

Furthermore, we look at promising future avenues for research, identify their challenges, and outline the general framework of our vision.

Finally, we identify edge AI methods that we regard as the most significant directions towards achieving autonomous and decentralized orchestration. We survey the state of the art in these methods in order to pinpoint the main challenges that need to be overcome so that they can be deployed in the computing continuum environment.

The overall contribution of this survey is to justify the claim that to support the constantly growing requirements of intelligent applications in the device-edge-cloud computing continuum, resource orchestration needs to embrace edge AI and emphasize local autonomy and intelligence. In more detail, the contribution can be broken down as follows:

- We provide an encompassing view on computing continuum resources and orchestration, proposing a broader definition for orchestration, which relies on the finding that considering *Everything as a Resource* (EaaR) simplifies the overall view.
- We bring together several different orchestration taxonomies from the literature, proposing a new, more holistic taxonomy.
- We introduce current and emerging orchestration paradigms, pinpointing their main deficiencies with regard to achieving an autonomous and intelligently optimized device-edge-cloud computing continuum orchestration.
- We provide an early vision of autonomous, weakly coupled and secure orchestration in the computing continuum.
- We identify the main opportunities and promising research directions brought forth by the vision, and identify the main challenges that currently obstruct its realization.
- We provide an extensive overview of the current state of the art in AI methods that we regard as the key factors in realizing an autonomous computing continuum.

The rest of the article is structured as follows. Section II gives an overview of related work in edge AI and orchestration. Section III briefly defines the main concepts required for studying orchestration, that is, agent modelling, distributed AI, autonomy, computing continuum and orchestration, and sets the scope using these definitions. Section IV presents current and emerging orchestration paradigms, while Section V charts out the opportunities and development directions in computing continuum orchestration, and presents an early vision of autonomous orchestration in the computing continuum. Section VI identifies requirements for AI methods in the computing continuum, and surveys the current state of the art, mainly focusing on methods in distributed learning and decision making, while also covering some other promising approaches. Finally, Section VII concludes the article.

II. Related Work

A. Edge AI

Recently, there has been a surge in edge AI related surveys. As seen in Table I, which summarizes the main contributions and limitations of such surveys, the majority of them strongly emphasise the AI on edge aspect. In other words, the focal point of interest has been in how to enable AI model learning on top of the edge infrastructures. For example, Park et al. [14] provide a very comprehensive survey of communication-efficient model building on edge, focusing on introducing technical and theoretical enablers for accurate and low latency model training and inference.

We focus on AI for edge. This aspect of applying AI techniques to the optimization of edge in order to make it function in a more intelligent and autonomous manner has received considerably less attention. Most notably, Deng et al. [9] categorize AI for edge into Topology (Orchestration of Edge Sites and Wireless Networking), Content (Data Provisioning, Service Provisioning, Service Placement, Service Composition and Service Caching) and Service (Computation Offloading, User Profile Migration and Mobility Management). However, they do not provide a comprehensive overview of different AI methods utilized inside this taxonomy, but focus only on what AI methods have been applied to Wireless Networking, Service Placement and Caching, and Offloading.

On the other hand, Lim et al. [13] focus on how Federated Learning (FL) has been applied to edge network problems, namely in cyberattack detection, edge caching and offloading, base station association and vehicular networks. As a part of their Multi-access Edge Computing (MEC) survey, Pham et al. [19] investigate how Machine Learning (ML) has been applied to several different edge problems: edge caching, computation offloading, resource allocation, security and privacy, big data analytics, and mobile crowdsensing. However, in both articles, the overview is limited, in the former due to the focus only on FL, and in the latter because the authors focus on providing a very concise overview of ML for edge in the context of a broader survey on MEC.

Wang et al. [16] provide a comprehensive survey of the union of Deep Learning (DL) and Edge Computing. They focus mainly on DL on Edge aspect, but also give an overview of how DL has been used for optimizing edge. The covered edge management issues are edge caching, task offloading,

TABLE I: Edge Intelligence related surveys

Reference	Topic	Main contribution	Limitations
Deng et al. 2019 [9]	Edge AI	Surveys compactly both sides of Edge AI, that is, AI for Edge and AI on Edge. Provides a taxonomy for AI for Edge, which consists of three categories: Topology, Content and Service.	State of the Art in Edge for AI focuses on Wireless Networking, Service Placement and Caching, and Offloading.
Xu et al. 2020 [12]	Edge AI	Focuses on AI on Edge aspect, categorizes Edge intelligence into Edge Training, Edge Inference, Edge Caching and Edge Offloading.	Does not survey AI methods that have been applied to solving edge problems.
Lim et al. 2020 [13]	FL	Focuses on the FL implementation challenges and solutions on edge, as well as surveys how FL has been used as a tool in optimizing mobile edge network problems.	Focuses solely on the application of FL in MEC problems.
Park et al. 2019 [14]	Edge ML	Focuses on technical and theoretical enablers of Edge ML, particularly from the viewpoint of efficient communication.	Main emphasis is on AI on Edge aspect, i.e., enablers for training an accurate model on edge with low latency, high scalability and efficiency.
Park et al. 2021 [15]	Edge ML	Surveys communication and ML principles and enablers for achieving communication-efficient distributed learning on edge.	Focuses on AI on Edge aspect.
Wang et al. 2020 [16]	Edge DL	Surveys comprehensively the relationship of DL and Edge Computing from the viewpoints of DL for Edge and DL on Edge.	Focuses solely on DL techniques.
Zhou et al. 2019 [17]	Edge AI	Surveys architectures, key performance indicators and enabling technologies for DL model training and inference on edge.	Focuses on AI on Edge aspect.
Hao et al. 2021 [18]	Edge AI	Surveys enabling single-layer and cross-layer design methodologies for Edge AI.	AI on Edge viewpoint, that is, what kind of hardware and software design methodologies exist for developing and deploying AI applications on Edge.
Pham et al. 2020 [19]	MEC	As a part of the survey, how ML has been applied to solving MEC problems is covered.	Provides a concise overview of the State of the Art in each MEC problem area, focusing on DL based works.
Xu et al. 2022 [20]	AI for IoV	Surveys AI methods that have been utilized in optimizing edge server placement, as well as computation and service offloading in Internet of Vehicles.	Focus on IoV.
Shi et al. 2020 [21]	Edge AI	Surveys communication challenges and proposed communication-efficient algorithms and systems for Edge AI.	Focuses on AI on Edge aspect (edge training and edge inference).

edge communication, security and joint edge optimization. Finally, Xu et al. [20] survey AI solutions for edge server placement and offloading in Internet of Vehicles (IoV). Their survey is very limited due to the focus on IoV.

B. Orchestration

The device-edge-cloud continuum contains many elements, from fundamental ones such as energy or time through hardware and virtual elements to service, application and workflow related elements. In literature, there is an abundance of orchestration surveys that usually focus on a specific subset of these elements, as well as on a specific part of the device-edge-cloud continuum. In other words, a majority of the surveys focus on a specific domain, such as the management of containers, networks or tasks. There does not seem to exist surveys that would try to holistically piece together different aspects on orchestration in the whole device-edge-cloud continuum. Hence, in this section, we will highlight significant surveys from different aspects. These surveys serve as the main basis for our more encompassing view to orchestration in the device-edge-cloud continuum. The main contributions and limitations of the presented surveys are summarized in Table II.

Toczé and Nadjm-Tehrani [23], Hong and Varghese [25], as well as Mampage et al. [28] all provide surveys on resource management. Toczé and Nadjm-Tehrani propose a taxonomy for resource management at the edge. Their taxonomy has four main categories: resource type, resource management objective, resource location, and resource use. They see that there are six different resource types: computation, communication, storage, data, energy and generic (e.g., resources are abstracted to a virtual value). For the objective of the resource management, they see five main goals: estimation, discovery, allocation, sharing, and optimization. For the resource location, they adopt two perspectives: where the resources are located within the architectures, and whether the resources are stationary or mobile. Finally, for the resource use they have two aspects on what is the purpose for which the resource will be used: functional properties (enabling functionalities in applications) and nonfunctional properties (realizing desirable properties such as energy efficiency).

Hong and Varghese propose a taxonomy for resource

TABLE II: Orchestration related surveys

Reference	Topic	Main contribution	Limitations
Taleb et al. 2017 [2]	MEC orchestration	Provides a survey for MEC service orchestration in edge-cloud architectures.	The focus is on service orchestration in edge-cloud continuum.
Guerzoni et al. 2017 [22]	Multi-domain orchestration	Provides a survey on architectures for orchestration in software defined infrastructures and proposes a reference architecture for end-to-end multi-domain orchestration.	Focuses on architectures for enabling multi-domain resource and service orchestration in software defined networks, lacks an edge computing aspect, and no consideration of AI based orchestration solutions.
Toczé and Nadjm-Tehrani 2018 [23]	Resource management	Proposes a taxonomy for resource management in the device-edge-cloud continuum.	The focus is on the management of physical and virtual resources.
de Sousa et al. 2019 [24]	Network orchestration	Provides a comprehensive survey of network service orchestration and proposes a taxonomy for network orchestration approaches.	Focuses on the life cycle management of network services and does not include an edge computing aspect.
Hong and Varghese 2019 [25]	Resource management	Proposes a taxonomy for resource management in fog computing, focusing on architectures, infrastructure and algorithms.	The focus is on the management of physical and virtual resources.
Zhong et al. 2021 [26]	Container orchestration	Provides a comprehensive overview of and a taxonomy for ML based container orchestration.	Focuses on container life cycle management.
Versluis and Iosup 2021 [27]	Workflow orchestration	Provides a systematic literature review on workflow orchestration and taxonomizes four key areas inside it.	The focus is on the management of workflows.
Mampage et al. 2022 [28]	Resource management	Provides a taxonomy for resource management in serverless computing environments.	The focus is on the management of physical and virtual resources.
Costa et al. 2022 [29]	Fog orchestration	Proposes a generic architecture for fog orchestration based on an extensive literature review.	The architecture consolidates approaches from the literature, that is, the paper does not state concrete enablers for realising the whole architecture.

management in fog/edge computing, focusing on three main aspects: architectures, infrastructure, and algorithms. They classify architectures based on data flow, control and tenancy. Data flow is analysed in terms of how data or workloads are transferred within a fog/edge environment. They identify three types of data flow architectures: aggregation, sharing between peers, and offloading to edge. Control of resources can be centralized, distributed or hierarchical. Tenancy relates to whether an edge node must host multiple applications and support multiple users. Infrastructure is classified into hardware, software and middleware. Hardware is either computing (for data processing) or network (for traffic processing) devices. System software manages the computation, network, and storage resources of the devices. Middleware provides complementary services to system software. The resource management algorithms are classified into discovery (identifying edge resources within the network), benchmarking (capturing the performance of resources), load-balancing (distributing workloads across resources efficiently), and placement (placing workloads on appropriate resources).

Finally, Mampage et al. propose a taxonomy for resource management in serverless environments. They see that there are three major aspects to resource management in serverless platforms: application workload modelling, resource scheduling and resource scaling. Their taxonomy has four main categories: resource management elements, which consists of the three aforementioned main aspects; deployment environment, which relates to understanding the factors influencing the design of a serverless platform; workload management, which relates to understanding the application requirements and structure, the nature of the workload, and data locality; and Quality of Service (QoS) goal.

The viewpoint of all three resource management surveys is limited to the orchestration of the lower level elements, i.e., fundamental, physical and virtual elements.

Taleb et al. [2] provide a survey on MEC orchestration, focusing in particular on MEC service orchestration while also providing insights into the joint orchestration of MEC services and virtualized network functions. They have three main aspects on MEC orchestration: (1) allocating resources for services, placing services on platforms, selecting the platform on which allocate a service request, and considering the reliability of MEC service deployments; (2) supporting service mobility within a set of platforms; (3) the joint optimization of virtualized network functions and MEC services on platforms in order to achieve cross-layer optimization and efficient resource usage. Their view is limited on the orchestration of services and virtualized network functions on edge-cloud platforms.

Guerzoni et al. [22] survey different architectures for end-to-end orchestration of resources and services in the future 5G environments. They also form a reference architecture that includes all the capabilities that end-to-end orchestration is expected to fulfill. They see multi-domain orchestration as the key enabler behind end-to-end orchestration that spans from the infrastructure layer to the application layer in software

defined infrastructures. They define orchestration as "the automated arrangement and coordination of complex networking systems, resources and services. It has an inherent intelligence and implicitly autonomic control of all systems, resources and services." They have a visionary aspect in their survey in terms of proposing multi-domain orchestration architecture, but they only focus on the components for realising multi-domain orchestration, they do not consider AI solutions for the end-to-end orchestration, and they do not include an edge computing aspect (storage and computing elements are seen as cloud nodes).

de Sousa et al. [24] provide a comprehensive survey of network service orchestration (NSO). They also propose a taxonomy for NSO solutions. They define NSO "as the automated management and control processes involved in end-to-end services deployment and operations performed mainly by telecommunication operators and service providers, involving different types of resources and potentially multiple operators". They regard orchestration to be responsible for decoupling high-level service/application layer from the underlying resources (physical and virtual). They see that orchestration consists of three functionalities: service, resource and lifecycle orchestration. Their taxonomy consists of seven main categories that they regard key features in NSO solutions: service models (the type of services related to NSO), software (software-related characteristics of the orchestration solutions), resource (the type of underlying resources for the network service deployment), technology (target technologies for NSO), scope (the application domain in terms of network segments considered by NSO), architecture (single- or multi-domain, organization and functions), and Standards Development Organization (standardization activities in scope of NSO). Their view is limited on service life cycle orchestration in cloud platforms.

Zhong et al. [26] provide an extensive survey of ML-based container orchestration approaches. They form a taxonomy for ML-based container orchestration solutions, consisting of five main categories: application architecture (the composition of containerized applications), cloud infrastructure, optimization objectives, behavior modelling and prediction (workload characterization, performance analysis, anomaly detection and dependency analysis), and resource provisioning operations for containerized applications. Their view is limited on the life cycle management of containers.

Versluis and Iosup [27] provide a systematic survey on workflow orchestration, focusing on taxonomizing four areas: workflow formalism, workflow allocation, resource provisioning, and applications and services. Formalism is concerned with the language used to represent workflows. Allocation places the workflows onto available resources in such a way that the scheduling targets are met. They see that allocation has five main categories: scheduling target, optimization strategy, scheduler structure, allocation technique and workflow instantiation. Resource provisioning is concerned about making decision when to allocate resources and how much given current and predicted demand. Applications and services are focused on the services provided by cloud, resource and environment types, and service execution models. Their paper has a strong emphasis on workload management in cloud infrastructures.

Costa et al. [29] systematically review fog orchestration literature and propose a generic architecture for fog orchestration. They define fog orchestration as "a management function responsible for service life cycle. To provide requested services to the user and assure the Service Level Agreements (SLAs), it must monitor the underlying infrastructure, react timely to its changes and comply with privacy and security rules". They analyze 50 papers focusing on five aspects: the goals of orchestration, orchestrated entities, control topology, architecture layers, and orchestration functionalities. Based on the results, they formulate a generic fog orchestration architecture in terms of functionalities that an orchestration framework should fulfill. They identify eight functionalities: admission control of incoming requests; service management to handle the life cycle of a service; resource management to provide the infrastructure on which services will be executed; monitoring of the resources; optimization in terms of implementing algorithms that satisfy different objectives under given constraints; communication management; a node agent to manage the node locally; and finally, security management. However, because their architecture is composed of the approaches found in the literature, they do not provide any concrete enablers towards realising all the components of the architecture.

III. Definitions and Scope

A. Artificial Intelligence

AI applications are commonly defined as computational agents, that is, processes with autonomy and intelligence. Agents are capable of pursuing their objectives rationally, relying on their individual capabilities. They acquire information on their environment through perceptions (e.g. sensor data or messages) and, based on their autonomous decision-making process, can affect the environment with actions [30].

An AI agent may possess various degrees of intelligence, defined by its reactivity, sociality, proactivity, and learning capability [31]. Reactivity refers to the agent's capability to react to changes in its environment, sociality to its interactions with other agents, and proactivity to its capability of taking initiative. Furthermore, learning allows the agent to evaluate its actions in the environment, and derive and explore new actions with the aim to reach its objective.

Agents possess individual knowledge of their (perhaps partially) observable environment, as well as of themselves and others. This knowledge is necessary for making decisions on actions and receiving and evaluating feedback of the actions, and subsequently, complex cognitive capabilities such as context-awareness and adaptivity in a dynamic open environment. In the scope of this article, such knowledge is encapsulated into objects referred to as models. The process of building and adapting these models, based on the agent's experience, is referred to as learning (or training).

Decision algorithms, which employ the learned models, encapsulate an agent's intelligence. Decision making uses the models to map objectives, percepts and previous actions to a new action which best increases the agent's probability of success. A popular choice is Reinforcement Learning (RL), which assumes the environment rewards for successful actions. RL models usually comprise one or both of an actor (or policy) model, which aims to find the appropriate action, and a critic (or value) model, which estimates the accumulating reward, given some choice of actions [32].

B. Distributed AI

MAS distribute the functionality of an application among a number of autonomous agents. Such distribution is necessary in an environment where individual agents do not have enough information or resources to achieve their objectives. Instead, the agents must cooperate on their individuals objectives and collaborate on shared ones, communicating their understanding of the environment and their progress towards the objectives.

Distributed application logic is implemented with agents that have distinct roles and behavior. The agents negotiate and share information and/or resources with each other to coordinate their efforts. Furthermore, a MAS can have an open or a closed organizational structure, which governs relationships, rules, objectives, policies and authority [31]. Such policies may comprise, for example, how agent interactions are conducted, and what an agent can expect from others. The behavior of a MAS thus emerges through the actions and interactions of autonomous or partially autonomous individual agents, with the guidance of an orchestrator or through a choreography of the autonomous participants. In open systems, such as the Internet of Things (IoT), a MAS often needs to dynamically reorganize itself to adapt and to evolve, in response to changes in the participating agents or in the environment. These aspects ultimately facilitate individual and collaborative learning to improve operations towards common objectives and proactive behavior(s).

C. Computing continuum and orchestration

Computational resources span over the network infrastructure, all the way from a central data center to the user at the edge of the network. Several architectural approaches, including fog and edge computing, Mobile Cloud Computing (MCC), and MEC [7, 33], take advantage of these resources, expanding the cloud computing paradigm. Although each approach encompasses its own paradigm and requirements, they bring services closer to the user while simultaneously addressing challenges inherent in edge application deployments such as latency requirements, bandwidth constraints or energy utilization.

The cloud, with ample resources for computing and storage, is still often a necessity, calling for hybrid edge–cloud architectures [34, 35, 36], or even a continuum of computational resources between the devices and the cloud where applications can choose the best resource usage policy based on current needs [37, 38]. Accordingly, in this article, we collectively refer to edge and fog computing, MEC, and other similar distributed computing approaches with heterogeneous and opportunistic resources by the term *computing continuum*.

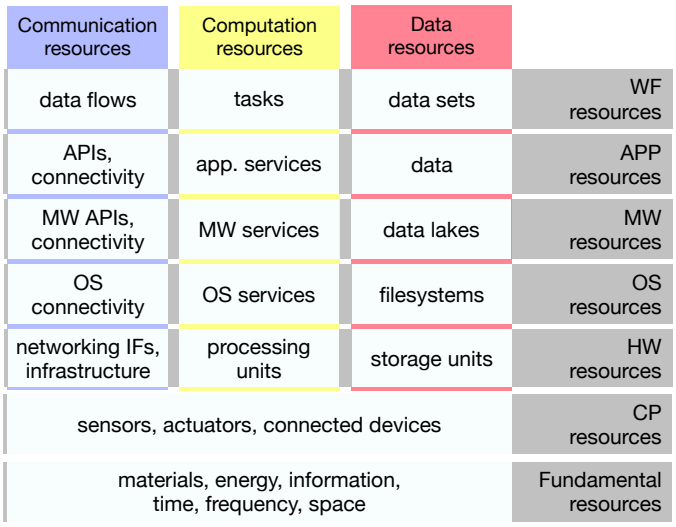

Fig. 2: **Resources in the computing continuum.** The resources reside on any of a number of abstraction layers, from fundamental resources such as energy or time, to cyber-physical (CP), hardware (HW), operating system (OS), middleware (MW) and application layers (APP) or, finally, the layer of resources related to a particular application workflow (WF). In this hierarchy, higher-level resources rely on the lower-level ones, using them to fulfill their function. Further, the resources can be grouped as per their usage in three categories, namely, communication, computation and data resources.

Synthesizing the taxonomies in a number of recent works (see e.g. [28, 29, 25, 23, 26], and those mentioned in Section II-B), in this article, we take a holistic approach to the resources in the computing continuum. These resources, as depicted in Fig. 2, are present on a number of levels, ranging from fundamental resources such as energy or time (see [30]), through cyber-physical (CP), hardware (HW), operating system (OS), middleware (MW) and application (APP) resources, finally to workflow (WF) resources catering to the highest level application business logic or clients. In this hierarchy of levels, higher level resources rely on the lower ones to fulfill their function. Further, from hardware level up, the resources can be divided in three distinct categories, namely, communication, computation, and data-related.

It should be noted that this hierarchy of levels does not constitute a layered architecture in the sense that a level would only be aware of its immediate lower level. For example, data sets on the workflow level may be sourced from sensors on the cyber-physical level.

This holistic viewpoint is not emphasized in many of the related studies. These studies (see Section II-B) often refer to entities on hardware and operating systems levels as resources,

and entities on middleware and application levels as services. However, there is considerable ambiguity in these conventions, and we find that an explicit consideration of EaaR (Everything as a Resource) simplifies the overall view.

The cyber-physical resources of a computing continuum may comprise sensors, actuators, and other connected user devices which have a physical form and function. They may act as sources (sensors) or sinks (actuators) of data flows in the computing continuum.

Hardware resources in the communication category comprise, for example, network interfaces, access points, and base stations. The computational hardware resources refer to processing units (e.g., CPU, GPU, AI related accelerators), whereas data-related hardware resources include, for example, hard drives and SSDs.

Operating system resources include, for example, connections (and related abstractions such as sockets), operating system services such as processes (threads), and filesystems, as well as support for VM and containers.

Middleware resources in the communication category include, for example, sessions, Virtual Private Networks (VPN), and slices of the mobile networks [2, 3]. Computational resources include middleware services offered to applications, such as those available in the edge/fog computing frameworks (e.g. ETSI MEC [39]), as well as container frameworks such as Kubernetes or Docker Swarm (see Section IV-B). Data resources in the middleware layer refers to, for example, databases, data warehouses, or data lakes, and distributed file systems. The middleware resources are provided through OS services, libraries, and local and remote APIs.

Application resources include higher-level APIs and network overlays, application services, often packaged as containerized microservices or serverless functions, the data generated and consumed by the application.

Workflow resources are the highest-level category, consisting of the data flows, tasks, and data sets available for individual application workflows initiated by application clients or business logic.

A simplified example of two IoT applications in the computing continuum is depicted in Fig. 3. The workflows of the applications start with data lifted from sensors. The data is processed in a sequence of tasks, running on containerized services in edge devices or cloud-based serverless functions. The containerization frameworks and the serverless functions are provided by two mobile network operators and two cloud providers, respectively. Both workflows finally end on actuators. In the depicted example, the applications share some of their sensors with each other.

Managing these resources in the computing continuum is often referred to as *orchestration*. In more detail, the term orchestration is used to refer to functions such as the automated management (i.e., configuring and coordination) of complex services, dynamic resource allocation, efficient and optimized resources utilization, control of functions, or real-time service delivery [22, 2]. However, related work often scopes orchestration to certain aspects of the continuum, such as networks and connections, application services, or tasks and workflows. Network orchestration thus refers to the configuration and management of communication networks. In contrast, service orchestration refers to the management and configuration of the life cycle of application components encapsulated as services.

In this article, again synthesizing the concepts presented in recent studies on orchestration [28, 29, 25, 23, 26], we holistically define orchestration as the management of resources in the computing continuum, from fundamental resources to workflow resources, as depicted in Fig. 2. This management can be further divided into a number of distinct *functions* (Fig. 4), with particular *attributes*, aiming for a set of possible *objectives*, set by different *stakeholders*.

Orchestration functions include: lifecycle management, comprising functionality such as creating, deleting, starting, stopping, or updating resources; allocation, comprising for example the placement of resources, scheduling access to them, or their migration, scaling and replication; discovery, supporting the registration and subsequent lookup of available resources; dataflow management, with functions to, e.g., aggregation, sharing, offloading and caching of data resources; and monitoring, which keeps track of the state and capabilities of each resource, and estimates their performance.

Furthermore, these functions may be implemented in various ways. The implementations may differ from each other, and these differences may be described by attributes such as security, privacy, or fault tolerance.

Moreover, orchestration may rely on three different types of control topologies, the first a centralized control topology, a decentralized one, or a fully distributed (e.g., peer-to-peer) one [29].

Dustdar et al. [38] use Resources, Cost, and Quality as the three fundamental objective categories, which we adopt here. Further, the objectives of orchestration are often related to the essential resources (i.e., spectrum, time, energy, or funds) and can often be described as a combination of corresponding budgets or as a relation between the budgets and resource usage [23]. For example, KPIs related to the QoS, most often expressed as latency or throughput, are related to the time budget for an application running in the continuum.

Finally, orchestration may be multi-domain, with a number of administrative domains, and many possible stakeholders setting the objectives. These stakeholders may comprise, e.g., end-users, owners of devices, application providers, or infrastructure providers such as mobile network operators (MNOs), Internet service providers (ISPs), or edge operators.

D. Autonomy

Autonomy refers to an agent's ability to make decisions without any influence of external authority such as users, administrators or other agents. While such decisions can be made by the agent on the behalf of the external authority, that authority itself cannot influence a fully autonomous agent making the decision [40].

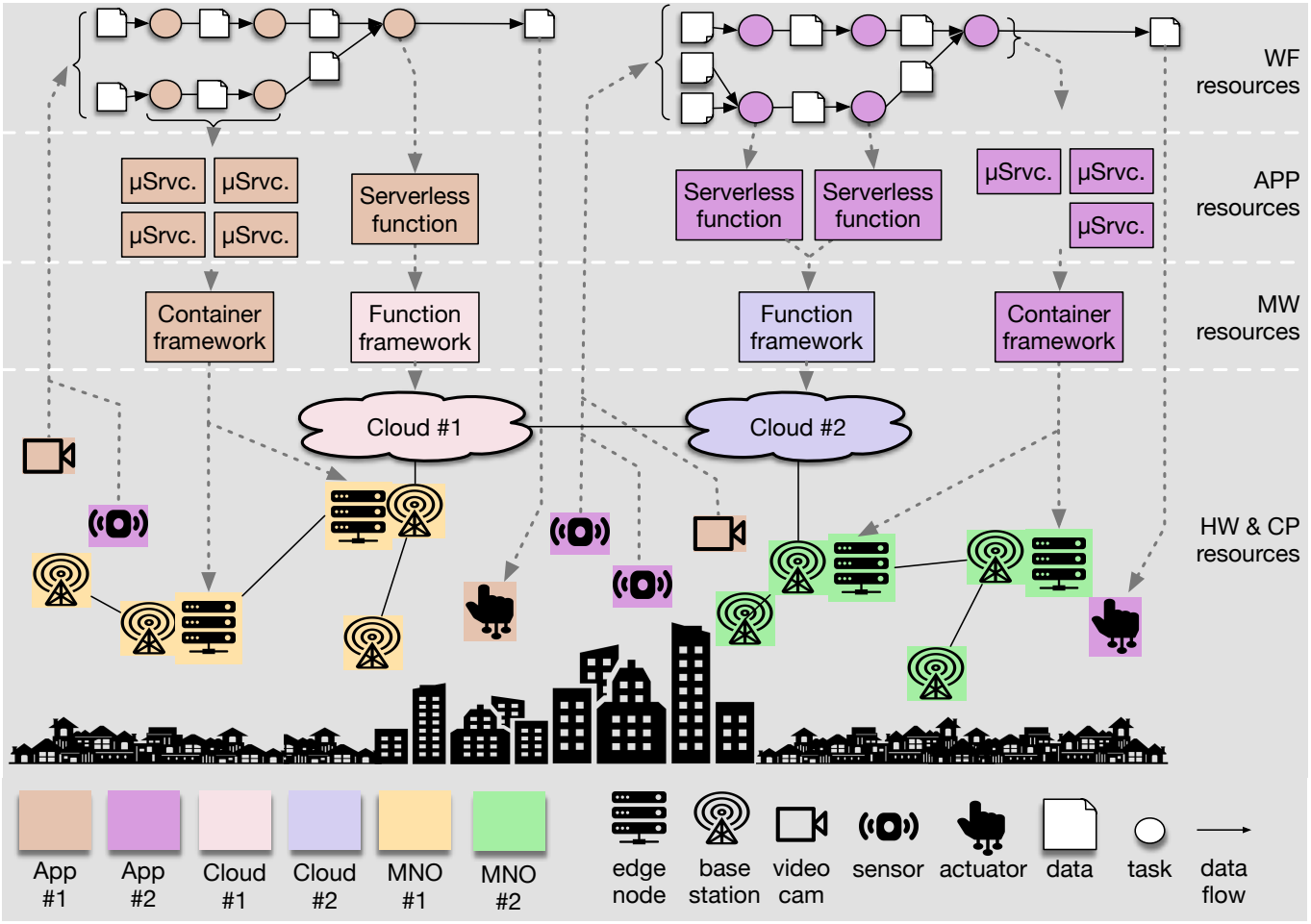

Fig. 3: **Two example IoT applications in the computing continuum.** The workflows of two IoT applications, comprising tasks, data flows and data sets, are executed in containerized microservices and serverless functions, deployed on edge and cloud resources provided by Mobile Network Operators (MNOs) and cloud providers. Hardware (HW) and cyber-physical (CP) resources such as sensors, actuators, edge nodes and base stations are deployed in the vicinity of the users. Color coding corresponds to administrative domains.

Many large scale complex systems make use of autonomy to cope with complexity. For example, the internet routing protocol [41], a supremely complex system responsible for managing internet traffic, heavily relies on the autonomy of the individual routers. Furthermore, autonomy makes distributed systems fault-tolerant, i.e., robust against the failure of one or more of its components [42].

However, unlike traditional distributed systems, the computing continuum system is characterized by multiple computing tiers (e.g. cloud, edge/fog, IoT), and the complexity arising from the entanglement of the highly heterogeneous infrastructure underlying these tiers. Building on the concept of autonomy in distributed system, we need to identify the benefits of autonomy in the highly heterogeneous and complex ecosystem of the computing continuum.

MASs, composed of interacting agents each of whom is autonomous, provide a viewpoint to autonomy in the computing continuum systems. We thus adopt the definition of autonomy by Carabelea et al., explained as *An agent* X *is autonomous with respect to agent* Y *for decision* p *in the context* C*, if, in* C*, its behaviour regarding* p *is not imposed by* Y [43].

E. Scope

This survey is located in the intersection of distributed AI and computing in the device-edge-cloud continuum. We focus on distributed solutions for the orchestration of the computing continuum, modelling a deployment in the continuum as a network of connected resources, on a number of abstraction layers as depicted in Figs. 2 and 4.

Orchestration in the computing continuum is increasingly complex. To address this complexity, and to increase system fault tolerance, the system must adopt autonomy. Local components must be able to operate autonomously, not relying on resources at other sites, while still keeping in synchronization with their global objectives.

We thus envision the computing continuum system consisting of autonomous, intelligent agents. These agents may reside

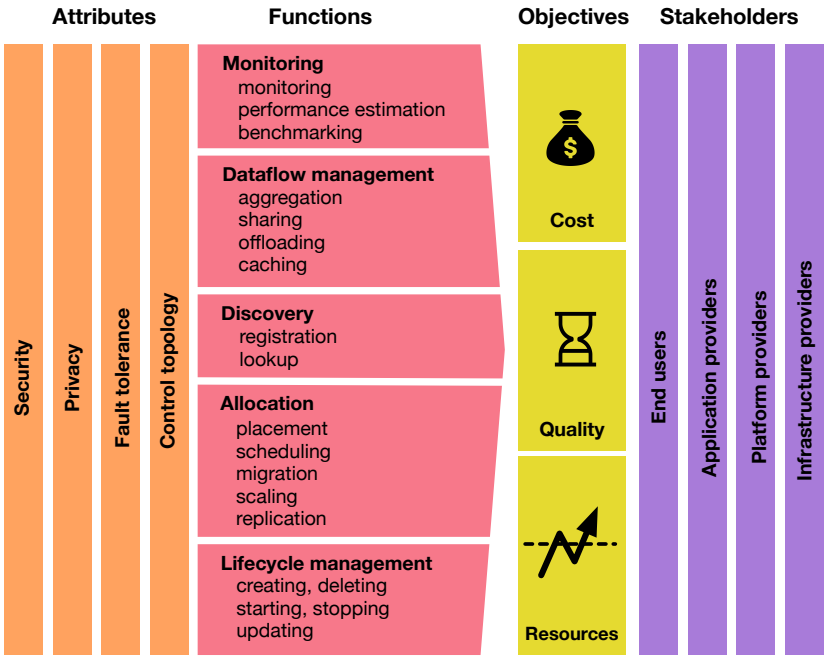

Fig. 4: **Continuum orchestration taxonomy.** Orchestration, that is, managing resources in the computing continuum, can be divided to a number of functions such as lifecycle management or monitoring, as well as overarching attributes such as security or privacy. Orchestration aims to reach certain objectives, set by a number of possible stakeholders such as end users and infrastructure providers.

on any of the computing tiers and resource abstraction layers (Fig. 2), and they act as service endpoints in their own roles, as advocated by Xiong et al. in their vision of decentralized internet of things [34]. Each of these agents maintains its objectives privately, and acts autonomously to maximize its selfish interest, i.e., to meet its objectives as best as it can.

We thus consider the solutions to comprise a MAS, and be composed of intelligent and autonomous or semi-autonomous agents with high reactivity, proactivity, sociality, and learning capabilities.

Deep learning can be regarded as a significant key factor behind smarter edge platform management [16]. Hence, this survey mainly assumes that the agent models are Artificial Neural Network (ANN) models; however, other parameterized models are also possible.

Finally, this survey also touches upon security and privacy aspects in developing an autonomous computing continuum. The main emphasis will be on what type of security issues the integration of AI with the edge environment causes, and what are the potential ways to mitigate them.

IV. Orchestration Paradigms

A. Control topologies

Three main paradigms cover the control topologies of the computing continuum orchestration: centralized, decentralized, and distributed control. Of these three, centralized orchestration is similar to classical master–worker models, defining an explicit controller component that manages resource allocation and task assignment between the nodes.

Liu et al. [44] designed an edge orchestrator that was able to overcome the resource limitations of individual mobile devices (nodes) by offloading computational tasks to an external server. The orchestrator receives task allocation petitions, and an optimization algorithm is invoked for optimal resource allocation in the worker external server.

Analogously, Xiong et al. [34] presented KubeEdge as a hybrid cloud–edge platform where an edge orchestrator was integrated for resource allocation from a cloud environment. The infrastructure was able to coordinate computing resources from the cloud to fulfill the requirements of the edge tasks, resulting in an improved performance in containerized applications.[1]

Conversely, in distributed architectures, nodes decide by themselves what tasks to allocate and how global resources are distributed based on the priority of individual tasks and resource availability. Distributed control[2] is based on resource allocation algorithms that omit edge controllers entirely. These resource allocation algorithms are distributed over the nodes, which negotiate to reach a consensus on the optimal task allocation. Agreed tasks are then distributed over the computational resources available in the continuum.

A paradigmatic example is the algorithm developed by Castellano et al., called *Distributed resource assignment and orchestration algorithm* (DRAGON) [48]. This resource al-

[1]While the KubeEdge approach is mostly centralized, some elements were related to a decentralized architecture, see discussion in Section IV-E.

[2]Distributed orchestgration is sometimes also referred to as *choreography*, see e.g. [45, 46, 47].

location algorithm allows a set of applications to reach an agreement on how edge infrastructure resources have to be assigned without the need for a centralized orchestrator. Each application has a DRAGON agent that starts a voting procedure to acquire its resources to deploy a given task. Then, each application participates in a resource election protocol where the winning applications allocate their demanded resources on a certain physical node.

Another example of a resource distribution algorithm is an approach developed by Fizza et al. [49], which allocates incoming tasks by prioritizing deadline requirements while taking into account privacy constraints. The Privacy Aware Scheduling in a Heterogeneous Fog Environment (PASHE) algorithm can schedule resource allocation into heterogeneous edge devices (each with a different computation capability) by offloading public loose-deadline requiring tasks into centralized cloud environments and private urgent tasks to be performed locally.

Other aspects apart from resource capabilities can be considered during orchestration or even prioritized over resource management. An example of this is the work by Auluck et al. [50], who developed an orchestrator that dynamically allocated the incoming tasks attending to deadline and security aspects. Workloads are performed in edge data centers (more insecure) or cloud data centers (more secure) depending on the security tags of each task and the expected delay of the network.

Decentralized approaches are in the middle of centralized and distributed, with a hierarchy between a small number of edge controllers and the resources each of these control [29]. In more detail, decentralized control partitions the edge devices in groups such that each group has a designated local edge controller. The local controllers subsequently co-operate with their peers to reach global goals.

An example of this is the work by Tocze et al. [51]. They designed an edge orchestrator for task allocation (ORCH Framework), which addresses the problem of Distributed Dynamic Task and Mobile Edge Placement (D2TEP). The ORCH framework divides the Edge area into edge device neighborhoods, each orchestrated by an area orchestrator. By combining stationary and mobile edge devices, the framework can flexibly serve all edge tasks (either mobile or stationary). This approach is suitable for architectures with a vast number of edge devices, for which voting processes may take too long, or edge scenarios where node churn (devices joining and leaving) is expected, so that they can be included in a certain orchestration group (by proximity, for instance). Such decentralized orchestrators do not have any adaptation algorithms for changing their strategies, making them very robust in performance for specific tasks. On the other hand, that lack of adaptation may be too rigid for dynamic environments.

There is no consensus on which control topology best suits the computing continuum. The election of one approach over the others may mainly depend on the available resources and the number of nodes these resources have to be shared. However, other aspects besides those must be considered, such as low-latency requirements, communication protocols to be used between nodes, the distance between equivalent (the same level in the hierarchy, if there exists one) physical nodes, and the heterogeneity of the applications and services to be deployed on each node. Ultimately, the decision should come from a comprehensive analysis of the requirements and characteristics of the architecture to be implemented.

Security aspects must also be considered when comparing orchestration strategies. Centralized architectures rely on a single operation controller, making the controller a prominent target for external attacks. Moreover, network failures and cloud unavailability are significant risks that would disable the orchestration capabilities of the centralized cloud-based orchestrators. On the other hand, distributed approaches are autonomous and fault-tolerant at network and cloud levels. Still, they are more resource-demanding as they require communication between a large group of agents to reach a consensus. Furthermore, resource optimization algorithms have to be constantly running, which increases the overall resource consumption of orchestration tasks [52].

Resource optimization algorithms in distributed architectures also sometimes suffer from too long runtimes in reaching decisions when many edge devices are involved. Communication times between distant devices and a sufficiently high number of devices in the network can result in voting processes and task allocation strategies taking too long for tight deadlines to be fulfilled.

Finally, communication overhead may also be a determinant factor. Centralized architectures usually have a higher communication overhead than the distributed ones [52], as they need a high amount of information to be shared between each of the nodes and the central orchestrator (tasks to be deployed, available resources, latency requirements, deadlines...). This intrinsic communication overhead could be a problem in systems with a demand for edge dynamism and low latency.

B. Container orchestration

A container is a software encapsulation of the processes related to an application service. Containers allow running microservice-based applications reliably, in isolation of the processes of the underlying operating system or other containers, as well as enable easy migration from one computing environment to another [53]. Furthermore, containerization can be deployed to provide computational capabilities for systems in the computing continuum, enabling batch processing at scale, control planes, as well as IoT and AI workloads [54].

Microservice-based applications that are often run in clusters of hundreds of geographically distributed containers must be fault-tolerant and available [53]. Managing the life-cycle of such a high number of containers, scaling them up and down, as well as replicating, migrating, starting, and stopping them as necessary is getting increasingly complex [53]. To overcome this complexity, container orchestration systems have been developed to manage the lifecycle and workflow of the containers in an automated manner in large and dynamic environments [55].

In more detail, container orchestration refers to controlling and automating the deployment of containerized resources taking into account different requirements such as availability, scaling, and networking. The functions related to container orchestration include scaling containers up and down across the host infrastructure, scheduling and managing clusters, migrating containers from one host to another, allocating resources between the containers, load balancing, monitoring containers, managing networking overlays, and providing security [53].

Furthermore, container orchestration must meet certain standards related to secure networking between edge and cloud without impacting its local network and require low resources in terms of memory, processing power, and storage [56].

To meet these requirements, container orchestration minimizes the use of communication, computation, and storage resources such as memory, CPU/GPU, disk space, volumes (i.e., interacting with local and remote file systems), as well as ports and IPs that refer to the configuration of IP addresses and application ports in containers. While allocating and guaranteeing the necessary resources for the containers, container orchestration must also optimize the resource usage requirement of the orchestration process itself.

Currently, there are several orchestration platforms available in the market. Kubernetes[3], Docker Swarm[4] and Apache Mesos[5] are the best known container orchestration platforms. While all of these platforms are open source, Kubernetes and Docker Swarm are the major players in the container orchestration. At the same time, Apache Mesos is a seminal orchestration platform that enables dynamic resource sharing between microservice-based applications.

In more detail, Kubernetes is an open-source container orchestration platform introduced by Google to automate, deploy, manage and scale containerized applications across a cluster of nodes/machines [57]. Based on a centralized control topology with a master-slave architecture, the Kubernetes master node works as control plane, managing the cluster by deploying applications to slave nodes. The slave nodes, in turn, host and run the application containers. To operate, Kubernetes uses a set of objects called "Pods," which are the basic control unit of Kubernetes. Pods consist of containers that share the same processes, interfaces, IP addresses, ports, and memory.

Furthermore, Kubernetes manages resource scheduling and networking. Resource scheduling here refers to detecting and utilizing the available pods at the nodes. At the same time, networking facilitates communication between containers, using container network interfaces through overlay networks [58]. Kubernetes thus enables pods to communicate with each other.

Docker Swarm is a container orchestration framework built especially for Docker[6]-based containers. Docker Swarm functions by clustering and scheduling Docker containers deployed across multiple host nodes/machines. Docker Swarm shines in its fast deployment, simplicity, and high availability [59].

The control topology of Docker Swarm is also centralized and based on the master-slave model. A master node is responsible for the entire cluster, keeping track of the status of the slave nodes, and scheduling the containers. The slave nodes are responsible for launching the received containers. To select a slave node, Docker swarm applies a strategy that considers the resource requirements (such as CPU and memory) of the service it will deliver. The examples of the selection strategies include the default *Spread* strategy, which selects the node having the least number of containers; the *Binpack* strategy that selects the most packed containers; and the *Random* strategy that randomly selects a slave [59].

In a cluster, to connect containers that are hosted on different nodes, the Docker swarm uses an overlay network. The network allows the master node to determine which slave nodes are still functional. Furthermore, the Docker swarm has a load balancer on every node to balance the workload across nodes and containers.

Apache Mesos, originally developed by the University of California, Berkeley, is a container orchestration platform that manages the workloads and applications in large-scale clustered environments. Apache Mesos manages workloads by node abstraction, bringing the resources of different nodes into a single pool of resources and eliminating the need for assigning specific nodes per workload. To do this, Mesos separates the processes such as CPU, memory, and file system which run in a cluster and keeps these processes from interfering with each other [60].

In summary, Kubernetes, Docker Swarm, and Apache Mesos all rely on a centralized control topology. Further, they all descend from the cloud-native container orchestration paradigm, emphasizing container life cycle management while mostly disregarding the continuum environment characterized by, for example, intermittent communication, heterogeneous and geographically distributed computing, and siloed and non-IID data. These limitations make them and, indeed, the container orchestration paradigm as a whole unsuitable for taking full responsibility for the orchestration in the computing continuum. However, as noted by Costa et al., a container orchestration system could be used as a component of a broader orchestration framework in the continuum [29].

C. Workflow management systems

Workflow management systems refer to software systems that facilitate the coordination of workflows. In more detail, workflow management systems offer an infrastructure for setting up, specifying, executing, and monitoring workflows, which comprise tasks that run on available computing resources [61, 62, 63].

Workflow management, as studied by Versluis and Iosup [27], comprises a formalism, that is, a language such as CWL, DAX, or YAWL to describe workflows, as well as systems implementing that formalism. Furthermore, a system doing workflow management implements a number of different

[3]https://kubernetes.io/
[4]https://docs.docker.com/engine/swarm/
[5]https://mesos.apache.org/
[6]https://www.docker.com/

functions such as scheduling, optimization, and instantiation of workflows and tasks. Workflow management may have many different targets, such as cost, latency, resource usage, or load balancing, and the schedulers themselves may follow a number of different control topologies, including centralized, decentralized, or distributed.

As such, workflow management systems follow closely the taxonomy described in Section III-C. The difference comes in scope: workflow management focuses on the workflow resources and application resources layers in Fig. 2, largely omitting those below.

Apache Airflow[7] is an Apache Software Foundation project developed by Airbnb in 2016. Apache Airflow is designed to offer a lightweight workflow management system to model, maintain, and monitor workflows. Airflow executes each workflow as a directed acyclic graph of tasks. These tasks are usually atomic, run independently, and do not exchange data with each other. Airflow can scale horizontally on clusters orchestrated by Apache Mesos. It can be utilized to execute workflows in diverse computing platforms such as workstations, edge, and cloud platforms. It also offers multiple interfaces to commonly used cloud environments such as Amazon S3, Google Cloud, or HDFS. This feature enables the users to access and utilize multiple clouds only with one deployment [64].

Kubeflow[8] is an open-source workflow orchestration tool based on the Kubernetes container orchestration environment. It is designed to easily deploy ML workflows on a Kubernetes cluster. Kubeflow functionality echoes that of Kubernetes, as it aims to deploy, scale, and manage ML workloads [65]. Kubeflow also allows running automated ML tasks and supports hyperparameter tuning [66], thus supporting end-to-end ML workflows [67].

Furthermore, Kubeflow aims to evaluate the end-to-end performance of deployed workflows with seven different ML models, evaluating resource consumption in terms of GPU utilization and time requirements for CI/CD (i.e., continuous integration and continuous delivery) pipelines [68].

MLflow[9] is another open-source workflow management platform. It is developed by Databricks and designed to manage and streamline the complete ML lifecycle, enabling users to bring their software and workflows into the ML lifecycle [69].

MLflow is composed of four components, which can be utilized individually or together. MLflow Tracking component is an API that records experiments, tracking data such as parameters or input data. MLflow Models is a generic model packaging format that can operate across diverse ML environments. MLflow Projects is a packaging format that uses a YAML configuration file for packaging software into reusable projects. MLflow Model Registry component is a collaborative hub that manages the lifecycles of ML model deployments [70].

[7]https://airflow.apache.org/

[8]https://www.kubeflow.org/

[9]https://mlflow.org/

However, none of the above workflow management systems consider the communication, computation, and data-related challenges in the computing continuum. We were able to find only one proposal related to learning on heterogeneous devices, on non-IID data [71]. However, that proposal was restricted to FL systems only.

D. Network orchestration

Network orchestration refers to the automated management and control processes for the deployment and operation of end-to-end services in telecommunication networks [24]. As such, network orchestration must consider both the low-level resources (such as bandwidth, networking interfaces, CPUs, GPUs, storage) available on the fundamental and HW resource layers of Fig. 2, network services (e.g., virtualized network functions) using those resources [72] on the OS and MW layers of Fig. 2, and finally the application workflows using the services on the Application and Workflow layers Fig. 2 [24].

In essence, network orchestration maps the user service requests to underlying resources on a number of layers, thus scheduling access and sharing the virtual and physical resources in the network [73]. Further, network orchestration is responsible for managing the life cycle of network services such as virtual network functions (VNFs), using a set of orchestration functions such as registering, instantiating, scaling, updating, and terminating [73].

Open, standardized network orchestration is currently an emerging area for research. Many studies are devoted to efficiently performing resource utilization and orchestration in the networks. Hirwe et al. [74], for example, aim at minimizing load balancing using a run time procedure in order to optimize VNF placement and service chaining in NFV. Kuo et al. [75] aim to minimize resource utilization through optimizing a joint problem of VNF placement and routing in the network. Pham et al. [76] propose a game-theory-based theoretical model to capture the competition among network service providers in a multi-domain NFV.

Furthermore, Salhab et al. [77] propose a predictive approach for network slicing, aiming for optimal decisions for network slicing to dynamically utilize shared network resources. Bari et al. [78] identify the optimal number and placement of VNFs to improve operational costs and network resource utilization. To orchestrate the virtual resources and network functions, Guerzoni et al. [79] propose embedding a virtual network to deal with the mapping of virtual resources on the physical infrastructure in a network. Finally, Khan et al. [80] develop an algorithm based on graph neural networks to estimate resource requirements for VNFs. They propose an intent-based system that automates the orchestration of end-to-end workflows and services across multiple orchestrates.

However, all the above studies focus on orchestrating the communication resources and the computational and data resources related to implementing communication-related services (e.g., VNFs) realising the network objectives SLAs.

There is little consideration towards considering the applications setting those objectives, who need to balance their communication requirements with their computational and data resources. Further, even within the orchestration of communication resources, interoperability between vendors and operators remains an open issue [81].

E. Edge/fog orchestration

A number of approaches attempt to address the shortcomings of the above paradigms in the computing continuum. For example, Costa et al. [29] draft a common architecture for fog orchestration, synthesizing dozens of proposals. The draft emphasizes admission control, service and resource management, monitoring, optimization, and communication management, covering most of the functionality in Fig. 4 while omitting container life-cycle management, as well as the orchestration of the lower layers of network resources (see Fig. 2). Further, at this early stage, the proposed architecture is still missing details such as the control topology.

In addition to a number of academic proposals (see, e.g., [23, 25, 29]), a number of frameworks are already available for developers or in the various stages of standardization. The ETSI MEC [39] standard defines an orchestration architecture for Multi-Access Edge Computing. In the ETSI MEC architecture, a centralized orchestrator is responsible for monitoring, lifecycle management, scheduling, and migration. Further, the standard defines interoperation with, e.g., cloud systems, defining functionality such as migration of applications between cloud and MEC [82]. However, ETSI MEC originates from the telecom industry and, as such, may be unsuitable for the orchestration in the fog domain or of low-resource IoT devices.

A number of approaches such as MicroK8s[10], KubeEdge[11], and OpenYurt[12] extend the container orchestration framework Kubernetes towards edge devices, albeit with different architectural approaches. MicroK8s provides a lightweight but full Kubernetes implementation, extending the cloud to the edge resources with enough resources to support the streamlined framework. KubeEdge and OpenYurt both introduce a new abstraction layer for the edge resources and connect this layer to a cloud-based main Kubernetes cluster. However, while KubeEdge implements the edge layer with custom components, OpenYurt instead adopts Kubernetes native plug-in and operator mechanisms for the implementation.

Unlike MicroK8s, which relies on Kubernetes' centralized control topology, with their newly-introduced edge layers, both KubeEdge and OpenYurt are somewhat akin to a decentralized one. Indeed, edge clusters in KubeEdge and OpenYurt have limited autonomy in case their cloud connections are severed. However, both architectures rely on containerization, which is often unsupported on the most lightweight devices. Including these devices in the orchestration framework would thus require custom solutions.

[10]https://microk8s.io/

[11]https://kubeedge.io/en/

[12]https://openyurt.io/

F. Emerging paradigms

1) Multi-domain orchestration: Multi-domain orchestration is an emerging computing paradigm that aims to provision end-to-end network service delivery across multiple (more than one) infrastructure providers, each one having their own administrative domains. Such paradigm is particularly useful for upcoming resource-hungry applications, such as holographic applications or tactile internet, where provisioning necessary computing, storage, and communication resources that meet the required quality of service by the application developers and the desired quality of experience by the end users may not be located within an administrative boundary of a single cloud service provider.

Current standard architectural Frameworks, such as ETSI Management and Orchestration (MANO) [83] provide tools and setups to provision virtual network functions, their configurations, and deployment within the infrastructure of a single cloud service provider. Another architectural framework ETSI Multi-access Edge Computing (MEC) [84] allows applications to be virtualized at the edge and access the network. This virtualization at the edge within ETSI MEC framework could be implemented as virtual network functions within the ETSI MANO framework. However, both of these architectures fall short in assembling functionalities provided by multiple service providers into one single function which could operate over multiple infrastructure in end-to-end fashion, owned by different service provider.

Recent work by Francescon et. al. [85], X–MANO presents a proof-of-concept for cross-domain network service orchestration. It introduces an information model which allows network network service provider to advertise its resources to other network service providers in a privacy preserving manner. For this, it introduces a Multi–Domain Network Service Descriptor (MDNS) which lets service provider to expose its network services without revealing internal implementation details. It introduces programmable network service to enable network service providers to implement their custom life-cycle management polices.

Genez et. al. [86] proposes scheduling mechanism based on time discretization with controlled granularity to allocate resources for a service from using resources owned by multiple service providers. It models resource scheduling as a directed acycle graph where nodes represent specific jobs of the service and edges represent dependency among them, and provides a solution for the scenario where these nodes may be owned by different administrative domains.

Osmani et. al. [87] proposes an extension to Kubernetes, a popular choice for container orchestration. It proposes a Federated Kubernetes framework for building a unified management mechanism for multiple Kubernetes clusters, where each individual Kubernetes cluster is owned by a different service provider, and is subject to their control polices. To build this unified management mechanism, it proposes the Network Service Mesh, a framework that providers a secondary network connectivity among Kubernetes containers, and ser-

vices various roles such as client, endpoint, control manager, data manager in different domains to facilitate inter-domain interactions. Moreover, Subramanya and Riggio [88] propose AI-driven Kubernetes-based orchestration which uses time-series forecasting techniques to manage dynamic resource up-scaling in multi-domain scenarios.

While from a resource orchestration point of view, multi-domain orchestration is an ideal option for resource-hungry applications in the upcoming metaverse age, where resources available locally may not be enough, it poses challenges in terms of privacy, security, heterogeneity, and how to align conflicting interests of different service providers at the same time. A full fledged multi-domain orchestration system could be the final realization of the computing continuum system, however current efforts focus mostly on either network or container orchestration. These cover only a subset of the full range of orchestration requirements; further, containerization is often not supported on the wide range of lightweight devices without including custom solutions.

2) FaaS orchestration: Function as a Service (FaaS) is a cloud computing service under the emerging paradigm of serverless architecture. FaaS allows application developers to develop, run and manage individual, isolated application functionalities at the cloud [89]. It offloads the responsibility of building and managing the infrastructure associated with developing and running the application from the application developer to the cloud service provider. Developers may write and update application code on the fly as microservices, which can then be executed in response to events. Moreover, these events may be triggered by the application user in real-time, with, say, user clicking a particular element in the application GUI.

In a nutshell, FaaS enables application developers to focus only on developing the application, while the cloud service provider takes the complete responsibility of the resource allocation and run-time management, as well as the security measures. As such, FaaS offers the benefits of faster development turnaround to the developer. Furthermore, FaaS promises built-in scalability, as the cloud service provider manages application resources, and scales them up under sudden bursts of demand. In addition to simplifying developing, this also reduces costs [90].

These benefits make FaaS ideal for applications, especially those which provide basic utilities, such as Google Maps. However, FaaS has its disadvantages. Application developers may need to write their applications to be compatible with the back-end infrastructure of the cloud service provider, enforcing a vendor lock-in. Moreover, debugging and testing becomes complex since developers always need to cope with changes in the infrastructure of the cloud service providers.

Over recent years, many commercial platforms and open source platforms supporting FaaS have emerged. Among commercial platforms, Amazon AWS Lambda holds the leading position in terms of market share and a range of services it enables. In FaaS, application developers define a workflow which may consist of a single or multiple functions [28]. This workflow can be interpreted as a state machine. To orchestrate the state machine, cloud server uses lightweight services, encapsulated as containers or micro-virtual machines, which execute the functions defined in the state machine when triggered by an event, say, a user clicking on an application component which invokes the function. Each service may host one or more functions depending on the amount of memory each function requires. Orchestration of the services follows the strategy of bin packing problem, placing different functions to services in order to maximize memory utilization. While FaaS platforms follow this strategy on a high level, they differ in their implementation details, such as their strategies for resource allocation, billing, and tools provided to application developers to write the workflow of their functions [91].

From an orchestration point of view, FaaS partly simplifies the resource management, due to the statelessness of the services, while also extending the scope of container orchestration towards that of workflow management. Moreover, in the computing continuum, the lightweight FaaS functions can be hosted at nearby IoT or edge nodes, thus benefiting objectives related to, e.g., latency. Statelessness could enable different functions of any given application to be hosted on devices across different tiers, thus offering the benefit of portability.

However, FaaS poses a challenge for applications with heavy context or state data, such as Augmented Reality (AR)/Virtual Reality (VR), or those employing AI/ML models. Indeed, migrating these functions may need moving large data sets across the network [92]. Furthermore, orchestrating workflows over these functions requires careful consideration of how the state information in the workflows is maintained. While the full view of FaaS orchestration is still largely an unsolved problem, developers and cloud service providers may need to consider caching the data geographically close their user base, a strategy which is often used by video streaming applications like Netflix [93].

3) WASM: WebAssembly (WASM) is a portable low-level bytecode which offers a compact representation, efficient validation and compilation, and safe execution with little overhead [94]. While originally designed to run in web browsers, WASM has recently gained traction as a general-purpose compilation target for a number of programming languages and run environments [95].

As such, WASM could provide an alternative to containerization, especially in resource poor edge devices incapable of supporting containers or container orchestration, and especially for FaaS workloads [96]. While some early cloud based orchestration frameworks for WASM exist (see, e.g., discussion on WASMcloud by Rac and Brorsson [97]), to the best of our knowledge, there are currently no orchestration frameworks available for applications based on distributed WASM runtimes in the computing continuum.

V. Research Themes and Challenges

In this section we look at certain emerging research themes that may affect the future of orchestration, and synthesize the

themes to provide an early vision of a future continuum orchestration paradigm. Furthermore, we describe the challenges related to the emerging research themes and the vision.

A. Computing Continuum management model

Computing continuum invalidates the idea of having single-tier, centralized orchestration, such as exclusively employed on the cloud. On the contrary, the challenges in computing continuum, discussed in Section VI, push towards distributing orchestration along the tiers, addressing management from a holistic perspective [38].

Indeed, the complexity and scale of the computing continuum challenge current methodologies for managing distributed Internet-based systems. Unlike Cloud systems, the continuum cannot be considered as *elastic* [98], rendering online Cloud-Fog-Edge computing as an outdated management methodology. General and adaptive modelling of the Computing Continuum Management (CCM) is thus in a key role in building novel solutions.

In particular, each stakeholder in the computing continuum sets its own objectives for the operation of applications. However, it remains an open question how these objectives and their fulfillment can be measured. Furthermore, the computing continuum comprises a high number of resources on many different computing tiers and layers of abstraction. Maintaining the objectives while keeping the system in balance is also an open question.

In more detail, Morichetta et al. [99] recognize for example the following particular research topics in relation to CCM:

Flexible representation. To orchestrate the resources in the computing continuum, CCM needs a flexible and adaptive representation of those resources. Since the architecture of the system may change, with new resources appearing and disappearing opportunistically, the representation must be able to reflect these changes.

Operational link to an underlying infrastructure. Any application running in the computing continuum is highly dependant on the underlying resources. Since those resources may be heterogeneous and dynamic, comprising for example a wide variety of different IoT devices as well as Edge, Fog, and Cloud configurations, any CCM methodology has to consider the infrastructure as a key component.

Temporal evolution. The computing continuum is constantly changing and evolving. Any management model must consider this change, and allow for concept and data drift. Moreover, the rate and direction of this change (say, in terms of the orchestration objectives related to costs, quality and resource usage) may also be considered to allow for appropriate reporting and consequent action.

Causality relations. The computing continuum comprises an ecosystem of multiple interacting resources and stakeholders. A global perspective thus has to consider the whole of this ecosystem, not restricting to individual resources or their activities. To understand how actions propagate across this ecosystem, CCM needs to keep track of causal relationships between the resources.

Proactive adaptation. The complex causal relations between the resources may lead to a cascade of failures as issues can propagate across the computing continuum. Maintaining stakeholder objectives thus requires prompt and proactive action to prevent such failure propagation.

Emergence. Causality and complexity, when not properly managed, can together cause harmful emergence and interference between system components. These issues are discussed in more detail in the following subsection Section V-B.

Learning framework. The ecosystem complexity and scale make it impossible to draw a complete management plan in the design phase. Therefore, setting management methodologies inside a learning framework is required to provide incrementally better solutions and adaptations.

B. Weakly coupled, autonomous control

A novel orchestration paradigm must strike a balance between local autonomy and centralized control [100]. Indeed, local autonomy brings all the benefits discussed above but it is not sufficient alone. This is because achieving system goals is a known challenge when agents are completely autonomous or even if they cooperate locally with each other. The main reason is emergence: due to nonlinear interactions, the system level behavior cannot be predicted from the behavior of the individual agents. Autonomous continuum agents might not get the resources they need, or the agents might interfere with each other. The goals might not be reached at all, or even if reached, resource usage might be too high [101].

Hence, some centralized control is needed to reach system goals with sufficient performance. In other words, loose (weak) coupling is required: the agents are nearly autonomous, but fair resource allocation and agent cooperation are ensured via minimal centralized control. Such minimal centralized control enables achieving system goals that require compromises from the agents. Furthermore, centralized control allows global optimization: balancing resource usage so that the goals are reached and performance requirements fulfilled [102].

Loose coupling leads to a hierarchical system where the higher levels control the lower ones, operating at lower resolutions and having wider perspective to the system and the environment [100]. The highest levels can be located in the cloud. The higher levels can use Multi-Objective Optimization (MOO) to realize fair resource allocation for the lower-level agents, i.e., a Pareto-optimal allocation. This optimization can be learnt over time and updated/re-learnt when changes in the system or its environment invalidate what has been learnt. MOO can help to reach goals while fulfilling constraints on resource usage. Loose coupling is a favorable property also for interaction between agents at the same hierarchy level – the interaction should be minimized.

Loose coupling introduces the same benefits for both applications and edge orchestration. Multi-agent systems are a natural choice for loose coupling. The centralized control can be realized through goals and constraints (specifically on resource usage). The agents have otherwise local autonomy but they advance the goals given by the higher level and do not

violate the given constraints. The higher levels monitor the progress and update the goal and constraints when needed, based on the optimization they perform.

From the perspective of continuum orchestration, the open questions in loose coupling are related to the implementation of orchestration functionality with autonomous agents. The general question is: How can the targeted system behavior be achieved with a set of independent, loosely-coupled agents? More specifically, the following topics need to be studied, among others:

Degree of centralization. Due to its benefits, local decision-making should be favored and centralized control used only when necessary. What is the optimal balance between independent local decision-making and centralized control in a large system of systems? How does this balance vary over time, for example, when abrupt changes are experienced in the operation environment?

Emergence. In a distributed, loosely coupled system, patterns of activity between the agents may emerge. How can harmful emergence be avoided in such systems? How can emergence be used in achieving the system-level goals?

C. Semantic communication in the computing continuum

Current communication paradigm employs information theory to quantify the maximum data rate that can be supported by a communication channel. This theory was developed in the 1940s by Claude Shannon [103], and it has been guiding the design of the information systems up until 5G and strikingly continues to be so even in the current mushrooming 6G visions.

However, this traditional focus has completely disregarded the semantic aspects of communication, viewing the meaning of the messages as largely irrelevant to communication [104]. In particular, as distributed intelligent agents interact, new notions of information are required to describe the common context between the agents and the potential of the agents to learn to efficiently communicate [105] changes in that context.

One possibility is to base this notion of information on the von Neumann–Morgenstern theory of sequential games [106], where information refers to imperfect knowledge of state. In more detail, as distributed agents interact, a shared context emerges, resulting in the exchange of information states that are minimally sufficient for the accomplishment of common goals. This would reduce the amount of information in the classical sense that agents would need to transmit or receive. However, the prerequisites for such efficient joint behavior emerging are not known, and neither is the mathematical characterization of such networked systems.

Other fundamental problems related to semantic communication include the following topics, among others:

Emergence. How and under what conditions cooperative communication among agents emerges and is robust to deviations between agents having different priors/beliefs/knowledge/architectures?

Convergence. Under which conditions will agents converge to a shared language when learning from data?

Scarcity. How to deal with the fact that knowledge will be limited per agent in terms of incomplete information, limited compute power, memory for training, inference, reasoning, planning).

D. Security and Privacy

Edge AI, that is, distributed AI methods deployed in the computing continuum, can play a vital role in enhancing the security and privacy of future orchestration in the computing continuum. Added intelligence in the continuum allows the deployment of advanced and smart security mechanisms already at the edge of the network, closer to the external users and devices. Thus, overall security can be improved as it is possible to detect and mitigate some attacks via external user devices already at the edge of the network [107]. The early detection of some attacks will be essential to eliminate the impact as well as the propagation of such attacks to the critical core network elements.

Moreover, edge AI based methods are processed in local vicinity of the user instead of in the remote cloud, reducing the need of transferring raw user data between the user and the Cloud via untrusted backhaul networks such as the Internet [108]. Moreover, local processing eliminates the possibility of various security attacks such as Sybil, Denial of Service (DoS)/Distributed Denial of Service (DDoS), fiber tapping, hidden pulse attacks, jamming attacks, as well as privacy leakages associated with a long-distance data transmission phase [33].

The heterogeneous ecosystem of IoT is suffering from various security issues due to the lack of device processing capabilities, improper security standardization, energy-saving strategies, lack of expertise, and untrusted device manufacturers [109]. Thus, the implementation of advanced security mechanism at the network-level as Security-as-a-Service (SECaaS) [110, 111] is one of the viable solutions to address the above limitations. Edge AI methods can implement a more advanced and intelligent SECaaS mechanism for IoT devices [112]. Added intelligence along the computing continuum allows deploying advanced security mechanisms such as threat intelligence tools, intelligent deep packet inspectors, and firewalls [113]. For example, for limiting the effects of DDoS attacks, edge AI methods can provide a network perimeter around cloud resources. This perimiter could then intelligently adapt the packet processing rate at the router to stop the cloud from being overwhelmed with DDoS requests, time-out half-open connections, and equip the router with an intelligent filter which may drop packets which are likely to be used for the attack.

Furthermore, edge AI can be used to implement AI algorithms in a decentralized manner. Thus, some attacks on edge AI systems can be restricted to the local vicinity, not impacting other edge devices. In this way, edge AI can also eliminate the possibility of a single point of failure associated with centralized AI algorithms [114].

Finally, managing and processing user data locally increases [115] or even guarantees user privacy, given the

applied distributed learning and decision making processes employ privacy-guaranteeing approaches such as differential privacy [116].

Some particular research topics related to security and privacy are listed below:

Physical security. One of main challenges is the physical security of the AI components. AI integration with edge computing will promote edge AI resources to a critical role in network management and orchestration [117, 118]. However, deployments in the computing continuum typically reside outside the central data infrastructure, in physically insecure premises in the wild. The computing continuum should thus have extra precautions in place to mitigate the impact of physically tampering with devices, adding malware to edge devices, swapping or interchanging devices, or injecting rogue data sets and AI algorithms. Additional security measures such as hardware root of trust, crypto-based ID, encryption for in-flight and automated patching should be deployed to obtain tamper-proof edge AI devices and services. As an example, Sachdev et al. [119] discuss the critical security and privacy issues for Edge AI in IoT/IoE digital marketing environments along with some possible mitigation mechanisms.

In addition, the impact of capturing edge devices with AI is much more severe than capturing an edge device without AI due to added intelligence and distributed control functionalities. If an attacker takes control of an edge AI device, he might be able to jeopardize or take control of almost all the localized network services of a particular local vicinity. Moreover, decentralization of Edge AI opens up unknown security issues such as synchronization attacks, information protection issues, membership inference, and API-based attacks [114].

Trust. As various stakeholders can connect their resources to the computing continuum [120], establishing trust between the stakeholders can prove challenging. Ensuring trust in multi-domain orchestration is thus an important factor for beyond 5G networks.

Ensuring the trust in the Edge AI system is challenging due to its distributed nature and the potential liability issues [115]. Strategies such as open implementation, well-defined specifications, and reputation systems can be used to address these trust issues [121]. In addition, it is necessary to define and differentiate the control role between devices by the user and resources provided by providers (e.g., AWS, GCP, Azure). One benefit of edge AI is that the user data required to perform the AI processing will keep local at the edge servers, leading to improved privacy. This can be achieved if edge servers have higher privacy and trust levels than the cloud server. Since the edge by itself does not guarantee privacy, this might raise new challenges to ensure the privacy of user data [115].

There are also challenges regarding the confidentiality and integrity of machine learning. To prevent the leakage of personally identifiable information, the training data needs to be anonymized or protected with cryptographic techniques. In addition, the final machine learning model needs to be protected against model inversion and membership inference attacks. Anynomization techniques, such as differential privacy, guarantee that personal information does not leak from the training data or from the final model.

However, such methods can lead to a reduction in classification performance, as well as a lower convergence rate [122]. They also have limitations when applied on unstructured data, such as images [123]. Cryptographic techniques do not have such limitations, but incur a high computational or communication cost. Research is needed to determine the correct tradeoff between privacy protection and the performance of the system.

AI integrity. Integration of AI/ML techniques will lead to the new set of AI/ML related attacks such as data injection, data manipulation, logic corruption, poisoning, evasion attacks [124, 125], which Edge AI systems need to mitigate, calling for novel defence methods [126].

For example, in order to train accurate models, the training data needs to be accurate. Poisoning attacks influence the training data resulting in generalization errors in the final classifier. While there are methods that attempt to detect and exclude poisoned samples for simple models, such as linear regression [127], the poisoning problem is still largely unsolved [128]. In particular, for FL, poisoning attacks can lead to a substantial decrease in classification accuracy and recall even with a small number of poisoned model updates [129]. Novel methods are needed to detect poisoned samples and local model updates before the global model computation, as well as to prevent misclassification through evasion attacks.

Another integrity issue arises from the model training process. In edge computing, training is potentially distributed or executed on an untrusted entity. Malicious parties need to be prevented from inserting false training data or a backdoor into the final model. Homomorphic encryption can be applied to train the model on encrypted data and verifiable computing to attest its correctness, but both require significant computational resources from the computing party or the edge devices. An alternative approach is secure multiparty computing. However, due to its interactivity it incurs a significant communication cost and typically does not scale well with the number of participants [130]. The tradeoff between computing and communication costs needs to be carefully considered especially for use cases where both computing power and bandwidth is limited.

E. Synthesis

Synthesizing the above research topics, we can provide an early vision of decentralized, multi-domain, multi-tenant orchestration in the computing continuum. In our vision, resources in the continuum are modelled as agents, and each agent is trying to fulfill externally set objectives on cost, quality and resource usage. While trying to reach those objectives, the agents make decisions on when and how to conduct actions related to orchestration functions (see Section III-C). Moreover, to conduct those actions, the agents may need to negotiate with other agents to provision resources. As an example, a task agent may need to make a decision on offloading computations to cloud from an edge node, and

needs to negotiate the terms of the offloading with the cloud agent to understand the consequences of that decision.

The objectives are set on the agents according to their place in a hierarchy. Each stakeholder, be that cloud or network service provider, or an app, is represented by a stakeholder agent on the highest level of the hierarchy. These agents have service level objectives to fulfill, related to cost, quality, and resource usage, set by the application developer or domain manager. The stakeholder agents break down and pass on their objectives to the agents below them in their administrative domain, organized as clusters and further sub-clusters (Fig. 5).

The distance measure for forming these clusters may vary based on the environment. For example, forming clusters based on proximity minimizes latency and maximises connectivity. Furthermore, the clusters need to be dynamic to handle the opportunistic nature of the continuum. As an example, the highest level cluster agents could be cloud and edge cluster agents, with edge sub-clusters comprising individual locations of interest, with new agents appearing and disappearing as new devices move through that location.

Meanwhile, agents negotiate for resource usage within as well as across administrative boundaries. Within administrative boundaries, the negotiation can be considered co-operative, while over the boundaries negotiation may be competitive. While negotiating, the resource agents need to consider their own objectives, broken down from those of their stakeholder agent and passed to them along the control hierarchy (Fig. 6).

This framework follows the principle of *weak coupling*, with a hierarchy of agents where each agent makes autonomous decisions on how to reach its objectives. Over administrative boundaries, the agents may need to *learn to communicate* for their negotiation. *Security and privacy* must be considered

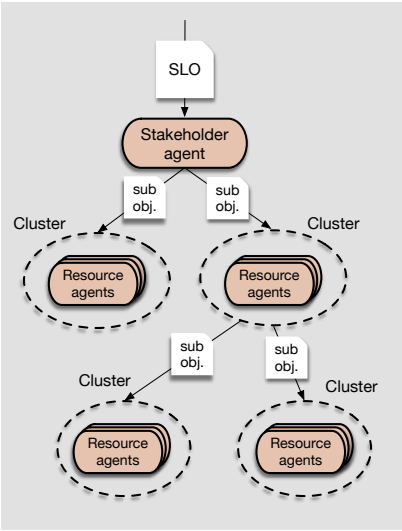

Fig. 5: An example of the objective overlay of decentralized orchestration. The application stakeholder agent pursues its objectives, related to cost, quality and resource usage, breaking them down for sub-objectives for high-level clusters. These cluster, in turn, oversee the agents in their sub-clusters.

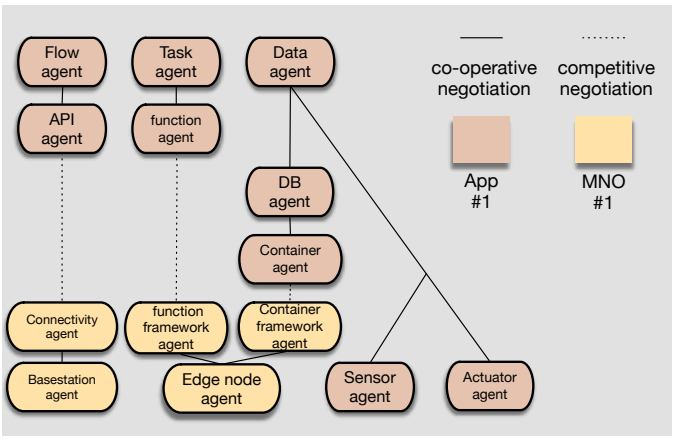

Fig. 6: An example of the negotiation overlay of decentralized orchestration (see App #1 and MNO #1 in Fig. 3). Resource agents negotiate both within (solid lines) as well as over administrative boundaries (dashed lines) for the use of resources.

from the very beginning.

However, the computing continuum poses a number of challenges in realizing the vision of autonomous, intelligent AI agents. Communication is intermittent and fluctuating, computation resources are distributed, heterogeneous and opportunistic, and data is distributed, siloed and non-IID, and at times sensitive. Finally, there can be a massive number of resources, applications, tenants and other stakeholders over a number of domains, and these stakeholders may have partially conflicting objectives.

These challenges set requirements for the orchestration of the computing continuum, as depicted in Fig. 7. Orchestration must be decentralized, as the resources are; further, weak coupling [102] and local autonomy allow the approaches to survive alone if connections are severed. Non-IID data requires distributed edge intelligence, with localized learning and decision-making, while the numerous stakeholders and tenants present in the continuum demand approaches that support balancing multiple objectives. Finally, security and privacy must be considered for both APIs and execution as well as data and AI models.

However, implementing AI methods to answer these requirements is challenging. In particular, the following topics must be considered:

Common context and standards. While AI agents may learn to communicate and negotiate, they need some common context and standards to ensure interoperability and security of transactions. For example, distributed learning of models requires that AI agents share information on their models with each other. What is the minimal common context (e.g., protocol for sharing model information, or means of describing model metadata such as structure or purpose) required for this sharing of model information? How is resource discovery implemented? Can learning be fully distributed, or is a central authority required to fulfill some function?

Moreover, for multi-agent and distributed decision making,

an agent may need to know what the other agents observe, how they act, and how in general they make decisions. The extent to which this is possible or reasonable may vary due to, for example, the availability of computational capacity, data, or communication links. Is it possible to define a minimal set of resources required for decision making? Can the methods be scaled based on available resources?

Finally, what is the minimum amount of common knowledge agents can be assumed to have at the beginning of negotiation? What type of common ground can be instantly established for negotiation within and across administrative boundaries?

Convergence. The computing continuum is a challenging environment for conducting distributed learning. Is it always possible to guarantee convergence in model learning, or do the agents have to accept that in some cases, learning does not result in a usable model? What consequences does this have for the agent's subsequent decision-making? Moreover, what are the conditions under which one can guarantee a convergence into a stable and optimal decision making strategy, considering also the possible adaptation of the strategy when the underlying environment changes?

Lack of training data. Data in the computing continuum is often geographically distributed and locked behind siloes. Furthermore, online training of a decision making strategy from scratch may be unfeasible. The algorithms are data hungry, and can make very awful decisions particularly during initial learning stages. Building training data sets in such an environment may be challenging, calling for pre-training, or for methods that use unlabeled data, or learn quickly from few samples. What is the best approach in each case?

Adaptivity. The computing continuum is in a constant state of flux, with nodes appearing and disappearing, connectivity fluctuating, users moving, and application components migrating between nodes. AI methods need to be able to scale their resource usage to adapt to this change. However, on the other hand, AI methods must strike a balance between stability and adaptation, not reacting to temporary changes. What is required to identify the balance point here?

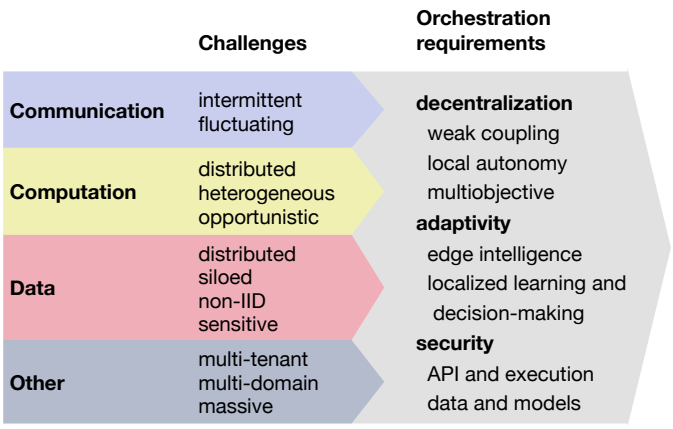

Fig. 7: Challenges inherent in the computing continuum, and subsequent characteristics required of the AI approaches

VI. Survey of Edge AI Approaches

This section presents promising AI approaches which address some of the challenges described in Section V and may lead into solutions that consider all of them. In particular, we look at distributed and secure methods for ML, decision making, and negotiation, and finally consider shortly certain other emerging approaches. We will be focusing more on introducing research fields that we believe will be the key areas of the research for the future computing continuum orchestration rather than on providing an in-depth analysis of the methods inside the fields due to the enormous amount of work each field entails.

A. Distributed Learning

Orchestration in the computing continuum can be modelled as a hierarchical network of intelligent, autonomous agents that manages the resources of the platform in a decentralized manner. These agents need ML models to make predictions about processes and future states, which supports the agents' decision making processes. Based on data, the agents must learn different dynamics in their environment to improve their performance and to adapt to the uncertain, evolving environment.

Each agent has access to the data they have collected, but this data may not have enough volume or diversity to train accurate models. In addition, an edge agent may not have enough resources for the training and inference of complex models. Hence, it is inevitable that agents must somehow collaborate with other nearby agents in the training and inference of ML models.

Below, we introduce methods that, with further development, can be key enablers for achieving efficient distributed and decentralized ML model training and inference among a network of AI nodes, addressing some of the challenges described in Section V-E. The focus is on supervised learning methods.

1) Federated Learning: Training ML models on edge requires distributed learning architectures and algorithms. FL has quickly become the de facto training paradigm for distributed model training in edge environment. FL, first introduced by Google [131], aims to train a global ML model in a distributed manner. The global model is most typically an ANN model, but it can also be some other parameterized model. Original version of FL, often called *vanilla FL*, trains a global model in a centralized manner on decentralized data. Each agent participating in the training has their own training data that they use to train a local model. Then, the local parameter updates are sent periodically to a central server that aggregates the updates and sends the resulting global model back to agents. The goal is to minimize the following objective function:

$$\min_{\theta} f(\theta), \text{ where } f(\theta) = \sum_{k=1}^{K} p_k F_k(\theta). \tag{1}$$

Here, $\theta \in \mathbb{R}^d$ are the d parameters of the model, K is the total number of agents, $p_k \geq 0$ is the weight for agent k, and $\sum_k p_k = 1$. Usually $p_k = \frac{n_k}{n}$, where n is the total number of samples, n_k is the number of samples at agent k. F_k is the local objective function of the agent k, defined as $F_k(\theta) = \frac{1}{n_k} \sum_{i=1}^{n_k} f_i(\theta)$, where f_i is the local loss on sample i.

In *Federated Averaging* (FedAvg), the baseline FL algorithm introduced by McMahan et al. [131], agents exchange model parameters with a central server that aggregates them into a global model update with $\theta_{t+1} \leftarrow \sum_{k=1}^{K} p_k \theta_{t+1}^k$. The local parameters at agent k, i.e., θ_{t+1}^k, have been calculated by applying Stochastic Gradient Descent (SGD) locally a specified number of times after the previous global weight update θ_t. FedAvg can also be implemented based on exchanging gradients instead of the parameters, in which case the central aggregator calculates the mean of the local gradients that is subsequently used by the agents to update their parameters locally [14].

Vanilla FL approach has multiple issues and deficiencies, which renders it incompatible with our vision of an autonomous edge. Starting from communication efficiency, wireless links with low bandwidth and intermittent connectivity are an essential part of the edge environment, while communication is a major bottleneck in FL. The communication costs of vanilla FL can be partly controlled with the update period that determines how often local parameters are sent to the central aggregator. For example, Wang et al. [132] propose a control algorithm for dynamically changing the update period so that global loss function is minimized under a fixed computational and communicational resource budget. However, exchanging model parameters (or gradients) can cause huge communication overhead, as modern ANN models can have millions of parameters. One common solution is to reduce the size of the transmitted update by model compression schemes, which include, for example, model sparsification or low-rank approximations, quantization of parameters, lossy compression, and Golomb lossless encoding [133].

Agents participating in training are usually heterogeneous in terms of hardware, as well as resources such as energy or network connectivity. This can cause stragglers in the learning; agents may take too long to respond or completely drop out. As a consequence, some agents can have outdated models. Solutions to the straggler issue include asynchronous communication (which often relies on a bounded-delay assumption that may be unrealistic), actively selecting the participants to the training based on system resources, and increasing the fault tolerance of training to drop outs through algorithmic redundancy with methods such as coded computation [133]. Another solution is to offload the tasks and data to a neighbouring node even though this requires considerations of privacy and affects the learning latency, as well as the accuracy of the model [15]. On the other hand, if the dynamics of resource availability are modeled as a time series, Gaussian Process Regression (GPR) can be used to predict future resources, which will allow identifying agents that are more likely to be stragglers in advance [15]. Model parameter dynamics can also be predicted with GPR, which allows agents to estimate the model parameters of others in order to continue local training under erratic connectivity.

Incentive mechanisms are important for building accurate models with honest participation. Agents with higher quality data should be incentivized to participate in the learning, and they should also be rewarded for their greater contribution. In competitive settings, in particular, each agent gaining the same global model regardless of their contribution to the training is not desirable. The incentive can be, for example, a final model with a different level of performance, that is, the quality of the model is proportional to the level of contribution [134]. Additionally, blockchain technology provides many secure incentive mechanisms [135].

When the goal is to train a global consensus model, non-IID data across the agents causes significant problems because distributed training algorithms usually assume that the data is IID, and convergence guarantees are given with the assumption of IID [133]. Data augmentation is one technique for mitigating the non-IID data problem. In its simplest form, data augmentation can be the sharing of data between the agents. For example, in the experimental study by Zhao et al., they were able to increase the model accuracy significantly by sharing only 5% of the agents' local data [136]. However, such sharing of data compromises privacy and communication efficiency, as well as undermines one of the key ideas of distributed learning, which is to keep the training data local. In an approach termed *federated augmentation*, a server trains and shares a generative model that can be used by the agents to augment their data locally [137].

To ensure the convergence under non-IID data and varying system resources, Li et al. propose an algorithm called FedProx [138]. It introduces two modifications to FedAvg in order to achieve this. First, it adds so called proximal term to the local objective function so that each agent k calculates the local model update θ_{t+1}^k by solving $\theta_{t+1}^k = \arg\min_\theta h_k(\theta; \theta_t)$, where $h_k(\theta; \theta_t) := F_k(\theta) + \frac{\mu}{2}\|\theta - \theta_t\|^2$. The proximal term addresses non-IID data by restricting the local model updates to be closer to the received global model at the beginning of iteration t. Then, it introduces a notion of γ_k^t-inexactness, that is, θ^* is a γ_k^t-inexact solution of $\min_\theta h_k(\theta; \theta_t)$ if $\|\nabla h_k(\theta^*; \theta_t)\| \leq \gamma_k^t \|\nabla h_k(\theta_t; \theta_t)\|$, $\gamma_k^t \in [0, 1]$ $\forall k, t$. Parameter γ_k^t measures how much local computation is performed to solve the local problem $\min_\theta h_k(\theta; \theta_t)$ on agent k at the iteration t. The smaller the value of the parameter, the higher the accuracy of the solution, meaning also that more local computation is required.

Note that the value of γ_k^t is implicit, because it is not recovered. As in FedAvg, a given number of local iterations is specified to be carried out before global aggregation, but if an agent has not had enough resources to calculate all the iterations, it will send its partial solution for global aggregation nevertheless. The global update is then calculated with $\theta_{t+1} \leftarrow \frac{1}{K} \sum_{k=1}^{K} \theta_{t+1}^k$. In other words, introducing the notion

of γ_k^t-inexactness allows to incorporate also partial work (from stragglers) in the global update, because it provides a way to theoretically analyze the convergence rate under varying system resources. Li et al. show through their experiments with five different non-IID data sets that FedProx significantly improves the convergence rate when compared to FedAvg even when 90% of the agents are stragglers.

How to train an accurate and reliable global model with low latency is a crucial question, especially in distributed learning on edge, where the training procedure encounters issues of non-IID data, energy and memory limitations, as well as low communication bandwidth. Park et al. provide an in-depth overview of different theoretical and technical enablers for low latency federated training and accurate inference under on-device and communication constraints [14]. They focus particularly on reliability guarantees, latency reduction and scalability enhancement in distributed model training. They present, among other things, suitable mathematical tools for quantifying generalization error bound of a federated model, e.g., with meta distribution, and for examining whether training loss reaches a target loss level, e.g., with extreme value theory.

2) Federated Distillation: Traditional distributed learning algorithms exchange model parameters. An approach termed as *federated distillation* (FD) exchanges model outputs instead of model parameters, which reduces communication costs [137]. In FD, each agent keeps track of the local average logits per label, sending them periodically to a central server. The central server calculates the global average logits per label, which are then downloaded by the agents. Finally, agents use the global average logits in a distillation regularizer that penalizes larger differences between sample's local logits and the global averaged logits of its true label. Thus, each agent updates its local parameters as follows:

$$\theta \leftarrow \theta - \mu \nabla \{ l(F(\theta, x_i), y_i) + \lambda \cdot l(F(\theta, x_i), \hat{F}^k_{t,y_i}) \}. \quad (2)$$

Here, μ is the learning rate, $l(\cdot)$ is the loss function, $F(\cdot)$ is the prediction function that gives the normalized logit vector for sample i, x_i are the features and y_i is the ground truth label of sample i. λ is the weight for the distillation regularizer and $\hat{F}^k_{t,y_i}$ is the global average logits for label y_i received by agent k at the beginning of iteration t. Loss function $l(\cdot)$ is usually either cross entropy or mean squared error [137, 139].

The communication payload size of FD is proportional to the output dimension rather than the model size, which reduces communication costs substantially. FD also allows each agent to have a different model, e.g., a agent with less resources can have a simpler neural network. However, FD is more vulnerable to non-IID data distribution across agents and produces less accurate models when compared to FL [139].

The communication-accuracy trade-off in FD, that is, even though the communication costs are substantially reduced, the accuracy is usually worse when exchanging model outputs instead of parameters, can be exploited in mobile communication systems, where uplink data rates are usually much lower than downlink data rates. The idea is to utilize FL in downlink (agents download model parameters) and FD in uplink communication (agents upload model outputs) [140]. The central server must do a model output-to-parameter conversion by utilizing knowledge distillation to update the global model it maintains. The server minimizes the difference between the uploaded outputs and the outputs of the global model, which naturally requires a set of samples. For this, agents also send a set of seed samples to the server, which have been encoded to preserve privacy.

3) Fog Learning: Hosseinalipour et al. introduce a paradigm called *Fog Learning* [141, 142], which shifts from the centralized star topology FL towards a semi-decentralized architecture. They develop a multi-layer cluster-based learning architecture especially suitable for collaborative model training across the computing continuum. The nodes form a hierarchy, where the nodes holding the data and training the local models are at the bottom, the main server for global aggregation is at the top, and the model parameter aggregations and the parameter updates travel across multiple layers. The nodes in each layer form clusters, and inside each cluster collaboration can be enabled with device-to-device (D2D) communication. Nodes on the same level may dynamically change the clusters they belong to during the model training.

For solving the minimization problem in Eq. (1) inside the hierarchical, cluster-based network, Hosseinalipour et al. develop an algorithm called *multi-stage hybrid federated learning* (MH-FL) [142]. As the main server is only interested in the weighted average of the local parameters, MH-FL is based on local aggregations at each network layer, which can happen either through distributed aggregation or instant aggregation. Distributed aggregation is for D2D enabled clusters, where all nodes can participate in a consensus scheme that ensures all devices inside the cluster agree on the average of their model parameters. The parent node of the cluster needs to then only fetch parameters from one node inside the cluster, and scale the obtained parameters by the number of nodes in the cluster to approximate the sum of the nodes' parameters. Instant aggregation is used in the case where D2D communication is not enabled inside a cluster. It is similar to the conventional aggregation of FL, that is, the parent node collects and aggregates the parameters from the nodes in the cluster.

The learning accuracy of MH-FL is able to approach that of the centralized gradient descent [142]. The experiments also show that MH-FL can result in 50% device energy savings on average and 80% reduction in the number of parameters transferred over the network layers compared with conventional FL.

4) Peer-to-peer Learning: FL relies on central aggregator, which creates a single point of failure, compromises scalability and maintains higher communication costs due to the star topology of the network. Our vision of an autonomous edge requires less reliance on centralized entities and more independent and decentralized ways to coordinate model training.

In decentralized learning, agents are organized as a virtual

and possibly dynamic network graph with a small maximum node degree, and the nodes can only share their local model parameters in their neighbourhood. These local models should gradually converge to a global model, i.e., agents should reach *consensus* [143, 144]. This is usually formulated as a global consensus problem [145]:

$$\text{minimize} \quad \sum_{k=1}^{K} f_k(\theta_k) \tag{3}$$

$$\text{subject to} \quad \theta_k = \Theta \; \forall k. \tag{4}$$

Here, $f_k(\cdot)$ is the local objective function and θ_k are the local model parameters of agent k, and Θ are the global model parameters. The minimization problem in Eq. (3) differs from the minimization problem in FL (Eq. (1)) in the way that the additive objectives depending on a shared global variable in Eq. (1) have been turned into separable objectives depending on local variables. The formulation in Eq. (3) can be easily split across computing nodes, while the constraint in Eq. (4) ensures that the local variables are equal. This allows us to split the global model optimization into K separate problems that can be solved in parallel.

The problem in Eq. (3) can be solved with a primal-dual method called Alternating Direction Method of Multipliers (ADMM) [145]. However, ADMM relies on a central parameter server for handling the constraint in Eq. (4), meaning that updating Θ requires collecting all θ_k from the agents. Elgabli et al. [146] propose a method called Group ADMM (GADMM) that is a communication-efficient decentralized algorithm for solving the problem in Eq. (3) under the constraint that $\theta_k = \theta_{k+1} \; \forall k \in \{1, \cdots, K-1\}$. This constraint implies that each agent only has joint constraints with two neighbours, except for the 'end' agents that have only one. GADMM divides the agents into head and tail groups, where each agent in the head group communicates only with two agents in the tail group and vice versa (i.e., the overlay topology is a logical chain). Consequently, at each communication round, only half of the agents are competing for the limited communication bandwidth.

GADMM allows all the agents in the same group to update their parameters in parallel, while the parameters of agents belonging to different groups are updated in an alternating manner. If we denote the head group as $\mathcal{N}_h = \{\theta_1, \theta_3, \cdots, \theta_{K-1}\}$ and the tail group as $\mathcal{N}_t = \{\theta_2, \theta_4, \cdots, \theta_K\}$, the primal variables (parameters) and the dual variables are updated as follows. At iteration $t+1$, each head agent first updates their parameters as

$$\theta_{k\in\mathcal{N}_h}^{t+1} = \arg\min_{\theta_k}[f_k(\theta_k) + \langle\lambda_{k-1}^t, \theta_{k-1}^t - \theta_k\rangle + \langle\lambda_k^t, \theta_k - \theta_{k+1}^t\rangle + \frac{\rho}{2}\|\theta_{k-1}^t - \theta_k\|^2 + \frac{\rho}{2}\|\theta_k - \theta_{k+1}^t\|^2]. \tag{5}$$

Each head agent sends its update to its two tail neighbours, after which each tail agent updates their parameters as

$$\theta_{k\in\mathcal{N}_t}^{t+1} = \arg\min_{\theta_k}[f_k(\theta_k) + \langle\lambda_{k-1}^t, \theta_{k-1}^{t+1} - \theta_k\rangle + \langle\lambda_k^t, \theta_k - \theta_{k+1}^{t+1}\rangle + \frac{\rho}{2}\|\theta_{k-1}^{t+1} - \theta_k\|^2 + \frac{\rho}{2}\|\theta_k - \theta_{k+1}^{t+1}\|^2]. \tag{6}$$

Each tail agent sends its update to its two head neighbours, after which every agent updates locally their dual variables λ_{k-1} and λ_k as

$$\lambda_k^{t+1} = \lambda_k^t + \rho(\theta_k^{t+1} - \theta_{k+1}^{t+1}). \tag{7}$$

Elgabli et al. [146] also extend GADMM to Dynamic GADMM (D-GADMM), where the set of neighbours to each agent varies over time, meaning that the overlay topology is still a logical chain, but the physical neighbours can change. Hence, in D-GADMM, all the agents periodically check their connections, and if some sort of change is detected, they broadcast their parameters to the new neighbours. After this they find their new logical neighbours in the chain and send the right dual variable λ_k^t to their right neighbour to ensure that they share the same dual variable. D-GADMM can also be used to change the logical chain topology even when the physical topology itself does not change, which improves the convergence speed of GADMM.

Besides primal-dual methods, also primal methods have been proposed for solving the problem in Eq. 3. These methods often utilize Distributed Stochastic Gradient Descent (DSGD), and there exist different approaches based on how the model parameters are exchanged and aggregated between the peers. As a side note, the optimization problem in Eq. 3 is usually formulated differently for primal methods. The subindex k is dropped from the parameter vector θ, and hence the constraint in Eq. 4 is not needed. In other words, the problem is turned into an (equivalent) unconstrained global minimization problem of the form $\min_{\theta\in\mathbb{R}^d} \sum_{k=1}^{K} f_k(\theta)$ that depends on a shared global variable.

In primal approaches that utilize gossip protocol, agents select one or a few of their neighbours for model parameter exchange in order to propagate model updates across the network. The choice of neighbours can be done randomly or in a more optimized manner with the utilization of virtual topologies. However, purely random gossip algorithms usually lead to convergence issues in large scale systems [9]. To this end, Daily et al. propose GossipGraD [147], an algorithm that scales DSGD on large scale systems by combining asynchronous communication with gossip partner selection and partner rotation to ensure faster propagation of model updates and faster convergence. GossipGraD was developed particularly for training DL models on distributed memory systems, which consist of a set of processing nodes interconnected by a high-speed network. Furthermore, as noted by Savazzi et al. [148], early approaches on gossip based methods, such as GossipGraD, cannot be fully used in D2D networks because they ignore medium access control and half-duplex constraints.

Hu et al. [149] propose a segmented gossip approach that is supposed to be more suitable for edge environment with wireless, low bandwidth connections. The basic idea of segmented gossip is to use so called segmented pulling in the local model aggregation phase, which means that each agent pulls different parts of the model parameters from different agents, and then rebuilds a mixed model for aggregation. In other words, rather than pulling all model parameters from a single peer, segmented gossip pulls parameters from a different peer for each parameter segment, which means that the total transmission traffic is now distributed among several links rather than just one. For the aggregation itself, each agent needs to rebuild a certain number of mixed models in order to propagate updates across the network faster and ensure the model quality. That is, if the parameters are divided into S segments and the required number of mixed models is R, then the agent needs to collect $S \times R$ segments from its peers. To get the final aggregation results, the received parameters are averaged segment-wise with the agent's own local parameters, after which the agent can continue its local training until the next aggregation phase.

Diffusion approaches are another type of primal methods for solving the optimization problem in a decentralized manner. There are two main strategies for diffusion: Combine-Then-Adapt (CTA) and Adapt-Then-Combine (ATC) [150, pp. 456-469]. In CTA diffusion, for every iteration t, each agent k first combines the existing parameters of its neighbourhood from the previous iteration, obtaining an intermediate result (*combination step*). Then the agent k uses this intermediate result to perform a gradient descent update (*adaptation step*). These steps are shown in Eq. 8, where $\mathcal{N}_k$ is the neighbourhood of the agent k (including the agent k itself), $\{a_{l,k}\}$ is a set of nonnegative coefficients that satisfy $\sum_{k=1}^{K} a_{l,k} = 1$, $a_{l,k} = 0$ if $l \notin \mathcal{N}_k$, $l = 1, 2, \ldots, K$, μ_k is a small constant step size parameter of agent k, and $\nabla_\theta f_k$ is the gradient vector evaluated at the intermediate result ψ_k^{t-1}.

$$\begin{cases} \psi_k^{t-1} = \sum\limits_{l \in \mathcal{N}_k} a_{l,k}\, \theta_l^{t-1} \\ \theta_k^t = \psi_k^{t-1} - \mu_k \nabla_\theta f_k(\psi_k^{t-1}) \end{cases} \tag{8}$$

In ATC diffusion, the combination and adaptation steps are reversed. Now for every iteration t, each agent k first uses the parameter estimate from the previous iteration to perform a gradient descent update, obtaining an intermediate result. Then the agent k combines all these intermediate results from its neighbourhood. The ATC procedure is shown in Eq. 9.

$$\begin{cases} \psi_k^t = \theta_k^{t-1} - \mu_k \nabla_\theta f_k(\theta_k^{t-1}) \\ \theta_k^t = \sum\limits_{l \in \mathcal{N}_k} a_{l,k}\, \psi_l^t \end{cases} \tag{9}$$

CTA diffusion propagates the information across the whole network more thoroughly, because the information is diffused by aggregating the neighborhood updates from the previous iteration, as well as by evaluating the gradient vector at the aggregate value. ATC diffusion also includes the information from the neighbourhood's data, because the combination step incorporates the gradient vectors of the neighbors, with each gradient evaluated at the respective update θ_l^{t-1} from the previous iteration.

More general diffusion approaches exchange gradients in addition to model parameters. The purpose is to increase the information flow and speed up the convergence by utilizing also neighbours' local data, even though these approaches lead to increased communication costs [151]. The core idea of these more general diffusion strategies is to introduce a new local objective function $f_k^{loc}(\theta) \mathrel{\widehat{=}} \sum\limits_{l \in \mathcal{N}_k} c_{l,k} f_l(\theta)$ for each agent k, which is a weighted combination of the local objectives of the agent k's neighbourhood (including the agent k itself) [152]. Here, $\{c_{l,k}\}$ is a set of nonnegative coefficients that satisfy $\sum\limits_{k=1}^{K} c_{l,k} = 1$, $c_{l,k} = 0$ if $l \notin \mathcal{N}_k$, $l = 1, 2, \ldots, K$. With this new formulation, we can modify CTA in Eq. 8 by setting $\nabla_\theta f_k(\psi_k^{t-1}) \mathrel{\widehat{=}} \sum\limits_{l \in \mathcal{N}_k} c_{l,k} \nabla_\theta f_l(\psi_k^{t-1})$, and ATC in Eq. 9 by setting $\nabla_\theta f_k(\theta_k^{t-1}) \mathrel{\widehat{=}} \sum\limits_{l \in \mathcal{N}_k} c_{l,k} \nabla_\theta f_l(\theta_k^{t-1})$. Here, rather than taking the gradient descent step based only on the gradient of the agent k, we take the weighted average of the whole neighbourhood's gradients evaluated at the same point ψ_k^{t-1} (for CTA) or θ_k^{t-1} (for ATC) as the descent direction. This naturally increases the cooperation between the agents, as now each agent k can more efficiently utilize information from its neighbours' local data sets to update its estimate.

With the aim of reducing communication costs of gradient exchange, Savazzi et al. propose Consensus based Federated Averaging with Gradient Exchange (CFA-GE) [148], which combines elements from average consensus methods [153] and CTA diffusion. CFA-GE was especially developed for training ANN models in large scale networks with intermittent connectivity.

CFA-GE consists of four stages. At the beginning of each iteration t, each agent k receives the model parameters of its neighbours from the previous iteration and combines them with a consensus-based aggregation into an intermediate result ψ_k^t (includes the parameters of the agent k). Each agent k then sends this intermediate result back to its neighbours, each of whom calculates a gradient evaluated at ψ_k^t using the local data. After agent k has received all gradients from its neighbours, it calculates a weighted average of them and uses the average to update ψ_k^t, obtaining a second intermediate result $\tilde{\psi}_k^t$. Finally, each agent k does a number of SGD updates with its local mini-batches starting from $\tilde{\psi}_k^t$.

The CFA-GE algorithm described above requires many communication rounds, as well as synchronization because each agent k has to wait for the gradients from its neighbours. Hence, Savazzi et al. [148] also propose a two-stage asynchronous algorithm for implementing CFA-GE. This improved algorithm is achieved by removing the stages of sending the intermediate results ψ_k^t and then waiting for the neighbours' gradients. Now, these gradients are predicted using the past aggregated models from the neighbours, meaning that at the

beginning of each iteration t, each agent k now receives the aggregated models ψ_l^t and the gradient predictions from its neighbours l. Each agent k can use this information to finish the whole model update without communicating with other agents, and at the end of the iteration it will simply send its new aggregated model ψ_k^{t+1}, as well as the gradient prediction it made to its neighbours.

Utilizing distributed ledger technologies provides another way to decentralize the model training. Integrating FL and blockchain technology has the additional benefit of providing security and trust for the system, because all training events are recorded in a distributed ledger. Blockchain technology can address the threat posed by adversarial agents that try to tamper the training, a threat that is rarely addressed by the proposed FL methods in the literature [14]. Furthermore, blockchain can provide all the information needed to initialize the training process in the first genesis block [154]. This information includes the model structure, hyperparameters, and optimization algorithm.

Kim et al. propose *BlockFL* [155], a blockchain-based decentralized FL architecture where agents send their local updates to so called miners (edge nodes with more resources) that share and cross-verify all the local updates before starting a Proof-of-Work (PoW). The miner who first finishes PoW receives a mining reward and creates a new block that is added to the blockchain. Each agent updates the local model from the most recent block, i.e., the global update is computed locally. While blockchain offers security, traceability and reliability, the substantial workload of the miners, comprising model verification, sharing and PoW, requires substantial computational, communicational and energy resources, as well as increases the latency of the training [135]. In addition, there seems to be a lack of efficient cross-verification methods for the local model updates [156].

Shayan et al. propose Biscotti [154], which is a privacy-preserving, blockchain-based fully decentralized FL system. Biscotti utilizes many cryptographic primitives to defend against model poisoning attacks and information leakage attacks: Multi-Krum defense prevents peers from poisoning the model, differential privacy protects shared model updates from inference attacks, and Shamir secrets provide secure aggregation. They propose their own Proof-of-Federation (PoF) consensus algorithm for generating the new block, which represents a single iteration of SGD. PoF consists of three stages: adding noise to the SGD updates, validating the updates, and securely aggregating the updates. Verifiable Random Functions (VRFs) are used to select a subset of peers responsible for the different stages of the PoF. The peers are selected proportional to their stake which is the reputation that a peer acquires by positively contributing to the model training.

Experiments with Biscotti showed resilience against label-flipping poisoning attacks when 30% of the peers were malicious [154]. They also showed that Biscotti was able to converge under node churn (nodes joining and leaving/failing) that happened at the rate of 1 node joining and 1 node failing every 1.875 seconds. However, Biscotti has its drawbacks. It introduces a privacy-accuracy trade-off when differential noise is added to the SGD updates, which reduces the accuracy of the final model. Multi-Krum based model update validation scheme also has issues, it needs to observe a large number of honest updates in each round, and it can also reject updates from peers that have non-IID data. In addition, Biscotti does not scale to large ANN model with millions of parameters due to communication overhead, and the stake based selection of the peers for the PoF stages does not account for peers that turn malicious after gaining enough stake in the system.

All the previously handled peer-to-peer learning methods assume full cooperation over the whole training time. In addition, the agents do not decide about whether they cooperate or not; the algorithms require them to exchange information at every round. With regard to our vision, we are interested in agents that are more self-interested and consider whether it is beneficial to cooperate. To this end, Yu et al. apply ATC diffusion strategy to learn a common target parameter among a network of self-interested agents [157]. They formulate all the interactions between the agents as successive one-shot games. Each agent has a cost function that combines estimation accuracy and communication cost. At the beginning of each round t, agents are randomly matched into pairs. Each agent k has calculated the adaptation step in Eq. (9) to obtain the intermediate result ψ_k^t. Then, based on some prior knowledge that can be exchanged when the agents are paired, each agent k evaluates the expected cost of sharing ψ_k^t with the other agent. However, if the agents choose their action purely based on the minimization of instantaneous combined cost, the dominant strategy is not to share estimates. This is because the agents do not have a way to predict each other's actions, which results in a lack of belief in the other agent's actions.

To address the lack of belief, Yu et al. introduce a reputation scheme to allow an agent to assess the belief it has about the other agent's actions. Each agent k holds a scalar reputation parameter that summarizes the history of the other agent's actions as viewed by the agent k. These reputation parameters can be used to predict whether the other agent will exchange information, and they also serve as incentives to share because if an agent chooses not to share when it is beneficial, the agent's reputation score is reduced. The sharing is beneficial when the improvement in the parameter estimation accuracy outweighs the cost of the communication. Using this reputation scheme, Yu et al. formulate a simple action-choosing policy for the agents. Their experiments with 10 agents show that the agents will cooperate in the beginning of the training, but will stop when the communication cost outweighs the improvement in the estimation accuracy. They also show that the convergence rate improves under their reputation scheme when compared to self-interested agents without a reputation scheme and to the fully cooperative agents.

Another thing to note is that the previous decentralized approaches target the convergence into a global consensus model. Vanhaesebrouck et al. take a peer-to-peer learning approach where the primary goal is for each agent to learn a personalized, local model [144]. They present a collaborative learning

setting, where agents have their own learning objectives and train their own local models while simultaneously interacting with a neighbourhood consisting of other agents with similar learning objectives. If we denote with $\theta = [\theta_1; \ldots; \theta_K] \in \mathbb{R}^{Kd}$ a stacked vector that consist of all local model parameters $\theta_k \in \mathbb{R}^d$, the learning problem can be formulated as

$$\min_{\theta} \left(\frac{1}{2} \sum_{k<l}^{K} W_{kl} \|\theta_k - \theta_l\|^2 + \mu \sum_{k=1}^{K} D_{kk} f_k(\theta_k) \right), \quad (10)$$

where $W \in \mathbb{R}^{K \times K}$ is a nonnegative weight matrix, where $W_{kl} = 0$ if $k = l$ or if k and l are not neighbours, $\mu > 0$ is a trade-off parameter, and $D_{kk} = \sum_{l=1}^{K} W_{kl}$ is a normalization factor. The weight W_{kl} is supposed to represent the similarity between the agents' objectives, that is, the more similar the objectives of agents k and l are, the larger the W_{kl}.

The leftmost term in Eq. (10) is a quadratic form used to smooth the models within the neighbourhoods. The larger the W_{kl}, the closer the models of agents k and l are encouraged to be. The rightmost term in Eq. (10) is a sum over the losses of each local model on the local data set. This term prevents significant decreases in the local model accuracy as a result of the smoothing.

Vanhaesebrouck et al. [144] reformulate the problem in Eq. 10 into an equal *partial consensus* problem and propose an asynchronous gossip algorithm based on ADMM for solving it. The partial consensus is formulated as a constraint requiring that two neighbouring agents agree on each other's personalized model parameters. Introducing such a constraint requires that each agent keeps a local copy of its every neighbour's model parameters.

The ADMM based algorithm is computationally expensive and requires the introduction of many auxiliary variables. Bellet et al. [158] propose a simpler and faster decentralized block coordinate descent algorithm for solving the same problem. They also address data privacy issues by applying differential privacy in the model updates sent between neighbours.

5) Efficient Model Inference: An important aspect to consider in model training is how to ensure efficient and fast inference with the models. Resource-constrained nodes often face significant challenges when they need to calculate ANN model outputs. A large ANN model may not fit into a node's memory, not to mention the computational costs involved with executing ANN models. While cloud-based inference would remove these challenges, it does not serve the latency requirements of real-time applications, and raises issues of data privacy as the input data must be sent to cloud. Thus, methods for adapting ANN models to fit into the resource-constrained nodes are called for. This usually introduces trade-offs between inference accuracy and latency [12].

Adapting ANN models can start from designing new model architectures that require less from the hardware, or from compressing existing models through, e.g., low-rank approximations, knowledge distillation, pruning or parameter quantisation [12]. Another viable option to reducing resource consumption and latency is conditional computation. This means that unnecessary calculations are turned off, such as early exit in the case of neural networks, where model execution is stopped when a target output confidence level is reached [9]. Model splitting is a popular approach for model inference, where a neural network is cut into parts that are executed by different nodes [9]. However, finding a cut-point that best balances the issues of latency, privacy and resource consumption is not a trivial task [16].

6) Secure Machine Learning: Considering the security and privacy of machine learning, there are three main components each with their own confidentiality and integrity threats:

1) **Data**. The accuracy of data is essential in training an accurate model. Poisoning attacks attempt to disrupt the training process resulting in a decrease in the performance of the trained model. The data is also needed for the training process, but may contain sensitive information which should not be disclosed to unauthorized parties.
2) **Training process**. The computing party needs to have access to the training data and can potentially deanonymize the data contributors or to insert its own data into the training set and create a backdoor (or a trojan) into the model [159]. Such a model could have surreptitious behavior on certain inputs and be hard to the detect. In FL, the local models could also include poisoned models, leading to a bad global model.
3) **Model**. If the trained model is not kept secret, it is possible to recover information about the training data through model inversion [160] or membership inference attacks [161].

Privacy-preserving machine learning techniques enable the training and using of machine learning models without the disclosure of the private training data. While FL is often considered to protect privacy by keeping the training data local, it is not sufficient especially if the local models are based on sensitive data due to inference attacks [161].

Anonymization techniques remove any personally identifiable information from the training data. Such techniques include, for example, k-anonymity [162], l-diversity [163], t-closeness [164] and differential privacy [165, 166]. These approaches can be used to hide the information of an individual inside the training set while maintaining the patterns of groups of users. In differential privacy, which has been extensively used in machine learning, a specific amount of random noise is inserted into the data to provide statistical hiding for individual data while preserving the utility of the data set.

Cryptographic techniques have been also suggested for the protection of privacy. Homomorphic encryption enables the computation and classification on encrypted data [167]. Fully Homomorphic Encryption (FHE) schemes enable any type of computations to be performed on the ciphertext space [168]. FHE is computationally demanding, but there are limited versions, such as additively homomorphic Paillier's cryptosystem [169], that are sufficient for certain classification

use cases [170, 171]. Limited homomorphic encryption is sufficient for the protection of neural network models [172]. A closely related primitive is functional encryption that enables a party in possession of a restricted secret key to compute a function on the plaintext directly from the ciphertext [173, 174], but learns nothing else of the plaintext. Functional encryption facilitates privacy-preserving machine learning without online functionality [175]. Furthermore, verifiable computing can attest the correctness of the computation by enabling the client to efficiently verify it [176].

Secure Multi-Party Computing (MPC) schemes are interactive protocols that enable users to jointly compute a function over secret inputs [177]. It enables agents to collaboratively compute a machine learning task with hidden inputs. It is also possible to hide the model parameters from the agents. The participants learn only the output of the function and nothing more. Secure MPC can be used to protect existing models, even those based on deep neural networks [178], but incurs a communication cost for the participants. Nevertheless, secure MPC is a promising approach and automatic tools for privacy-preserving collaborative machine learning have been devised based on it [179]. There are also schemes that combine the homomorphic encryption approach with the secure MPC approach in order to optimize the tradeoff between computation and communication costs [180, 181].

B. Decision Making and Negotiation

In the highly dynamic computing continuum, decision making strategies[13] learned by the agents must be stable, but adaptive enough to conform to changes. Agents have to learn strategies based on their own local experience and interactions with neighbouring agents in order to make optimal decisions.

In this section, we will focus particularly on sequential decision making methods that could lead to efficient solutions in the computing continuum orchestration. *Multi-Agent Learning* (MAL) is a field that studies learning paradigms for MAS and develops algorithms to create adaptive, optimal decision making policies for agents. Reinforcement learning is the most commonly studied learning paradigm for MAS [182]. We believe that reinforcement based learning is a key ingredient for intelligent computing continuum orchestration, and hence, this section will mainly focus on reinforcement learning based sequential decision making.

Making a decision can be equivalent to reaching an agreement between self-interested agents, which requires negotiation techniques. For example, an agent can make the decision about offloading a task by itself based on, e.g., the current load on the node, task size and predictions about the changes in the load. However, the decision of where to offload the task is the outcome of negotiation among the autonomous, self-interested agents. An agent cannot make such a decision by itself, as it would violate the autonomy of the other agent. Hence, this section will also touch upon the current state of multi-agent negotiation.

[13]Words 'policy' and 'strategy' are used interchangeably in this survey

Any communication between self-interested agents requires a common language. How this common language is established is a highly important research question, especially in the open computing continuum environment where different agents are designed and implemented by different people. Hence, we will also take a brief look at AI methods that enable learning a shared language.

1) Reinforcement Learning: **Fundamentals.** *Multi-Agent Reinforcement Learning* (MARL) environment is usually formulated as a stochastic game, also known as a Markov game [183, p. 160]. A Markov game is formally defined as a tuple $(\mathcal{N}, \mathcal{S}, \mathcal{A}, T, R)$, where $\mathcal{N}$ is a finite set of K agents, $\mathcal{S}$ is a finite set of states, $\mathcal{A} = A_1 \times \cdots \times A_K$ is the joint action space with A_k being the action space of agent k, $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the transition probability function ($T(s, a, \hat{s})$ is the probability of transitioning from state s to state $\hat{s}$ after taking action a), and $R = (R_1, \ldots, R_K)$, where $R_k : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is a real valued reward function for agent k.

MARL generally recognizes three different settings for the interaction between the agents: fully cooperative, fully competitive and mixed setting [184]. In fully cooperative settings, agents have a shared goal and hence, each agent usually also has the same reward function: $R_1 = \cdots = R_K$. The goal in fully cooperative settings is to maximize the common return. Fully competitive settings are formulated as zero-sum games, where two agents have opposing goals, i.e., $R_1 = -R_2$. Mixed settings are formulated as general-sum games, where there are no restrictions on the goals and relationships among the agents; agents are *self-interested* and their individual rewards can be in harmony or in conflict with each other.

Agents interact with the environment in discrete time steps. Each agent k follows a policy $\pi^k : \mathcal{S} \times A_k \to [0, 1]$ that maps each state-action pair (s, a^k) to the probability of selecting the action a^k in the state s. The state-value function of agent k $V^k : \mathcal{S} \to \mathbb{R}$ can be defined as the expected sum of discounted rewards that is obtainable from the state [184]. That is, for any joint policy $\pi(s, a) = \prod_k \pi^k(s, a^k)$ and state s

$$V^k_{\pi^k, \pi^{-k}}(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R_k(s_t, a_t, s_{t+1}) \middle| s_0 = s \right]. \quad (11)$$

Here, $-k = \mathcal{N} \backslash \{k\}$ is the set of other agents, $\gamma \in [0, 1)$ is a discount factor used to control how much effect future rewards have on the optimal decision, $R_k(s_t, a_t, s_{t+1})$ is the immediate reward received by agent k after the joint action a_t in state s_t, which transitions the state of the environment to s_{t+1}, t being the current time step. The individual action a^k_t of agent k is defined by its current policy π^k. The goal for each agent k is to find an optimal policy $\pi^{k,*}$ that maximizes the expected sum of discounted rewards[14], i.e., $\pi^{k,*} = \arg\max_{\pi^k} V^k_{\pi^k, \pi^{-k}}(s)$.

From the state-value function formulation in Eq. 11 we can see that the expected return of the agent k is influenced by

[14]For simplicity, we only present the infinite horizon optimality criterion with discounting. However, other criteria exist as well, such as a finite horizon criterion or an infinite horizon criterion with averaging [185]

the actions of the other agents. In the case where the other agents are also learning agents, i.e., their policies are non-stationary, the environment itself also becomes non-stationary. This opponent-induced non-stationarity is a particularly severe challenge in the computing continuum, as it violates the fundamental assumption of the Markovian property (a stationary environment) behind single-agent RL algorithms [186]. Thus, algorithms developed for finding optimal policies in the single-agent settings should not be applied as is in multi-agent systems, because all derived guarantees are lost [182].

MARL algorithms developed for finding the optimal policy $\pi^{k,*}$ for each agent k need solution concepts to address the dependence of an agent's optimal strategy to the strategies of the other agents. *Nash Equilibrium* (NE) is a common choice for the solution concept. A NE for a Markov game is defined as a joint policy $\pi^* = (\pi^{1,*}, \ldots, \pi^{K,*})$ such that for any $s \in \mathcal{S}$, $k \in \mathcal{N}$ and π^k, $V^k_{\pi^{k,*},\pi^{-k,*}}(s) \geq V^k_{\pi^k,\pi^{-k,*}}(s)$ [184]. In other words, the policy $\pi^{k,*}$ of agent k is its best response to the joint policy $\pi^{-k,*}$ of the other agents. Most of the MARL algorithms aim to converge to a NE, as it is a stable point from which none of the agents has any incentive to deviate. However, NE is not often unique, and it does not guarantee optimality, because it assures that no single agent can unilaterally improve their reward, not that the global reward is maximized or that there is no other equilibrium where agents could simultaneously improve their rewards [187]. In addition, convergence to NE is a reasonable goal only under the assumption that agents are perfectly rational and capable of infinite mutual modelling [184]. However, this is not a reasonable assumption in many real-life situations, such as when agents with bounded rationality (e.g., computationally limited, which is a relevant characteristic of agents in the computing continuum) are involved, because such agents are only able to do finite mutual modelling. These situations call for different solution concepts to pursue as the main goal of MARL algorithms.

One very important aspect besides the stability of a MARL algorithm is the capability to adapt to the strategies of other agents. In its extreme form, this is manifested in the so called agent tracking algorithms for mixed settings, in which agents follow and observe the actions of others to build models about their strategies, and then aim to provide their best response to these models rather than focusing on having a stable strategy [188]. However, a practical algorithm needs to find a balance between the *stability* and *adaptivity* of a decision making strategy. In other words, algorithm should detect when the change in the other agents' behavior is more permanent, and not react to every little change in behavior that may be a result of random exploration.

Developing MARL algorithms is very challenging, because in addition to opponent-induced non-stationarity, there is a plethora of other challenges to consider. These issues include balancing exploration-exploitation trade-off, sample inefficiency (it takes an incredible amount of samples and time to train an adept agent), sparse rewards (every transition does not return a feedback), dependability (in the sense of, e.g., guaranteeing a certain level of average reward), coordination, communication efficiency, preventing the overfitting of agents, handling dysfunctional or malicious agents, credit assignment problem (in fully cooperative settings), state and action space explosion (scalability) and partial observability. In this survey, we are mainly focusing on the core issues of opponent-induced non-stationarity, scalability and partial observability.

From theoretical perspective, the introduced Markov game framework forms the theoretical foundation of MARL along with the extensive form game framework [184]. In Markov games, agents choose their actions simultaneously and receive an immediate reward, whereas in extensive form games, agents choose their actions sequentially and receive a reward only at the end of an action sequence. Markov game framework can handle only the fully observable environment, whereas extensive form framework is able to handle the partially observable environment. Extensive form game framework is generally used in non-cooperative (fully competitive or mixed) settings [184]. However, extensive form games are not always a good model for large or realistic multi-agent settings [183, p.147].

How to handle partial observability, that is, not being able to observe the exact environment state, or the actions and/or rewards of the other agents, is an important question, because it is not often realistic to assume full observability in practical MARL applications already due to the noise in sensor observations and communication channels. It is possible to extend Markov game framework to the partially observable case. The resulting model is called Partially Observable Stochastic Game (POSG) [189], which is formally defined as a tuple $(\mathcal{N}, \mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{P}, R, b_0)$, where $\mathcal{N}$, $\mathcal{S}$, $\mathcal{A}$, and R are defined as in Markov games, $\mathcal{O} = O_1 \times \cdots \times O_K$ is the set of joint observations with O_k being the finite observation set of agent k, and $\mathcal{P}$ is a set of state transition and observation probabilities with $\mathcal{P}(\hat{s}, o|s, a)$ being the probability that taking the joint action a in state s leads to transition to state $\hat{s}$ and joint observation o. $b_0 \in \Delta(\mathcal{S})$ is the initial state distribution at time $t = 0$. Policy now maps action-observation histories to actions.

In fully cooperative settings, POSG reduces to decentralized partially observable Markov decision process (dec-POMDP) [190], where all agents have the same reward function. However, even in the fully cooperative case, partially observable models are very difficult to solve, because it has been proved that solving decentralized POMDPs is NEXP-complete [191]. Most MARL algorithms proposed for partially observable problems consider only fully cooperative settings, and they require some form of centralization so that the decentralized problem can be reformulated as a centralized POMDP [184]. Reformulation as a centralized model can also be achieved by communication, for example, if each agent broadcasts their observation, they all receive the joint observation and can compute joint beliefs, which basically turns the decentralized problem into a centralized POMDP.

Note that the traditional definition of POSG is underspecified, because it does not include the specification of the

communication capabilities of the agents. Oliehoek and Amato introduce Multiagent Decision Problem (MADP) to address this issue [190, p. 25]: they specify two models, environment model and agent model. The previously introduced POSG model can be interpreted as a Markov multiagent environment model, which is underspecified in that it does not specify the information on which the agents can base their actions, or how they update their information. This can be made explicit with an agent model, which for each agent k is a tuple $(\mathcal{I}_k, I_k, A_k, O_k, \mathcal{Z}_k, \pi^k, \iota_k)$. Here, $\mathcal{I}_k$ is a set of information states, I_k is the current information state of the agent, A_k and O_k are the action and observation sets of the agent, $\mathcal{Z}_k$ is the set of auxiliary observations that can be obtained through, e.g., communication, π^k is the (stochastic) policy of the agent, and $\iota_k : \mathcal{I}_k \times A_k \times O_k \times \mathcal{Z}_k \to \Delta(\mathcal{I}_k)$ is information state function or belief update function.

In MADP, the problem designer must specify optimality criterion (such as the expected sum of discounted rewards), the environment model, and a subset of the elements of the agent model. The algorithm developed for solving the problem must optimize the non-specified elements of the agent model, in order to maximize the value given by the optimality criterion. We believe that the inclusion of this type of agent model is particularly relevant in the future computing continuum orchestration solutions, as it characterizes the knowledge and capabilities of an agent, allowing a more local, agent-centric view on the decision problem.

Algorithms. Hernandez-Leal et al. provide a useful categorization of the proposed MARL algorithms based on how they address the opponent-induced non-stationarity [186]. They identify five categories with increasing sophistication in the approach to handling the non-stationarity: ignore it, forget it, respond to target opponent models, learn the opponent models, or rely on a theory of mind. Ignoring the opponent-induced non-stationarity has been a popular approach due to its simplicity and easiness: agents are treated as *independent learners* in an environment where opponent-induced non-stationarity is regarded as stochastic noise within the transition model, which enables the use of algorithms developed for single-agent learning, such as Q-learning. In other words, algorithms that ignore non-stationarity learn a stable policy while assuming that other agents are using stationary policies, i.e., are not learning. Obviously, if opponents[15] are learning or change their strategy, these algorithms will fail. We do not believe that this approach will be a sustainable and feasible approach in the computing continuum orchestration.

Algorithms using forget approach are typically model-free algorithms that adapt to the changing environment by updating the strategies with the most recent information while forgetting old information. A representative example of algorithms in this category is WoLF-PHC [192]. WoLF-PHC is an algorithm for mixed settings that does not require observability of the actions or rewards of other agents and is based on using Win-or-Learn-Fast (WoLF) principle in policy updates and Q-learning in Q-function updates. It handles non-stationarity by adjusting between two learning rates based on whether agent is interpreted to be winning or losing. The interpretation is determined by comparing the current policy's expected payoff with that of the average policy over time. WoLF-PHC can be used in cases where agents have heterogeneous learning strategies, but it assumes that the opponents are slowly adapting. This type of adaptation based on the newest information and the notion of the current status (such as winning or losing in WoLF-PHC) while forgetting old information could possibly lead to some orchestration solutions among the more resource-constrained agents that cannot be expected to uphold complex models about the other agents or extensive histories of their interactions with the environment.

Algorithms that respond to target models optimize against clear and defined opponent strategies. There exist many algorithms in this category due to the fact that it is easier to give guarantees against specific opponents than general classes [186]. Replicator Dynamics with a Variable Learning Rate (ReDVaLeR) [193] is a good example of algorithms in this category. It is a model-free algorithm for mixed settings, which provably learns a stationary best response against stationary opponents and NE against adaptive opponents that *all* use the same ReDVaLeR algorithm (self-play). In addition, ReDVaLeR provides a constant bound on expected regret, which makes the algorithm robust against opponents that use arbitrary non-stationary policies. However, it assumes that an agent can distinguish between self-play and otherwise non-stationary agents, requires that opponent actions are observable, and has been developed for repeated games (single state).

Many algorithms in the respond to target opponent models category have been either directly developed for or evaluated only in 2-player competitive games, and they do not often address in any way what happens when an agent encounters other agents that do not belong to the target classes. However, these algorithms can offer noteworthy solutions that could be useful in computing continuum orchestration solutions, such as the guarantee of a bounded regret against opponents using arbitrary non-stationary strategies in ReDVaLeR algorithm. In the open computing continuum environment it is unrealistic to assume that all agents are using the same learning algorithm; guaranteeing a certain level of performance against other agents using any type of learning algorithms is desirable.

Algorithms that learn opponent models form strategies based on how the opponent is behaving and aim to adapt the models to changes in the opponent behavior. That is, the algorithms update the opponent model and their strategy constantly to keep up with the non-stationary opponents. A representative example of algorithms in this category is BPR+ [194], which has been developed for a repeated game where the opponent switches among several stationary strategies in a way that it can either start using a completely new strategy or return to a previously used one. BPR+ learns all models through interactions, and stores previous models to memory when it detects a change in opponent's behavior, after which

[15]Similarly to Hernandez-Leal et al. [186], we are using the word 'opponent' to refer to another agent in the environment regardless of whether the objectives of the agents are aligned or not.

it starts learning a new model. The opponent behavior is modelled with an MDP. Thus, it needs to be able to observe opponent actions, can require a lot of memory resources and is not a very scalable algorithm.

Theory of mind algorithms are the most evolved ones in regard to handling non-stationarity as the models assume that the opponent is also establishing opponent models, whereas algorithms in learn models category do not take into account whether the opponent is modelling them or not. These kind of algorithms naturally require high computational resources to solve as they aim to perform complex, recursive and strategic reasoning. Thus, majority of the work studying this approach have been done in the context of one-shot, stateless games [186]. Interactive POMDP (I-POMDP) [185] is perhaps the most known framework in this category for sequential decision making. It extends POMDP model to distributed multi-agent systems by incorporating models of other agents into the state space.

Learn opponent models and theory of mind approaches could only be deployed by agents with basically unlimited resources due to their high computational and memory load. Furthermore, the approaches do not easily scale up to a large number of agents. Theory of mind approaches have been mainly used for predicting the outcome of one-shot games, and they cannot usually be used for online learning. For example, I-POMDP requires that the transition and observation probabilities are known. Some work that aims to enable online learning with I-POMDPs [195] and increase the scalability of I-POMDPs [196] does exist.

2) Deep Reinforcement Learning: Computing continuum environment modelled as a MAS consists of a large number of agents, which induces a scalability issue for MARL algorithms. This is due to the combinatorial nature of MARL: joint action space dimension increases exponentially with the number of agents. Many MARL algorithms are based on value or policy iteration, which involves tracking values for each state or state-action pair. This is obviously not possible when the dimensionality of state and joint action spaces grows. One solution to the scalability issue is function approximation: rather than tracking the state value or state-action value functions in tabular form, these functions are approximated with, for example, artificial neural networks. Combining deep learning with MARL has resulted in the formation of the field *Multi-Agent Deep Reinforcement Learning* (MADRL), the algorithms of which have been particularly successful in vision-based domains, such as games [197]. However, the main disadvantage of these methods is that providing theoretical guarantees about convergence of algorithms in the case of function approximations is extremely difficult for non-linear ANN approximations [184].

Using ANN models to approximate value functions brings the benefit of better generalization across states and removes the need to manually handcraft features that are used to represent state information. However, the problems of non-IID training data and divergence from optimal values pester the use of ANN approximators [197]. Many single-agent DRL methods that are based on the seminal method Deep Q-Network (DQN) [198], use experience replay memory to break the ties between the highly correlated sequential training samples (state-action pairs), and address the divergence issue by adopting two NNs to approximate value function so that the weights of the other, called target network, are updated only periodically. Transferring experience replay into MAS is not straightforward because past observations become obsolete due to the adaptation in agents' policies. The two main approaches that have been proposed to overcome this are fingerprinting to disambiguate the age of the samples, and leniency values, which are basically temperature values that gradually decay by the number of state-action pair visits [197].

Multiple MADRL algorithms have been proposed in the literature to overcome the same type of challenges that traditional MARL algorithms encounter [197, 199, 200]. One interesting method that aims to overcome the opponent-induced non-stationarity is Deep Loosely Coupled Q-network (DLCQN) [201]. DLCQN is based on the loosely coupled Q-learning [202], where each agent has an independence degree for each state, giving the probability that the agent can act independently in a given state. This degree is adjusted whenever a negative reward is received. However, loosely coupled Q-learning was originally developed for 2-agent robot coordination problems, and DLCQN does not extend this to a bigger number of agents. Nevertheless, the notion of an independence degree simplifies the decision making of an agent in the states where it can assume to be relatively independent of other agents. Studying the applicability of such a notion in computing continuum orchestration problems could be beneficial.

In MADRL algorithms, partially observable environments are usually handled by utilizing deep recurrent neural networks, such as Long Short Term Memory (LSTM) networks. Solutions that aim to overcome both, non-stationarity and partial observability, utilize some form of communication or centralization to share information and coordinate the actions of the agents. Currently, *Centralized Training with Decentralized Execution* (CTDE) is the standard, most widely adopted paradigm when it comes to learning in a partially observable, non-stationary MAS [184, 200]. It involves sharing information (e.g., observations and actions) during training through a centralized entity. This entity can provide the information for all the agents so that they can optimize their policies, or the central entity may also calculate the optimal policy. During execution, all this extra information is discarded, and an agent simply follows the policy it calculated during training or received from the central entity after the training. CTDE is particularly popular in algorithms developed for solving fully cooperative dec-POMDPs, where it simplifies the theoretical analysis [184]. CTDE works mainly in applications where the learning can be done a simulator or a laboratory, where there are no communication constraints and extra state information is available [199].

CTDE in its current state is not a very suitable paradigm for learning policies in the mixed settings of the computing con-

tinuum environment. Agents cannot use the policies learned in a limited, simulated environment as is in the dynamic computing continuum. Furthermore, agents should be able to adapt their policies if during execution they note that their performance with the current policy deteriorates. That is, they should be able to go back into the training phase, which in CTDE algorithms would require access to all the extra information that was available during the initial training in a simulator. Hence, training paradigm in the computing continuum must emphasize decentralization with the ability to communicate with the local neighbourhood, while taking into account that this communication can be intermittent and limited. However, we do recognize that policies learned in a simulated environment could provide prior information for learning in the real environment (i.e., transfer learning aspect, which will be discussed later).

Mixed setting is a notoriously hard setting when it comes to developing provably convergent algorithms even in the MARL domain [184], not to mention when utilizing non-linear ANN approximations. Mixed setting in RL is generally underexplored when compared to fully cooperative or fully competitive settings. We see mixed setting MADRL algorithms as an important research direction for developing solutions in computing continuum orchestration. The following paragraphs highlight some prominent MADRL algorithms developed for mixed setting that aim to account for the learning behavior of other agents.

Learning with Opponent-Learning Awareness (LOLA) [203] is a policy-gradient based algorithm, where an agent estimates the policy parameters of other agents based on experience (must be able to observe actions and rewards of all agents over an episode). Then it tries to predict the one-step policy parameter updates of other agents, and updates its own policy parameters taking into account the updated parameters of other agents, which affect its own expected reward. In other words, LOLA agent tries to anticipate the learning of other agent and provide its best response to this learning. However, in practical scenarios LOLA assumes that other agents do their one-step updates naively, i.e., do not try to anticipate the learning behavior of the LOLA agent, and that the agents' policies are parameterized in the same way.

Multi-agent Deep Deterministic Policy Gradient (MADDPG) [204] is an actor-critic policy gradient algorithm that can be applied in any type of MARL setting. MADDPG aims to learn a policy so that there is no need for a model of the environment dynamics, nor any restrictions on the communication method between the agents (i.e., MADDPG can also be used to learn communication actions without assuming any particular type of communication between the agents). MADDPG achieves this with CTDE paradigm, where each agent has their own critic, but these critics are trained in a centralized manner. The behavior of other agents is taken into account by augmenting each critic with extra information about the policies of other agents. Actors rely only on the local observations, i.e., only local information is needed during execution.

In addition to partial observability, agents may have to deal with extremely noisy observations that are weakly correlated with the true state of the environment. In this case, MADDPG where the action selection is based only on the extremely noisy local observations will not function well without some extra information. MADDPG-M that combines MADDPG and a communication medium has been proposed to address this type of circumstance [205]. In MADDPG-M, agents need to decide whether their observations are informative enough to share with other agents, and this communication policy is learned concurrently with the main policy that determines the environmental action of an agent. Communication policy is learned in a centralized manner in a slower time scale than the main policy that is learned in a decentralized manner based on both, the local observations and the information obtained through the communication medium.

3) Federated Reinforcement Learning: Training accurate ANN models requires a substantial amount of training data, and it would take from one agent a long time to collect enough observations in the environment to attain enough data for training accurate ANN models. Furthermore, the observation range of one agent is limited to its local view. Sharing observations between agents is one way to increase the sample efficiency and help one agent to attain a more global view on the environment state, but it is not always possible due to data privacy and communication efficiency issues.

Combining FL with MARL opens up the possibility for a group of agents to collaboratively learn ANN models (actor and/or critic models) by periodically exchanging encrypted ANN model parameters without actually sharing any observations [206]. The studies on the combination of FL and RL are currently at scarce particularly in the type of real-life settings encountered in the computing continuum, that is, a setting where agents share the same environment while having their own sets of observations and actions. We see federated RL as a promising research direction for enabling MARL training in real-life settings (basically bringing CTDE into a real-life environment). Furthermore, the possible applicability of decentralized FL methods (See Section VI-A) in RL settings is another open research direction, which could take us towards a decentralized training paradigm.

Even though FL allows sharing experience in a privacy preserving way, it has the disadvantage of confining each agent to using the same, homogeneous ANN structure. To address this issue, Federated Reinforcement Distillation has been proposed [207], where each agent shares their local proxy experience memory with an aggregator that creates a global proxy experience memory. Each agent then aims to minimize the cross entropy loss between their local policy and the policy of global proxy experience memory. An improved version that uses mixup data augmentation algorithm to interpolate the global experience replay memory has been proposed as well [208].

4) Multi-objective Reinforcement Learning: RL methods usually aim to optimize a single objective. However, any orchestration solution in the computing continuum must inher-

ently consider multiple objectives with regard to cost, quality and resources. Multi-Objective Reinforcement Learning (MORL) is a field that aims to develop methods for RL problems where multiple different objectives must be optimized. This means finding an appropriate trade-off between competing objectives. A central solution concept in MORL for defining the set of best trade-off solutions is *Pareto efficiency* [209]: the solution for the multi-objective problem cannot be Pareto dominated by any other solution, meaning that there cannot be any other solution where some objectives will be improved and no objective will be deteriorated. The set of such non-dominated, mutually incomparable solutions is called *Pareto front*.

In MORL, instead of a single scalar reward, agents receive a reward vector where each component measures the performance on a different objective. Majority of MORL approaches handle the multiple objectives with scalarisation. This means that the reward vector is mapped into a scalar value with a scalarisation function, which most typically is a linear weighted sum of the reward vector components [210]. After this, methods developed for single-objective RL problems can be used. However, scalarisation has multiple disadvantages. It requires a priori information about the preferences over different objectives, such as whether low latency is more preferred than low cost. Further, if the preferences change significantly over time, the policy must be re-learned with the new reward function. It is also important to note that non-linear scalarisation may in some cases break the applicability of single-objective RL methods [211], and that linear scalarisation is known to find solutions only from the convex regions of the Pareto front [212].

In many situations, a more desirable way to handle the multiple objectives is to find a set of policies that properly approximates the Pareto front, a so called multi-policy approach [209]. In this way, it is easier to evaluate the different trade-offs that can be done between the different objectives, and create a selection mechanism that can choose the policy for execution based on the current preferences over objectives. In principle, this could be done with the scalarisation approach by running the learning algorithm with different scalarisations. However, choosing the scalarisations so that the resulting policies are properly spread in the objective space is difficult [213], and running the same algorithm several times with different reward functions in the computing continuum is infeasible. Hence, the MORL algorithms for computing continuum should learn a set of optimal policies in a single run.

One example of an algorithm that can provide a set of optimal policies in a single run is Pareto Q-learning [209], which, however, has been proposed for a single agent setting. In general, MORL applied in a multi-agent setting has not been comprehensively explored. This line of research is essential for the future computing continuum orchestration solutions, as balancing trade-offs between different objectives is unavoidable. How to efficiently implement learning in a MAS with multiple objectives, while taking into account adaptation to other agents and changing preferences, as well as limited resources is an open research question.

5) Transfer Learning: Transfer learning aims to accelerate the learning process by reusing knowledge. Transfer learning methods make it possible to transfer knowledge from powerful, well-trained agents to the agents that are just starting the learning procedure or that do not have enough time, memory or computational resources to train as accurate models. For example, agents operating in the cloud have basically unlimited resources for reasoning and learning, and their knowledge could be utilized by the resource-limited agents operating in the edge.

Furthermore, learning a policy from scratch in a real environment is not feasible due to the huge amount of experience needed to train a policy, and because in the early stages of learning, agents can make very awful decisions. Hence, knowledge obtained through, for example, training in a simulated environment or from demonstrations by human experts could offer an excellent starting point for further learning in the real environment.

Generally, transferring knowledge has been studied from three different perspectives: an agent can reuse its own knowledge from previous tasks, agent can gain knowledge from observing other agents, or agent can get direct advice from experts [214]. The first aspect is referred to as intra-agent knowledge transfer, whereas the latter two belong to the inter-agent knowledge transfer.

In terms of the intra-agent knowledge transfer, one interesting paradigm is curriculum learning, where the idea is to divide a problem into subproblems of varying difficulty level. Then starting from the easier tasks, agent transfers knowledge across progressively harder tasks. These approaches are often more concerned with how to define the curriculum, i.e., a task sequence, rather than how to reuse knowledge, since all tasks inside a curriculum are assumed to be inside the same domain [214]. In this regard, Silva and Costa propose a method for automatically generating a curriculum based on object-oriented task descriptions [215].

The majority of the transfer learning literature is focused on the inter-agent knowledge transfer [214]. These approaches are based on the assumption that all tasks are in the same domain, and knowledge is transferred through either explicit or implicit communication. These methods can be direct action suggestions from other agents (usually in fully cooperative settings), transferring human expertise, receiving heuristics or reward shaping from teachers to improve exploration, learning from demonstration or imitating other agents through observing them. There is not yet a lot of work that consider knowledge transfer for agents learning with deep networks; however, there are works indicating that unprincipled knowledge transfer in MADRL can lead to more catastrophic negative transfer effects than in the traditional MARL [216, 214]. In addition, the security aspect of knowledge transfer between agents has not been considered practically at all, that is, agents participating in the knowledge transfer are assumed to be benevolent.

Transfer learning is among the most important research topics for the future computing continuum orchestration,

because the ability to reuse knowledge is a corner stone of the type of intelligent, adaptive and autonomous agents considered in our vision. Current research is heavily dependent on human efforts and validates proposed approaches mainly in simulations of simple problems [214]. However, developing solutions for open and complex real-life environments such as the computing continuum is extremely hard. Difficulties already start from such a fundamental issue as knowledge representation, that is, the transfer of knowledge between the agents relies on the fact that the agents have the same way of representing knowledge. To which extent such an assumption is valid among the heterogeneous agents of the computing continuum is an open research question.

6) Mean-field Theory: As previously stated, computing continuum entails a large number of agents. Most of the approaches for reinforcement learning in a MAS are designed for or validated in settings with a maximum of tens of agents. Many methods that aim to model other agents require a separate model for each of the agents in the environment. This very quickly becomes infeasible when the number of agents grows, not to mention when also considering agents with limited resources.

Solutions for this scalability issue could be searched with *mean-field game theory*. Mean-Field Games (MFGs) are designed for situations with possibly thousands of agents. In MFG, an N-player game is reformulated as a 2-player game, where each agent is playing against a virtual opponent, which presents an average player of the entire (homogeneous) population. In optimal control theory, where an agent population is modelled as a controlled stochastic dynamical system, solving MFGs involves solving two coupled partial differential equations, which are usually approximated with neural networks [217].

Studying mean-field theory in MARL has only recently gained attention [184]. One notable work is by Yang et al. [218], who introduce a mean-field Q-learning and mean-field actor-critic algorithms for non-cooperative settings. In their approach, an agent aims to find the best response against the mean action of its neighbourhood (which approximates the joint action in Q-function update), i.e., it needs to be able to observe the actions and rewards of its neighbours. Further research is needed to study the validity of mean-field theory in MARL, especially in the type of real-life settings that include partial observability.

7) Evolutionary Game Theory: Combining evolutionary game theory concepts with RL opens many interesting possibilities. The benefit of evolutionary game theory over traditional game theory is the fact that it does not require agents to be hyper-rational players, and an equilibrium can change dynamically [219]. Evolutionary game theory has even been proposed as the preferable framework for studying multi-agent learning formally [220]. This stems from the concept of *replicator dynamics*, which can be utilised in the game theory to describe an agent's strategy change over time as a game is repeatedly played and a policy iteratively updated. For MARL algorithms, a link between replicator dynamics and RL can be used to derive mathematical model of the infinitesimal time limit of various learning algorithms, i.e., a model of learning dynamics (predicting the behaviour of an algorithm in the limit). These kind of dynamical models are very useful in hyperparameter tuning or in the development of new learning algorithms [220]. Also, "the relative fitness" of different strategies can be compared within a population, dependent on the frequencies of those strategies in the population.

Evolutionary algorithms (such as genetic algorithms, evolutionary programming, genetic programming and evolutionary strategies) have been utilized in reinforcement algorithms that search the space of policies [221]. The main idea is to start with the set of policies and evolve them until a policy with good enough fitness is reached. Evolutionary Algorithm Reinforcement Learning (EARL) methods differ mainly on how the policies are represented, and whether the policy is seen as one 'chromosome' that is evolved as a whole or whether it is divided into parts.

Evolutionary algorithms are an extremely promising direction for complex MARL applications, as they can address several fundamental challenges in RL problems, such as large state spaces (though generalization and selectivity in policy representations), partial observability (if the change in environment is slow) and non-stationarity. However, they are not very suitable for online learning due to the requirement of large number of experiences, and the fact that some generated policies may be very bad. Thus, they should be trained in simulation models of the environment. In addition, EARL algorithms do not preserve information about bad states or decisions, and rarely visited states can drift to random, possibly bad actions due to mutations [221].

8) Swarm Intelligence: We have so far focused on the MARL paradigm that aims to make every agent a highly intelligent decision maker while assuming that the agents are fully rational. This, however, requires considerable resources from a single agent. In the hierarchy of the computing continuum, the lower the agents are, the less resources they have for running decision making algorithms. These agents should still be able to make coordinated decisions about, for example, optimally allocating their energy and communication resources so that they can transmit messages while minimizing the used transmit power and the interference they cause to each other. Such coordinated decision making should happen in a decentralized manner through local interactions. We regard Swarm Intelligence (SI) as a key MAL paradigm for developing self-organizing, decentralized and adaptive decision making solutions for agents with limited capabilities.

SI investigates the emerging behavior of a (large) group of simple, rationally bounded agents that achieve complex, intelligent behavior through their interactions [182]. An inherent assumption in many SI methods is that the agents are fully cooperative, which can also be a reasonable assumption for the low-level agents in the computing continuum at least inside the same administrative domain. An agent with very limited resources does not have enough capacity to learn complex behaviors as a unit, as is the purpose in MARL approaches,

but rather, the complex behavior must be an emergent property of a group of such agents interacting on a local level.

SI has been especially studied for combinatorial optimization, and a plethora of SI optimization algorithms have been proposed in the literature [222]. SI methods have also been investigated inside multiple different domains, such as robotics, Unmanned Aerial Vehicles (UAVs) and IoT [223, 224, 225]. Our purpose is not to provide a comprehensive analysis of the SI research, but rather briefly highlight a couple of concepts that we regard particularly interesting in terms of the computing continuum orchestration solutions, namely *well-informed agents* and *adaptive personality traits*.

When there is a group of simple agents sensing the conditions in an environment and trying to reach a coordinated decision, it is easy to imagine that some agents are more privileged than others: their observations can be less noisy and more accurate, or they may have access to information that the other agents do not have. These well-informed agents should have more impact on the behavior of the group. Such an idea is implemented, for example, in the simple, decentralized decision making algorithm proposed by Yu et al. [226], where the actions of the whole group of agents are guided by the actions of a few well-informed agents. The informed individuals are not explicitly assigned as leaders; the algorithm is based on implicit leadership, meaning that the well-informed agents have a positive confidence factor about their information state, which influences the behavior of the rest of the group.

Adaptive personality traits is a concept that has been utilized particularly in swarm robotics [227, 228]. The idea is to have a set of personality values that affect the action choice. These traits of personality are adapted based on reinforcement learning, that is, traits that lead to a positive feedback are enforced while traits with a negative feedback are weakened. Utilizing personality traits for simple agents could lead to novel solutions in the computing continuum orchestration. For example, they could offer an efficient way to implement the self-interested nature also for the resource-constrained agents; a mixture of two personality traits, cooperative and non-cooperative, determines whether an agent is more inclined to take an action that benefits its own utility or an action that benefits the collective utility.

9) Metareasoning: Bounded rationality is an inherent characteristic for majority of the agents in the computing continuum due to the limited amount of resources they have for reasoning about decisions. A promising research direction for handling the bounded rationality of the resource-constrained agents in the computing continuum is metareasoning. The idea of metareasoning is to monitor the decision making process of an agent to enable it to make 'good enough' decisions in situations where there is not enough time/resources for thorough deliberation [229, 230].

There exists different metareasoning structures for a MAS, which differ in terms of whether metareasoning is part of an operating agent or an agent of its own controlling the operating agents, and whether the metareasoning processes of different agents coordinate with each other or not. Metareasoning approaches have been applied in multiple different types of problems, such as coordination, planning and scheduling, communication, resource allocation, belief updating and task delegation [229].

Metareasoning approaches also differ in the way how the meta-level processes control the object-level processes (i.e., how the monitoring algorithms control the decision making algorithms). They can stop the execution of an object-level algorithm (in the case where the decision making algorithms are so called anytime algorithms [231]), modify parameters or reasoning rules of object-level algorithms, select an object-level algorithm for execution, determine whether it is beneficial for an object-level process to communicate or coordinate with others (and how), provide information about other agent's reasoning strategies (that it has deduced), or determine whether it is beneficial for agents to redefine their relationships (in hierarchical settings, e.g., master-slave) [229].

10) Negotiation: Reaching agreements between agents with conflicting interests is an essential part of the envisioned computing continuum orchestration. Depending on the problem at hand, this negotiation can be bilateral (one-to-one) or multilateral (one-to-many, many-to-many), and it can concern a single issue or multiple issues. Regardless of the number of participants or issues, negotiation in the computing continuum will always be an incomplete information game: agents do not initially know what the utilities and strategies of the other agents are.

Automated negotiation has been widely studied in the literature (see e.g. [232, 233, 234, 235, 236, 237, 238]). Generally, a negotiation mechanism has three main components [239, 233]: 1) the set of possible outcomes; 2) negotiation protocol that defines the rules for the negotiation; 3) the negotiation strategies of the participants that, given the set of possible outcomes and the negotiation protocol, define how the agents decide about their actions during the negotiation. There exist a plethora of negotiation protocols in the literature that differ in terms of, e.g., the number of participants, the number of issues, the way multiple issues are handled (e.g., negotiated independently in parallel, as a package deal, or sequentially), time constraints, the type of setting (cooperative, competitive or something in between), information state (complete, incomplete), and whether limited communication or computation is considered.

We do not believe that there exists one universally correct/suitable approach on negotiation in the computing continuum. Computing continuum poses many challenges that affect the details of the deployed negotiation mechanism. For example, negotiation over administrative boundaries can be considered much more competitive than negotiation inside the same administrative domain, as the agents inside the same administrative domain ultimately aim to reach a globally optimal system state within that domain. Further, agents higher in the hierarchy of the computing continuum have more resources for their negotiation mechanism than the agents on the lower levels. However, any deployed negotiation

mechanism in the computing continuum should address the problem of incomplete information, enforce the loose coupling and autonomy of the agents, and be secure due to the openness of the system and the self-interested nature of the agents. Security requires designing trust and reputation mechanisms so that the self-interested agents can better assess with whom to interact and are incentivized to be truthful in their interactions (see e.g. [31, pp. 381-414]).

In the scope of this article, due to the overwhelming amount of negotiation related work, we are focusing on giving a brief overview of two research themes that we see particularly important for negotiation in the computing continuum regardless of the actual details of the negotiation situation: opponent modelling and argumentation.

A simple way to handle the incomplete information of the participants in the negotiation would be to require them to share their private information at the start of the negotiation. However, such an approach is not feasible in the computing continuum due to the agents' self-interested nature: other agents may try to exploit an agent that reveals its private information. A better approach is to try to learn attributes of other agents (such as preferences) based on the information derived from the exchanged offers, that is, learning opponent models [237]. Opponent modelling is an essential part of an agent's adaptation capability because it enables an agent to change its behavior based on what it believes about its opponents. Adaptation is a crucial requirement for implementing negotiation in real-life settings, as, for example, van Bragt and La Poutré [240] have shown that non-adaptive agents can be exploited after collecting a sufficient history of their negotiation behavior.

Baarslag et al. [237] provide a comprehensive survey of opponent modelling in bilateral negotiation. Their main findings are that the used learning methods for opponent modelling fall generally into four categories (Bayesian learning, non-linear regression, kernel density estimation and ANN based learning), and that the models aim to learn some combination of four opponent attributes (acceptance strategy, deadline, preference profile and bidding strategy). Bayesian learning and non-linear regression are more adept for online learning as their estimates improve incrementally, whereas kernel density estimation and ANN based learning are more suitable when a training phase with sufficiently large negotiation history can be conducted.

What type of opponent modelling approaches for negotiation could be the most suitable and efficient for computing continuum is an open research question. Comparing current propositions is difficult due to lack of evaluation benchmarks for negotiation, meaning that many approaches are validated in the authors' own settings [237]. Further, as pointed out by Baarslag et al. [237], many approaches make assumptions that can be very limiting. For example, non-linear regression based approaches assume that the opponent's bidding strategy follows a known decision function with unknown parameters. In addition, opponent modelling in some settings seems to be underexplored, such as estimating an opponent's acceptance strategy in multi-issue negotiation.

Argumentation-based negotiation enables sharing meta-information during negotiation in addition to the bids or offers [241, 242]. The meta-information is shared in the form of arguments that try to affect the opponent's mental state. Rather than just alternating between proposals and counterproposals until a point of agreement or non-agreement is reached, argumentation tries to help the agents to understand why an offer was rejected. If an opponent rejected the offer due to not knowing some alternative way to reach its goals, an agent can try to convince the opponent to accepting the offer by providing this alternative way for reaching the goals. Argumentation can help the agents to find an agreement quicker and increase the likelihood of reaching an agreement in the first place, as well as increase the quality of the final outcome [241, 243, 242].

Whether and to which extent argumentation can be utilized in the computing continuum is an open research question. Even though argumentation-based negotiation is a very compelling approach for efficient negotiation, its current research state involves many open issues that hinder its applicability in open systems such as the computing continuum.

Argumentation aims to influence the mental state of another agent (e.g., its beliefs or goals) and requires a protocol for enabling the argumentative discussion. Knowledge state of an agent as well as the arguments must be expressed in a formal way in order to allow agents to reason about them. Currently, to the best of our knowledge, there are no consensus on how to represent arguments, which is crucial for enabling interoperability of agents in an open system. Further, it seems that the research on argumentation-based negotiation has strongly relied on the assumption that the agents are benevolent, without an incentive to be dishonest or withhold arguments for their own advantage. Such an assumption obviously does not hold in the computing continuum. To account for such a strategic behavior of agents in argumentation, a careful design of argumentation mechanisms is required [244].

11) Learning to Communicate: Communication is an essential part of a MAS. In the computing continuum, agents must exchange information with each other to ease the partial observability, make coordinated decisions and reach agreements. Enabling the communication between agents does not necessarily mean that agents must use some sort of predetermined communication protocol. Instead, a particularly interesting research question is whether the agents could learn a communication protocol while acting in the environment.

Some MADRL methods have been proposed for learning a communication protocol. Most notably, Foerster et al. [105] propose Reinforced Inter-Agent Learning (RIAL) and Differentiable Inter-Agent Learning (DIAL), each of which uses the CTDE paradigm and assumes a fully cooperative setting. Both methods use an ANN that outputs the agent's Q-values for environmental actions, as well as for communication actions. RIAL is based on Deep Recurrent Q-Networks and can also use parameter sharing, that is, a single network is learned whose parameters are used by all agents. DIAL, on the other

hand, directly passes gradients via the communication channel during training, the purpose of which is to provide feedback about the communication actions and speed up the learning.

Overall, the research on learning a communication protocol in complex and open real-life settings is at scarce. A complex setting means the type of environment where the number of agents can be large, and the agents are not fully cooperative, that is, they have their own individual utilities. Such a setting would require more consideration from the agent to whom send the messages rather than just broadcasting them to everyone. Further, a mixed setting requires the inclusion of trust and reputation mechanisms, because an agent cannot blindly trust every message it gets, and agents should be incentivized to be truthful in their communication.

An open setting means a type of environment where agents can leave and enter at any time. In an open setting, an agent needs to quickly learn the protocol the other agents have already learned. Moreover, the following questions remain open: 1) how the entry of new agents to the learning procedure will adapt the way older agents understand the messages, and 2) how older agents that learned a protocol at some point, then dropped out and joined in after the protocol had already evolved will be able to quickly adjust to the new meanings.

AI approaches for learning a protocol have also other unresolved issues that hinder their suitability for the computing continuum. They currently confine agents to using the same type of learning methods and models, which works when the MAS is implemented by the same people, but not when it comes to learning protocols between agents implemented by different groups of people. Further, they can be too resource consuming for the agents in the computing continuum, such as is the case with approaches based on using Deep Neural Networks (DNNs). Hence, alternative approaches should also be studied.

It is good to note that the traditional MAS protocols are not suitable for communication in the computing continuum, because they are specified as message flows without reference to the meaning of the messages, and they rely on a small set of primitives (i.e., message types) with unique definitions [31, pp. 108-114]. These definitions do not usually provide enough flexibility to accommodate the communication requirements of different applications, meaning that the designers must come up with ad hoc methods to convey different types of meanings inside the messages. Consequently, agents become tightly coupled. In addition, the focus on exchanging messages in a particular order can be seen as overconstraining an agent's interactions, and thus, violating its autonomy.

Alternative solutions to agent communication in the computing continuum could come from commitment-based protocols [245, 246, 31, pp. 118-121], where agents communicate with each other by creating and manipulating commitments. Commitment-based protocols are defined by stating the meaning of the application-specific message types in terms of the commitments that they establish between the communicating agents. These type of protocols can enforce loose coupling and autonomy of the agents, as they do not impose a specific ordering on the message flow between two parties, there is no need to hardcode into the agents how each application-specific message type should be interpreted as this interpretation can be derived from the protocol specification, and no agent is expected to provide more than it has committed to. In other words, commitments provide an underlying higher level of abstraction that can be used to provide the semantics of an application-specific message type.

C. Other approaches

1) Generative Adversarial Networks: Generative models can be defined as models that take a training data set drawn from some distribution, and then learn to represent an estimate of that distribution [247]. Generative models can offer a lot for achieving an autonomous computing continuum. They are able to represent and manipulate high-dimensional probability distributions, which is an important ability for edge agents that are faced with high-dimensional state and action spaces. Generative models can simulate possible future states, for example, by learning a conditional distribution over future states, given the current state and possible actions. An agent can then query this model to find an action that will most likely result in a desired state (see, e.g., [248]). Hence, generative models enable learning in an imaginary environment, where an agent can 'imagine' the consequences of its possible actions, and mistaken actions cause no harm to the agent. Generative models can also be used for guiding exploration of an agent and for inverse reinforcement learning [247, 249].

Besides reinforcement learning related benefits, generative models are also useful in supervised learning settings. For example, they can be used to augment the training data [250], which is particularly beneficial in the case of ANN models that require substantial amounts of training data to generalize well. In distributed learning, they can also mitigate the non-IID problem through data augmentation [137]. As another example, data synthesizing combined with self-supervised learning can help in automatically classifying service traffic data at the edge of the network without requiring any pre-knowledge of the available services or human effort for data annotation [251]. In other words, synthesizing data can help us in reliably identifying emerging patterns, which is a crucial skill in the dynamic computing continuum.

Generative Adversarial Networks (GANs) are a type of generative models that have achieved very promising results in domains such as image generation, natural language, time-series synthesis, digital pathology and security [252, 253]. The basic idea of a GAN is to set up a zero-sum game between two players [247]. The first player is called a generator, which creates samples that are supposed to come from the same distribution as the training data. The second player is a discriminator that divides its input into two classes: fake or real. The ultimate goal is to reach an equilibrium, where the generator is able to create samples that are indistinguishable from the real samples.

The generator learns an implicit presentation of the density function, allowing a user to draw samples from it. GANs have

several advantages over other types of generative models, such as they are not dependent on Markov chains, they can generate samples in parallel, and there is very few restrictions for the generator design. At the same time, GANs introduce a new type of challenging disadvantage: training a GAN is equal to finding an NE of a game [247].

More formally, the generator is represented by a function G that takes latent variables z as an input and has θ_G as its parameters. Function G maps z into observed variables x. The discriminator is defined by a function D that takes x as an input, has θ_D as its parameters, and outputs a single scalar value that indicates whether the input was real or fake. Both functions must be differentiable with respect to their inputs and parameters, and they are typically implemented as deep neural networks [247].

Both the generator and the discriminator have a cost function that is dependent on both their own parameters as well as those of the other player. The discriminator tries to minimize $J_D(\theta_D, \theta_G)$ while only controlling θ_D, and the generator tries to minimize $J_G(\theta_D, \theta_G)$ while only controlling θ_G. Because of this dependence on the parameters of both and the ability to control only one's own parameters, this minimization problem can be seen as a zero-sum game, the solution of which is a NE.

The original version of a GAN, proposed by Goodfellow et al. [254], uses the following value function for the minimax game

$$\min_G \max_D \; V(D,G) = \mathbb{E}_{x\sim p_{data}}[\log D(x)] + \mathbb{E}_{z\sim p_z}[\log(1 - D(G(z)))], \qquad (12)$$

where p_{data} is the distribution of the real data, p_z is the distribution of the latent variables, and $D(x)$ presents the probability that x comes from the real data. With this value function, D is trained to maximize the probability of assigning correct label to both real and fake samples, that is, to maximize $\log D(x) + \log(1 - D(G(z)))$, while G is simultaneously trained to minimize $\log(1 - D(G(z)))$. In practice, the game is implemented by alternating between optimizing D and G, and the objective of the generator is changed from minimizing $\log(1 - D(G(z)))$ to maximizing $\log D(G(z))$ in order to avoid vanishing gradients [254].

The main challenges plaguing the training of GANs are non-stable training, mode collapse and vanishing gradients [252]. Non-stable training stems from finding the equilibrium to the game between the players. While each player may reduce its loss on its individual update, that update can simultaneously cause the other player's loss to go up, resulting in a cycle where the players repeatedly undo each other's progress [247]. Mode collapse refers to the phenomenon where several different latent inputs to the generator result in the same output, reducing the diversity of the generator outputs. Vanishing gradients slow down or completely stop the training of the generator. This is a major issue especially at the beginning of the training, when the discriminator is able to separate the fake inputs from the real ones with high confidence, which saturates the loss of the generator [254].

A plethora of methods and heuristics have been proposed to mitigate at least some of the aforementioned issues in GAN training. One popular approach is to modify the loss functions of the discriminator and generator [252]. For example, Arjovsky et al. [255] proposed Wasserstein GAN (WGAN) to mitigate all three issues of non-stability, mode collapse and vanishing gradients. WGAN is based on minimizing Earth Mover's (EM) distance between the real data distribution and the generator data distribution, rather than the Jensen-Shannon divergence used in the original GAN [254]. Using EM as loss measure has shown significant improvements in the stability of the training, and it has the benefit of being a more meaningful metric, because it is a measure of distance in a space of probability distributions (i.e., it converges to zero as the distributions get close to each other) [255, 252, 256].

Other mitigation approaches include changing the architecture of the deep neural networks, introducing several generators or discriminators (to prevent especially the mode collapse), and modifying the optimization algorithm [252]. A major problem in many proposed heuristics and methods is the lack of solid theoretical foundation, which means that it depends on the context whether a particular approach is helpful [247, 252]. For example, a study by Lucic et al. indicate that it may be more worthwhile to properly tune the hyperparameters of the models rather than turn to alternative modified methods [257].

Furthermore, how to objectively compare the performance of different GANs is largely an open question, as it is not undisputable nor trivial what metrics to use for the quantitative evaluation of the generated samples [257]. It is also good to note that the majority of GAN related work has been done in image-based applications (see e.g. [256]). Hence, applying GANs in other types of problems appearing in the computing continuum environment is an open and promising research direction.

2) Few-shot learning: Typically a lot of training data is required to train an ML model, particularly when it comes to training an ANN model. However, collecting such large amounts of data is burdensome, and, in complex and dynamic environments, it is practically impossible to obtain a diverse data set that would have enough samples of every possible pattern in the data. In addition, completely new patterns can emerge at any moment due to the evolving nature of the environment, such as unprecedented attack techniques in the huge volume of network traffic. Furthermore, the closer we get to the device level in the computing continuum, the more resource-constrained the agents become, meaning that they can have the capability to participate to model training only with small amounts of data. Hence, in the dynamic computing continuum, being able to quickly adapt to a new task based on a few samples and training iterations is essential.

Few-Shot Learning (FSL) is an ML paradigm that is concerned with developing methods that are able to learn quickly from a limited number of examples. Currently, FSL has been

mainly focused on supervised classification and regression problems [258, 259]. There are also works that have developed FSL methods for reinforcement learning, with the aim of learning a policy based on only a few trajectories of state-action pairs [260].

In supervised ML, the core issue of FSL can be illustrated through error decomposition [258, 261]. Given a parameterized function $f \in \mathcal{F}$ that maps an input to an output, $\mathcal{F}$ being the space of all functions, we want to minimize the expected risk $E(f)$, which is the expected value of loss given the training data distribution. However, since the training data distribution is unknown, instead of $E(f)$, we aim to minimize the empirical risk $E_N(f)$, which is the average of losses over the training data of N samples. If we mark $\hat{f} = argmin_f E(f)$ as the function that minimizes the expected risk, $f^* = argmin_{f \in \mathcal{F}} E(f)$ as the function in $\mathcal{F}$ that minimizes the expected risk, and $f_N = argmin_{f \in \mathcal{F}} E_N(f)$ as the function in $\mathcal{F}$ that minimizes the empirical risk, the total error between $E(\hat{f})$ and $E(f_N)$ can be decomposed as

$$\mathbb{E}[E(f_N) - E(\hat{f})] = \mathbb{E}[E(f^*) - E(\hat{f})] + \mathbb{E}[E(f_N) - E(f^*)], \quad (13)$$

where the expectation is over the random choice of training data. Here, the approximation error $\mathcal{E}_{app} = \mathbb{E}[E(f^*) - E(\hat{f})]$ measures how close the functions in $\mathcal{F}$ can approximate the optimal function $\hat{f}$, and the estimation error $\mathcal{E}_{est} = \mathbb{E}[E(f_N) - E(f^*)]$ measures the effect of minimizing the empirical risk $E_N(f)$ instead of the expected risk $E(f)$ within $\mathcal{F}$ [261].

In general, increasing the number of training examples reduces $\mathcal{E}_{est}$, meaning that when N is sufficiently large, f_N can provide a good approximation of f^* [258]. However, in FSL, the number of available examples is small, which makes the empirical risk minimizer f_N unrealiable, as it may be far from providing a good approximation. To alleviate this problem, prior knowledge must be leveraged in FSL. In their FSL survey, Wang et al. [258] categorize FSL methods into three categories based on which aspect is improved using prior knowledge: data, model or algorithm.

Data based methods augment the training data so that standard ML models can be used on the augmented data. Augmentation can happen by transforming data from the training data set itself, from a weakly labeled / unlabeled data set, or from similar data sets [258]. In the scope of this article, these methods will not be elaborated further due to their ad hoc nature; they are tailored for a specific data set and cannot be easily applied to other data sets. Furthermore, existing methods are mostly concerned with data sets consisting of images [258].

Model based FSL methods aim to constrain the space of all functions $\mathcal{F}$ to a smaller function space by utilizing some prior knowledge [258]. This prior knowledge can come from other related tasks (multitask learning) so that the some parameters are directly shared among the tasks, or parameters of different tasks are encouraged to be similar with each other through regularization. Another way is to learn so called embedding function based on prior knowledge and/or task specific knowledge. This embedding function maps each sample to a lower-dimensional space, where similar samples are closer together while dissimilar samples are more easily differentiated. A smaller function space can be constructed for this lower-dimensional space. Finally, generative modelling can be utilized in a case where the observed data is assumed to follow a certain distribution. The shape of this distribution is constrained by utilizing a prior distribution learned from other related tasks.

Algorithm based methods use prior knowledge to alter the search strategy for the parameters of the best function in $\mathcal{F}$. This can happen in two ways: either prior knowledge (existing parameters or meta-learned parameters) offers a good initialization for the search, or prior knowledge (a set of related tasks) is used to learn a meta-learner (optimizer) that takes in the loss of the FSL learner and directly outputs the update to the task-specific parameters of the FSL learner [258].

In model and algorithm based FSL methods, meta-learning techniques offer an interesting *learning to learn* aspect, where some generalizable information is learned across a variety of similar tasks, and then this information can be utilized in solving new tasks with only a small number of training samples. We will focus on meta-learning based FSL methods, because these methods do not require hand-crafting a learning algorithm for a specific task, and they allow a quick adaptation to new information, which is crucial for tasks in a dynamic computing continuum. Furthermore, the focus is mainly on methods that can be applied to different types of learning problems, namely to both supervised and reinforcement learning problems.

A representative example of FSL methods that utilize meta-learning techniques is the Model-Agnostic Meta-Learning (MAML) algorithm, proposed by Finn et al. [262]. MAML can be used with any model trained with gradient descent and can be adapted to different learning problems, such as supervised classification and regression, as well as policy-gradient reinforcement learning. MAML meta-learner does not require any additional parameters or place any constraints on the model architecture. It can also be used with a variety of loss functions.

In MAML, the key idea is to learn globally optimal initial model parameters across a variety of learning tasks so that the parameters can then be quickly adapted to a new task by computing only a small number of gradient descent steps with a few training samples from that new task. For their algorithm, Finn et al. introduce a generic notion of learning tasks, i.e., each task is defined as $\mathcal{T} = \{\mathcal{L}(x_1, a_1, \ldots, x_H, a_H), q(x_1), q(x_{t+1}|x_t, a_t), H\}$, where x are observations and a are outputs, $\mathcal{L} \rightarrow \mathbb{R}$ is a loss function that provides task-specific feedback, $q(x_1)$ is a distribution over initial observations, $q(x_{t+1}|x_t, a_t)$ is a transition distribution, and finally, H is an episode length. This generic task definition is applicable to both supervised and reinforcement learning settings.

The meta-learning objective for finding the global model

parameters θ is expressed as

$$\min_{\theta} \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'}) = \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)}). \quad (14)$$

Here, $\mathcal{T}_i$ is a task sampled from the distribution over tasks $p(\mathcal{T})$, $\mathcal{L}_{\mathcal{T}_i}$ is the loss function of task $\mathcal{T}_i$, f is the parameterized model, and $\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ is the updated parameter vector after one gradient descent update on task $\mathcal{T}_i$, α being the step-size. More than one adaptation steps can be done on a task, but for the notational simplicity only one update is considered. The idea of this objective is to optimize the global model parameters θ so that a small number of updates with only a few samples on a new task will result in significant improvements in the task's loss function, i.e., result in a fast adaptation.

One deficiency in MAML is that it does not consider the ambiguity in the model function, i.e., it only learns a single function. Modelling this ambiguity is particularly important for few-shot learning tasks, because the few samples in the task may not provide enough information to learn an unambiguous model even if the initial model parameters learned from the related tasks would be the best possible ones. Hence, Finn et al. extend MAML approach to probabilistic MAML [263], which learns a prior distribution over the global model parameters. As a result, multiple possible models can be sampled from the distribution before adapting to a new task with gradient descent, which allows to reason about the distribution of the possible model functions, i.e., model the uncertainty in the parameters for the task.

Another deficiency in MAML is that it only learns an initialization; updating relies on SGD where the learning rate is manually set. Li et al. [264] propose Meta-SGD algorithm that extends MAML by meta-learning the learning rate α in addition to the initialization θ. In other words, α is now a vector of the same size as θ that determines both the update direction and learning rate, and the meta-objective in Eq. (14) is now minimized with regard to θ and α. Furthermore, the ANN architecture used in MAML is fixed. To this end, Elsken et al. [265] propose MetaNAS that is applicable to gradient-based meta-learning FSL methods. The idea of MetaNAS is to meta-learn an ANN architecture in addition to the initialization parameters, which allows the adaptation to a task-specific ANN architecture.

Meta-learners trained over a variety of tasks may suffer from overfitting to some tasks, which results in poor generalization on new tasks that deviate significantly from the meta-training tasks. To address this issue, Jamal and Qi [266] propose a Task-Agnostic Meta-Learning (TAML) algorithm, which learns unbiased initial model parameters by preventing the initial model from over-performing on any of the meta-training tasks. This prevention is done by minimizing an inequality measure over the losses of the sampled meta-training tasks. They build their approach on top of the MAML algorithm, meaning that the chosen inequality measure is added as a regularizer to the meta-learning objective presented in Eq. (14). Nevertheless, their idea of unbiased initial parameters is also applicable to many other meta-learning algorithms besides MAML [266].

Gradient-based meta-learning approaches that aim to find a shared starting point for task-specific adaptation, such as MAML, can suffer from poor generalization in high dimensional parameter spaces. This is due to the very small amount of data that is used to calculate the gradients, which often results in overfitting. Consequently, MAML is limited to using very simple, shallow ANN architectures. To mitigate this issue, Rusu et al. [267] propose Latent Embedding Optimization (LEO) that decouples the gradient-based adaptation from the high-dimensional parameter space. This is achieved by learning a data-dependent encoder that maps input data into a low-dimensional, stochastic latent space, in which the task-specific adaptation is performed with gradient descent. A decoder, which has been learned alongside the encoder, works as a parameter generator that maps from the latent space to the high-dimensional parameter space.

In other words, rather than learning a single global initialization, as in MAML, LEO learns globally optimized parameters for encoder-decoder architecture, which can be used to approximate a data-dependent conditional probability distribution over the high-dimensional parameter space. This allows to have a task-specific initialization point for the adaptation, as well as enables the modelling of the uncertainty in the model parameters, as in probabilistic MAML. The encoder-decoder architecture presented by Rusu et al. is specifically tailored for supervised classification models, but they show that the approach is applicable to supervised regression; application to reinforcement learning is left as future work.

In general, FSL methods have many open issues that hinder their real-life applicability. One very fundamental issue in meta-learning or multi-task learning based FSL methods is the task-relatedness. Meta-training is conducted over thousands of batches of FSL tasks with the assumption that the meta-training tasks relate to the new tasks on a level that leads to performance improvement in the new FSL task after adaptation. Negative transfer, where the performance on a new task deteriorates after adaptation, should be avoided. With regard to reducing the number of meta-training batches, it may be worthwhile to study how to combine transfer learning from pre-trained DNN models with meta-learning [268].

Majority of FSL related work has been done in the field of image classification [258, 259, 269], and the performance of these methods is far behind the traditional supervised methods [268, 269]. FSL methods usually also rely on unrealistic assumptions, such as that the samples and their labels in the support set for the FSL training are accurate and reliable, and that there exist a huge, properly labeled data set from which to extract prior knowledge for the FSL task. Studying FSL more in the context of unsupervised and semi-supervised learning would be beneficial for real-life applications (see e.g. [270, 259]).

FSL methods for supervised classification often assume that the model for the FSL task needs to only reason about the

novel classes in the support set and not about any previously seen classes. This phenomenon is referred to as catastrophic forgetting [259]. Especially in computing continuum, where new class concepts can appear in a dynamic manner, forgetting the old classes should not happen. For example, a security model trained to classify network traffic should not forget about previous attack classes when it is adapted to recognize new ones.

Many of the existing FSL methods also assume that the input-output data distribution in the FSL tasks is the same as in the previously seen tasks and data sets. This is an unrealistic assumption in dynamic environments where the conditions under which data is produced can change, resulting in shifts in the input-output distribution. These kind of situations require methods that can account for cross-domain adaptation [271, 259].

Finally, how to combine FSL with distributed learning in the computing continuum environment is a significant and interesting research direction with regard to our vision, because assuming that each agent in a computing continuum system would have thousands of labeled samples for collaborative model training is unrealistic. Some preliminary work that combines FSL with FL has been done. For example, Zhao et al. [272] propose a MAML based federated FSL method for supervised classification, but they focus on a cross-silo FL setting, where the data is siloed across a small number of participants so that the participants have mutually isolated classes of samples. In cross-silo FL [143], the participants are usually institutions or organizations, and they are assumed to be highly available during training.

Taking a step towards cross-device FL [143], where there can be a massive number of participants with varying availability, Fan and Huang [273] propose an adversarial learning procedure for federated FSL in supervised classification settings. The method aims to find an initialization for a global model consisting of a feature encoder and a classifier so that it quickly adapts to a new FSL task with unseen classes. However, their adversarial learning method can be computationally too heavy for resource-constrained participants, and their experiments only consider up to 30 participants, while not accounting for, e.g., stragglers or domain shift (i.e., the data for training and testing comes from the same data set in their experiments).

The combination of FSL and distributed learning is very uncharted. Preliminary work considers only supervised classification in a centralized FL setting with a small number of participants. A lot more research is required to assess the performance, scalability and efficiency of FSL in a variety of distributed learning settings, as well as the effects of a large number of heterogeneous participants, non-IID data, domain shifts, and possible decentralization of the learning.

3) Self-supervised learning: Supervised learning requires that every sample has a corresponding label. Obtaining a labeled data set for ANN training can require an expensive and time-consuming manual data labeling process, which usually also limits the size of the data set. In computing continuum, where the training data is distributed among several nodes, it is unrealistic to assume that each node has a properly labeled, high-quality data set with substantial number of samples. Furthermore, the question of who is annotating the data in a geographically distributed computing continuum is not a trivial question. Any extensive attempts to manually label data in such an environment are already made infeasible by the enormous amount of data that a distributed, dynamic system can generate every day at a fast pace. Hence, an essential question arises: how can the large amounts of unlabeled data be utilized in model training?

Self-supervised learning (SSL) is a learning paradigm that tries to tackle the data annotation issue by utilizing some type of automatic supervision signal extracted from the unlabeled data. This usually means learning a representation from the unlabeled data (often called a pretext task) that can be subsequently used in a model trained with a smaller amount of labeled data (often called a downstream task). SSL has been most commonly studied in the context of representation learning for image based and natural language processing applications, and more recently in an increasing amount for applications using graph data [274, 275, 276].

Liu et al. [274] categorize self-supervised representational learning into three categories: generative, contrastive and adversarial. The core idea behind each class of methods is to obtain an excellent pre-trained feature extractor for the downstream task, which requires designing a proper, downstream task related objective for the pretext task. Generative SSL methods use reconstruction loss to train a encoder-decoder architecture, contrastive methods use a a similarity metric based, contrastive loss in the latent space to train an encoder, and adversarial methods use a loss based on distributional divergence to train a generator for creating fake samples and a discriminator for distinguishing them from the real ones.

Generative methods have been popular in classification and generation because they are able to recover the original data distribution without assumptions for downstream tasks [274]. However, these methods have been inferior to contrastive methods in image classification. In addition, the reconstruction loss is often based on maximum likelihood, which means that the data distribution is modelled on a point-wise level (e.g., pixels in images), which is a low-level abstraction and makes the model very sensitive to rare samples.

Contrastive methods assume that the downstream applications are classification tasks. They have been very successful in image classification tasks, but their suitability for classification tasks in other domains such as natural language processing has not been extensively studied [274]. Furthermore, many constrastive methods require so called negative sampling, because they are often based on distinguishing between representations coming from augmentations of the same data point (positive samples) and those of other data points (negative samples). What type of negative sampling scheme to use is an essential question, and the role and necessity of negative sampling in contrastive methods is an open issue [274, 277].

Adversarial methods are mainly based on GANs. However, GANs only learn an implicit latent representation and

thus, cannot be directly used in self-supervised representation learning. Some methods that aim to utilize GANs in representational learning try to extract the latent representation by replacing the generator with an adversarial autoencoder [278] or by treating the generator as a decoder and adding an encoder to learn a mapping from real samples to the latent space (i.e., a converted generator) [279]. Another way to use GANs for representational learning is to train them to recover the whole output based on a partial input (e.g., image completion [280]) rather than training them to recover the whole output based on a latent input. GANs have been mainly utilized in vision based applications [274]. Extending them to a wider variety of self-supervised applications is an interesting research direction due to their ability to model high-level abstractions.

It is good to note that self-supervision does not remove the need for labeled data, but it can help to reach better performance compared to supervised learning when a small amount of labeled data is available. Newell et al. have shown in vision based tasks that SSL is not able to reach higher accuracy than supervised learning when there is enough labeled data available, but it can help to reach a better accuracy in low data regimes [281]. Even though SSL cannot help in improving performance beyond supervised methods with large amounts of labeled data, it still can help in model robustness and uncertainty estimation, as shown by Hendrycks et al. [282]. They find that SSL can make models more robust to adversarial examples, label corruption, and common input image corruptions, as well as better at detecting out-of-distribution samples.

As noted, SSL has been mainly studied in representational learning for image classification in centralized training settings. Applying SSL in a wider variety of problems appearing in the computing continuum environment is an important research direction, because acquiring large amounts of properly labeled data in a large distributed system is practically impossible. Applying SSL in computing continuum requires research on how to effectively enable distributed SSL. Some preliminary work that combine SSL with FL exist (see, e.g., [283, 284, 285]). Most notably, Saeed et al. [284] propose an approach that uses wavelet transform to learn representations from unlabeled sensor inputs. Their pretext task is a contrastive method where a deep temporal neural network is trained to determine if a given pair of a signal and its complementary view (a scalogram extracted with wavelet transform) align with each other (i.e., a binary classification problem). For training this network in a distributed manner, they adopt a vanilla FL setting, assuming a small number of participants at each training round and dividing the training data across participants randomly. Subsequently, more research on distributed SSL is required to assess, among other things, the effects of a large number of heterogeneous participants with limited resources, non-IID data, and complete decentralization of the training.

There are many other challenges besides the distribution of self-supervised training that hinder the use of SSL methods in autonomous computing continuum, some of the major ones being the following: establishing theory that could help in avoiding empirical misconceptions and adapting solutions from vision based domains to other domains; automating the design of pretext tasks, which in its current form seems to be done in a very ad hoc manner and relies heavily on human effort; and model adaptation when data distribution shifts.

VII. Conclusion

In this article, we studied orchestration in the device-edge-cloud continuum, and focused on *AI for edge*, that is, the AI methods used in orchestration. We claimed that to support the constantly growing requirements of intelligent applications in the device-edge-cloud computing continuum, resource orchestration needs to embrace edge AI and emphasize local autonomy and intelligence.

To justify the claim, we provided a general definition for continuum orchestration, and looked at how current and emerging orchestration paradigms are suitable for the computing continuum. We described certain major emerging research themes that may affect future orchestration, and provided an early vision of an orchestration paradigm that embraces those research themes.

Finally, we surveyed current key edge AI methods and looked at how they may contribute into fulfilling the vision of future continuum orchestration. We identified methods for distributed and secure machine learning, decision making and negotiation, described their fundamentals, and discussed their strengths and weaknesses in relation to the challenges in the computing continuum.

Acknowledgment

The authors would like to thank the researchers and alumni at the Center for Ubiquitous Computing (UBICOMP) at the University of Oulu, and Distributed Systems Group (DSG) at TU Wien, for the discussions and insights during the writing of this manuscript. In particular, dr. Víctor Casamayor Pujol (DSG) provided valuable feedback on the structure of the computing continuum, of which the authors are grateful.

References

[1] Zeyi Tao et al. “A Survey of Virtual Machine Management in Edge Computing”. In: *Proceedings of the IEEE* 107.8 (2019), pp. 1482–1499. DOI: 10.1109/JPROC.2019.2927919.

[2] Tarik Taleb et al. “On Multi-Access Edge Computing: A Survey of the Emerging 5G Network Edge Cloud Architecture and Orchestration”. In: *IEEE Communications Surveys Tutorials* 19.3 (2017), pp. 1657–1681. DOI: 10.1109/COMST.2017.2705720.

[3] Ibrahim Afolabi et al. “Network slicing and softwarization: A survey on principles, enabling technologies, and solutions”. In: *IEEE Communications Surveys and Tutorials* 20.3 (2018), pp. 2429–2453. ISSN: 1553877X. DOI: 10.1109/COMST.2018.2815638.

[4] Allan Dafoe et al. *Open Problems in Cooperative AI*. 2020. arXiv: 2012.08630.

[5] Allan Dafoe et al. “Cooperative AI: machines must learn to find common ground”. In: *Nature* 593.7857 (2021), pp. 33–36. ISSN: 14764687. DOI: 10.1038/d41586-021-01170-0.

[6] Konstantinos Samdanis and Tarik Taleb. “The Road beyond 5G: A Vision and Insight of the Key Technologies”. In: *IEEE Network* 34.2 (2020), pp. 135–141. DOI: 10.1109/MNET.001.1900228.

[7] Yongli Zhao et al. “Edge Computing and Networking: A Survey on Infrastructures and Applications”. In: *IEEE Access* 7 (2019), pp. 101213–101230. DOI: 10.1109/ACCESS.2019.2927538.

[8] Behnaam Aazhang, Petri Ahokangas, Hirley Alves, et al. *Key drivers and research challenges for 6G ubiquitous wireless intelligence (white paper)*. Ed. by Matti Latva-Aho, Markku Juntti, et al. 1st ed. Oulu, Finland: 6G Flagship, University of Oulu, 2019, pp. 1–36. ISBN: 9789526223537.

[9] Shuiguang Deng et al. “Edge Intelligence: The Confluence of Edge Computing and Artificial Intelligence”. In: *IEEE Internet of Things Journal* 7.8 (2020), pp. 7457–7469. DOI: 10.1109/JIOT.2020.2984887.

[10] Lauri Lovén et al. “EdgeAI: A vision for distributed, edge-native artificial intelligence in future 6G networks”. In: *The 1st 6G Wireless Summit*. Levi, Finland, 2019, pp. 1–2.

[11] Ella Peltonen et al. “6G White Paper on Edge Intelligence”. In: *CoRR* abs/2004.14850 (2020). arXiv: 2004.14850. URL: https://arxiv.org/abs/2004.14850.

[12] Dianlei Xu et al. “A Survey on Edge Intelligence”. In: *CoRR* abs/2003.12172 (2020). arXiv: 2003.12172. URL: https://arxiv.org/abs/2003.12172.

[13] Wei Yang Bryan Lim et al. “Federated Learning in Mobile Edge Networks: A Comprehensive Survey”. In: *IEEE Communications Surveys Tutorials* 22.3 (2020), pp. 2031–2063. DOI: 10.1109/COMST.2020.2986024.

[14] Jihong Park et al. “Wireless Network Intelligence at the Edge”. In: *Proceedings of the IEEE* 107.11 (2019), pp. 2204–2239. DOI: 10.1109/JPROC.2019.2941458.

[15] Jihong Park et al. “Communication-Efficient and Distributed Learning over Wireless Networks: Principles and Applications”. In: *Proceedings of the IEEE* 109 (5 May 2021), pp. 796–819. DOI: 10.1109/JPROC.2021.3055679.

[16] Xiaofei Wang et al. “Convergence of Edge Computing and Deep Learning: A Comprehensive Survey”. In: *IEEE Communications Surveys Tutorials* 22.2 (2020), pp. 869–904. DOI: 10.1109/COMST.2020.2970550.

[17] Zhi Zhou et al. “Edge Intelligence: Paving the Last Mile of Artificial Intelligence With Edge Computing”. In: *Proceedings of the IEEE* 107.8 (2019), pp. 1738–1762. DOI: 10.1109/JPROC.2019.2918951.

[18] Cong Hao et al. “Enabling Design Methodologies and Future Trends for Edge AI: Specialization and Codesign”. In: *IEEE Design Test* 38.4 (2021), pp. 7–26. DOI: 10.1109/MDAT.2021.3069952.

[19] Quoc-Viet Pham et al. “A Survey of Multi-Access Edge Computing in 5G and Beyond: Fundamentals, Technology Integration, and State-of-the-Art”. In: *IEEE Access* 8 (2020), pp. 116974–117017. DOI: 10.1109/ACCESS.2020.3001277.

[20] Xiaolong Xu et al. “Artificial intelligence for edge service optimization in Internet of Vehicles: A survey”. In: *Tsinghua Science and Technology* 27.2 (2022), pp. 270–287. DOI: 10.26599/TST.2020.9010025.

[21] Yuanming Shi et al. “Communication-Efficient Edge AI: Algorithms and Systems”. In: *IEEE Communications Surveys Tutorials* 22.4 (2020), pp. 2167–2191. DOI: 10.1109/COMST.2020.3007787.

[22] Riccardo Guerzoni et al. “Analysis of end-to-end multi-domain management and orchestration frameworks for software defined infrastructures: an architectural survey”. In: *Transactions on Emerging Telecommunications Technologies* 28.4 (2017). e3103 ett.3103, e3103. DOI: https://doi.org/10.1002/ett.3103. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/ett.3103. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/ett.3103.

[23] Klervie Toczé and Simin Nadjm-Tehrani. “A Taxonomy for Management and Optimization of Multiple Resources in Edge Computing”. In: *Wireless Communications and Mobile Computing* 2018 (June 2018), p. 7476201. ISSN: 1530-8669. DOI: 10.1155/2018/7476201. URL: https://doi.org/10.1155/2018/7476201.

[24] Nathan F. Saraiva de Sousa et al. “Network Service Orchestration: A survey”. In: *Computer Communications* 142-143 (2019), pp. 69–94. ISSN: 0140-3664. DOI: https://doi.org/10.1016/j.comcom.2019.04.008. URL: https://www.sciencedirect.com/science/article/pii/S0140366418309502.

[25] Cheol-Ho Hong and Blesson Varghese. “Resource Management in Fog/Edge Computing: A Survey on Architectures, Infrastructure, and Algorithms”. In: *ACM Comput. Surv.* 52.5 (Sept. 2019). ISSN: 0360-0300. DOI: 10.1145/3326066. URL: https://doi.org/10.1145/3326066.

[26] Zhiheng Zhong et al. “Machine Learning-based Orchestration of Containers: A Taxonomy and Future Directions”. In: *ACM Computing Surveys* (2021), pp. 1–36. ISSN: 0360-0300. DOI: 10.1145/3510415. arXiv: 2106.12739.

[27] Laurens Versluis and Alexandru Iosup. “A survey of domains in workflow scheduling in computing infrastructures : Community and keyword analysis , emerging trends , and taxonomies”. In: *Future Generation Computer Systems* 123 (2021), pp. 156–177. ISSN: 0167-739X. DOI: 10.1016/j.future.2021.04.009.

[28] Anupama Mampage, Shanika Karunasekera, and Rajkumar Buyya. “A Holistic View on Resource Management in Serverless Computing Environments: Taxonomy and Future Directions”. In: *ACM Comput. Surv.*

(Jan. 2022). Just Accepted. ISSN: 0360-0300. DOI: 10.1145/3510412. URL: https://doi.org/10.1145/3510412.

[29] Breno Costa et al. "Orchestration in Fog Computing: A Comprehensive Survey". In: *ACM Comput. Surv.* 55.2 (Jan. 2022). ISSN: 0360-0300. DOI: 10.1145/3486221. URL: https://doi.org/10.1145/3486221.

[30] Aarne Mämmela et al. "Multidisciplinary and historical perspectives for developing intelligent and resource-efficient systems". In: *IEEE Access* 6 (2018), pp. 17464–17499. DOI: 10.1109/ACCESS.2018.2816605.

[31] Gerhard Weiss, ed. *Multiagent Systems*. second. Cambridge, Massachusetts, US: The MIT Press, 2013. ISBN: 978-0-262-01889-0.

[32] Ivo Grondman et al. "A survey of actor-critic reinforcement learning: Standard and natural policy gradients". In: *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews* 42.6 (2012), pp. 1291–1307. DOI: 10.1109/TSMCC.2012.2218595.

[33] Pasika Ranaweera, Anca Delia Jurcut, and Madhusanka Liyanage. "Survey on Multi-Access Edge Computing Security and Privacy". In: *IEEE Communications Surveys Tutorials* 23.2 (2021), pp. 1078–1124. DOI: 10.1109/COMST.2021.3062546.

[34] Ying Xiong et al. "Extend Cloud to Edge with KubeEdge". In: *2018 IEEE/ACM Symposium on Edge Computing (SEC)*. 2018, pp. 373–377. DOI: 10.1109/SEC.2018.00048.

[35] Bharath Srinivas Prabakaran et al. "EMAP: A Cloud-Edge Hybrid Framework for EEG Monitoring and Cross-Correlation Based Real-time Anomaly Prediction". In: *2020 57th ACM/IEEE Design Automation Conference (DAC)*. 2020, pp. 1–6. DOI: 10.1109/DAC18072.2020.9218713.

[36] Haitao Yuan et al. "Profit-Maximized Task Offloading with Simulated-annealing-based Migrating Birds Optimization in Hybrid Cloud-Edge Systems". In: *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. 2020, pp. 1218–1223. DOI: 10.1109/SMC42975.2020.9283467.

[37] Daniel Balouek-Thomert et al. "Towards a computing continuum: Enabling edge-to-cloud integration for data-driven workflows". In: *The International Journal of High Performance Computing Applications* 33.6 (2019), pp. 1159–1174. DOI: 10.1177/1094342019877383.

[38] Schahram Dustdar, Victor Casamajor Pujol, and Praveen Kumar Donta. "On distributed computing continuum systems". In: *IEEE Transactions on Knowledge and Data Engineering* (2022), pp. 1–14. ISSN: 15582191. DOI: 10.1109/TKDE.2022.3142856.

[39] ETSI GS MEC 003 V2.2.1. *Multi-access Edge Computing (MEC); Framework and Reference Architecture*. Sophia Antipolis Cedex, France, 2019.

[40] Caleb R Dewey. "Autonomy without a Self". In: *Recuperado de http://www.academia.edu/25368636/Autonomy_without_a_Self_Jul_2016...* _ (2016).

[41] John Hawkinson and Tony Bates. *Guidelines for creation, selection, and registration of an Autonomous System (AS)*. 1996.

[42] Özalp Babaoğlu. "Autonomy or Interdependence in Distributed Systems? A Position Paper". In: *Proceedings of the 3rd Workshop on ACM SIGOPS European Workshop: Autonomy or Interdependence in Distributed Systems?* EW 3. Cambridge, United Kingdom: Association for Computing Machinery, 1988, pp. 1–3. ISBN: 9781450373364. DOI: 10.1145/504092.504095. URL: https://doi.org/10.1145/504092.504095.

[43] Cosmin Carabelea, Olivier Boissier, and Adina Florea. "Autonomy in Multi-agent Systems: A Classification Attempt". In: *Agents and Computational Autonomy*. Ed. by Matthias Nickles, Michael Rovatsos, and Gerhard Weiss. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 103–113. ISBN: 978-3-540-25928-2.

[44] Qiang Liu et al. "An Edge Network Orchestrator for Mobile Augmented Reality". In: *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*. 2018, pp. 756–764. DOI: 10.1109/INFOCOM.2018.8486241.

[45] Chris Peltz. "Web services orchestration and choreography". In: *Computer* 36.10 (2003), pp. 46–52.

[46] Sara Blanc et al. "A Service Discovery Solution for Edge Choreography-Based Distributed Embedded Systems". In: *Sensors* 21.2 (2021), p. 672.

[47] Firas Al-Doghman et al. "AI-enabled Secure Microservices in Edge Computing: Opportunities and Challenges". In: *IEEE Transactions on Services Computing* (2022).

[48] Gabriele Castellano, Flavio Esposito, and Fulvio Risso. "A Distributed Orchestration Algorithm for Edge Computing Resources with Guarantees". In: Apr. 2019. DOI: 10.1109/INFOCOM.2019.8737532.

[49] Kaneez Fizza et al. "PASHE: Privacy Aware Scheduling in a Heterogeneous Fog Environment". In: *2018 IEEE 6th International Conference on Future Internet of Things and Cloud (FiCloud)*. 2018, pp. 333–340. DOI: 10.1109/FiCloud.2018.00055.

[50] Nitin Auluck et al. "Scheduling Real Time Security Aware tasks in Fog Networks". In: *IEEE Transactions on Services Computing* PP (May 2019), pp. 1–1. DOI: 10.1109/TSC.2019.2914649.

[51] Klervie Toczé and Simin Nadjm-Tehrani. "ORCH: Distributed Orchestration Framework using Mobile Edge Devices". In: *2019 IEEE 3rd International Conference on Fog and Edge Computing (ICFEC)*. 2019, pp. 1–10. DOI: 10.1109/CFEC.2019.8733152.

[52] Reza Tourani et al. *Democratizing the Edge: A Pervasive Edge Computing Framework*. 2020. arXiv: 2007.00641 [cs.NI].

[53] Asif Khan. "Key Characteristics of a Container Orchestration Platform to Enable a Modern Application". In: *IEEE Cloud Computing* 4.5 (2017), pp. 42–48. DOI: 10.1109/MCC.2017.4250933.

[54] Emiliano Casalicchio. "Container Orchestration: A Survey". In: *Systems Modeling: Methodologies and Tools*. Ed. by Antonio Puliafito and Kishor S. Trivedi. Cham: Springer International Publishing, 2019, pp. 221–235. ISBN: 978-3-319-92378-9. DOI: 10.1007/978-3-319-92378-9_14. URL: https://doi.org/10.1007/978-3-319-92378-9_14.

[55] Isam Mashhour Al Jawarneh et al. "Container Orchestration Engines: A Thorough Functional and Performance Comparison". In: *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*. 2019, pp. 1–6. DOI: 10.1109/ICC.2019.8762053.

[56] Tom Goethals, Filip De Turck, and Bruno Volckaert. "FLEDGE: Kubernetes Compatible Container Orchestration on Low-Resource Edge Devices". In: *Internet of Vehicles. Technologies and Services Toward Smart Cities*. Ed. by Ching-Hsien Hsu et al. Cham: Springer International Publishing, 2020, pp. 174–189. ISBN: 978-3-030-38651-1.

[57] Inc Kubernetes. *Kubernetes-production-grade container orchestration*. 2020.

[58] Paridhika Kayal. "Kubernetes in Fog Computing: Feasibility Demonstration, Limitations and Improvement Scope : Invited Paper". In: *2020 IEEE 6th World Forum on Internet of Things (WF-IoT)*. 2020, pp. 1–6. DOI: 10.1109/WF-IoT48130.2020.9221340.

[59] Christophe Cérin et al. "A New Docker Swarm Scheduling Strategy". In: *2017 IEEE 7th International Symposium on Cloud and Service Computing (SC2)*. 2017, pp. 112–117. DOI: 10.1109/SC2.2017.24.

[60] Michael Frampton. "Apache Mesos". In: *Complete Guide to Open Source Big Data Stack*. Berkeley, CA: Apress, 2018, pp. 97–137. ISBN: 978-1-4842-2149-5. DOI: 10.1007/978-1-4842-2149-5_4. URL: https://doi.org/10.1007/978-1-4842-2149-5_4.

[61] Fabio Casati et al. "Specification and Implementation of Exceptions in Workflow Management Systems". In: *ACM Trans. Database Syst.* 24.3 (Sept. 1999), pp. 405–451. ISSN: 0362-5915. DOI: 10.1145/328939.328996. URL: https://doi.org/10.1145/328939.328996.

[62] Jia Yu and Rajkumar Buyya. "A Taxonomy of Workflow Management Systems for Grid Computing". In: *Journal of Grid Computing* 3.3 (Sept. 2005), pp. 171–200. ISSN: 1572-9184. DOI: 10.1007/s10723-005-9010-8. URL: https://doi.org/10.1007/s10723-005-9010-8.

[63] Ryan Mitchell et al. "Exploration of Workflow Management Systems Emerging Features from Users Perspectives". In: *2019 IEEE International Conference on Big Data (Big Data)*. 2019, pp. 4537–4544. DOI: 10.1109/BigData47090.2019.9005494.

[64] David Bernstein. "Containers and Cloud: From LXC to Docker to Kubernetes". In: *IEEE Cloud Computing* 1.3 (2014), pp. 81–84. DOI: 10.1109/MCC.2014.51.

[65] Johnu George and Amit Saha. "End-to-End Machine Learning Using Kubeflow". In: *5th Joint International Conference on Data Science & Management of Data (9th ACM IKDD CODS and 27th COMAD)*. CODS-COMAD 2022. Bangalore, India: Association for Computing Machinery, 2022, pp. 336–338. ISBN: 9781450385824. DOI: 10.1145/3493700.3493768. URL: https://doi.org/10.1145/3493700.3493768.

[66] Johnu George et al. *A Scalable and Cloud-Native Hyperparameter Tuning System*. 2020. arXiv: 2006.02085 [cs.DC].

[67] Jinan Zhou et al. "Katib: A Distributed General AutoML Platform on Kubernetes". In: *2019 USENIX Conference on Operational Machine Learning (OpML 19)*. Santa Clara, CA: USENIX Association, May 2019, pp. 55–57. ISBN: 978-1-939133-00-7. URL: https://www.usenix.org/conference/opml19/presentation/zhou.

[68] Yue Zhou, Yue Yu, and Bo Ding. "Towards MLOps: A Case Study of ML Pipeline Platform". In: *2020 International Conference on Artificial Intelligence and Computer Engineering (ICAICE)*. 2020, pp. 494–500. DOI: 10.1109/ICAICE51518.2020.00102.

[69] Andrew Chen et al. "Developments in MLflow: A System to Accelerate the Machine Learning Lifecycle". In: *Proceedings of the Fourth International Workshop on Data Management for End-to-End Machine Learning*. DEEM'20. Portland, OR, USA: Association for Computing Machinery, 2020. ISBN: 9781450380232. DOI: 10.1145/3399579.3399867. URL: https://doi.org/10.1145/3399579.3399867.

[70] Matei Zaharia et al. "Accelerating the machine learning lifecycle with MLflow." In: *IEEE Data Eng. Bull.* 41.4 (2018), pp. 39–45.

[71] Sin Kit Lo et al. "Architectural Patterns for the Design of Federated Learning Systems". In: *CoRR* abs/2101.02373 (2021). arXiv: 2101.02373. URL: https://arxiv.org/abs/2101.02373.

[72] Alex Galis et al. "Management and service-aware networking architectures (MANA) for future Internet — Position paper: System functions, capabilities and requirements". In: *2009 Fourth International Conference on Communications and Networking in China*. 2009, pp. 1–13. DOI: 10.1109/CHINACOM.2009.5339964.

[73] NFVISG ETSI et al. "Network functions virtualisation (nfv); management and orchestration". In: *NFV-MAN* 1 (2014), p. v0.

[74] Anish Hirwe and Kotaro Kataoka. "LightChain: A lightweight optimisation of VNF placement for service chaining in NFV". In: *2016 IEEE NetSoft Conference and Workshops (NetSoft)*. 2016, pp. 33–37. DOI: 10.1109/NETSOFT.2016.7502438.

[75] Tung-Wei Kuo et al. “Deploying chains of virtual network functions: On the relation between link and server usage”. In: *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*. 2016, pp. 1–9. DOI: 10.1109/INFOCOM.2016.7524565.

[76] Tuan-Minh Pham and Hoai-Nam Chu. “Multi-Provider and Multi-Domain Resource Orchestration in Network Functions Virtualization”. In: *IEEE Access* 7 (2019), pp. 86920–86931. DOI: 10.1109/ACCESS.2019.2926136.

[77] Nazih Salhab, Rami Langar, and Rana Rahim. “5G network slices resource orchestration using Machine Learning techniques”. In: *Computer Networks* 188 (2021), p. 107829. ISSN: 1389-1286. DOI: https://doi.org/10.1016/j.comnet.2021.107829. URL: https://www.sciencedirect.com/science/article/pii/S1389128621000165.

[78] Faizul Bari et al. “Orchestrating Virtualized Network Functions”. In: *IEEE Transactions on Network and Service Management* 13.4 (2016), pp. 725–739. DOI: 10.1109/TNSM.2016.2569020.

[79] Riccardo Guerzoni et al. “A novel approach to virtual networks embedding for SDN management and orchestration”. In: *2014 IEEE network operations and management symposium (NOMS)*. IEEE. 2014, pp. 1–7.

[80] Talha Ahmed Khan et al. “Generic Intent-based Networking Platform for E2E Network Slice Orchestration amp; Lifecycle Management”. In: *2020 21st Asia-Pacific Network Operations and Management Symposium (APNOMS)*. 2020, pp. 49–54. DOI: 10.23919/APNOMS50412.2020.9236962.

[81] Charalampos Rotsos et al. “Network service orchestration standardization: A technology survey”. In: *Computer Standards & Interfaces* 54 (2017). SI: Standardization SDN&NFV, pp. 203–215. ISSN: 0920-5489. DOI: https://doi.org/10.1016/j.csi.2016.12.006. URL: https://www.sciencedirect.com/science/article/pii/S0920548916302458.

[82] ETSI GR MEC 035 V3.1.1 (2021-06). *Multi-access Edge Computing (MEC): Study on Inter-MEC systems and MEC-Cloud systems coordination*. Sophia Antipolis, France, 2021.

[83] Gavin Craik. “Network Transformation;(Orchestration, Network and Service Management Framework”. In: (2019).

[84] Dario Sabella et al. “Developing software for multi-access edge computing”. In: *ETSI white paper* 20 (2019), pp. 1–38.

[85] Antonio Francescon et al. “X–MANO: Cross–domain management and orchestration of network services”. In: *2017 IEEE Conference on Network Softwarization (NetSoft)*. IEEE. 2017, pp. 1–5.

[86] Thiago A. L. Genez, Luiz F. Bittencourt, and Edmundo R. M. Madeira. “Using Time Discretization to Schedule Scientific Workflows in Multiple Cloud Providers”. In: *2013 IEEE Sixth International Conference on Cloud Computing*. 2013, pp. 123–130. DOI: 10.1109/CLOUD.2013.141.

[87] Lirim Osmani et al. “Multi-Cloud Connectivity for Kubernetes in 5G Networks”. In: *IEEE Communications Magazine* 59.10 (2021), pp. 42–47. DOI: 10.1109/MCOM.110.2100124.

[88] Tejas Subramanya and Roberto Riggio. “Centralized and federated learning for predictive VNF autoscaling in multi-domain 5G networks and beyond”. In: *IEEE Transactions on Network and Service Management* 18.1 (2021), pp. 63–78.

[89] “FaAS orchestration of parallel workloads”. In: *WOSC 2019 - Proceedings of the 2019 5th International Workshop on Serverless Computing, Part of Middleware 2019* (2019), pp. 25–30. DOI: 10.1145/3366623.3368137.

[90] Johann Schleier-Smith et al. “What Serverless Computing is and Should Become: The next Phase of Cloud Computing”. In: *Commun. ACM* 64.5 (Apr. 2021), pp. 76–84. ISSN: 0001-0782. DOI: 10.1145/3406011. URL: https://doi.org/10.1145/3406011.

[91] Pedro García López et al. “Comparison of FaaS Orchestration Systems”. In: *2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion)*. 2018, pp. 148–153. DOI: 10.1109/UCC-Companion.2018.00049.

[92] Lik-Hang Lee et al. “All one needs to know about metaverse: A complete survey on technological singularity, virtual ecosystem, and research agenda”. In: *arXiv preprint arXiv:2110.05352* (2021).

[93] Jie Deng et al. “Internet Scale User-Generated Live Video Streaming: The Twitch Case”. In: *Passive and Active Measurement*. Ed. by Mohamed Ali Kaafar, Steve Uhlig, and Johanna Amann. Cham: Springer International Publishing, 2017, pp. 60–71. ISBN: 978-3-319-54328-4.

[94] “Bringing the web up to speed with WebAssembly”. In: *ACM SIGPLAN Notices* 52.6 (2017), pp. 185–200. ISSN: 15232867. DOI: 10.1145/3062341.3062363.

[95] Niko Mäkitalo et al. “WebAssembly Modules as Lightweight Containers for Liquid IoT Applications”. In: *Web Engineering (ICWE 2021)*. Ed. by Marco Brambilla et al. Vol. 12706 LNCS. Cham: Springer International Publishing, 2021, pp. 328–336. ISBN: 978-3-030-74296-6. DOI: 10.1007/978-3-030-74296-6_25.

[96] Phani Kishore Gadepalli et al. “Sledge: A Serverless-first, Light-weight Wasm Runtime for the Edge”. In: *Middleware 2020 - Proceedings of the 2020 21st International Middleware Conference* (2020), pp. 265–279. DOI: 10.1145/3423211.3425680.

[97] Samuel Rac and Mats Brorsson. “At the Edge of a Seamless Cloud Experience”. In: *arXiv preprint arXiv:2111.06157* (2021).

[98] Schahram Dustdar et al. “Principles of elastic processes”. In: *IEEE Internet Computing* 15.5 (2011), pp. 66–71.

[99] Andrea Morichetta, Victor Casamayor Pujol, and Schahram Dustdar. “A roadmap on learning and reasoning for distributed computing continuum ecosystems”. In: *IEEE International Conference on Edge Computing (EDGE 2021)*. Guangzhou, China: IEEE, 2021, pp. 25–31. DOI: 10.1109/edge53862.2021.00021.

[100] Aarne Mämmelä and Jukka Riekki. “New network architectures will be weakly coupled”. In: *IEEE Future Networks Tech Focus* April (2022), pp. 1–8.

[101] Jukka Riekki and Aarne Mämmelä. “Research and education towards smart and sustainable world”. In: *IEEE Access* 9 (2021), pp. 53156–53177.

[102] Aarne Mämmelä and Jukka Riekki. “Subsidiarity and Weak Coupling in Wireless Networks”. In: *2021 Joint European Conference on Networks and Communications 6G Summit (EuCNC/6G Summit)*. 2021, pp. 598–603. DOI: 10.1109/EuCNC/6GSummit51104.2021.9482591.

[103] C. E. Shannon. “A Mathematical Theory of Communication”. In: *Bell System Technical Journal* 27.4 (1948), pp. 623–656. DOI: 10.1002/j.1538-7305.1948.tb00917.x.

[104] Qiao Lan et al. “What is Semantic Communication? A View on Conveying Meaning in the Era of Machine Intelligence”. In: *Journal of Communications and Information Networks* 6.4 (2021), pp. 336–371. DOI: 10.23919/JCIN.2021.9663101.

[105] Jakob Foerster et al. “Learning to Communicate with Deep Multi-Agent Reinforcement Learning”. In: *Advances in Neural Information Processing Systems*. Ed. by D. Lee et al. Vol. 29. Curran Associates, Inc., 2016. URL: https://proceedings.neurips.cc/paper/2016/file/c7635bfd99248a2cdef8249ef7bfbef4-Paper.pdf.

[106] J Von Neumann and O Morgenstern. “Theory of Games and Economic Behavior. Princeton University Press, Princeton, New Jersey, USA”. In: (1944).

[107] Jiale Zhang et al. “Data Security and Privacy-Preserving in Edge Computing Paradigm: Survey and Open Issues”. In: *IEEE Access* 6 (2018), pp. 18209–18237. DOI: 10.1109/ACCESS.2018.2820162.

[108] Pasika Ranaweera, Anca Jurcut, and Madhusanka Liyanage. “MEC-Enabled 5G Use Cases: A Survey on Security Vulnerabilities and Countermeasures”. In: *ACM Comput. Surv.* 54.9 (Oct. 2021). ISSN: 0360-0300. DOI: 10.1145/3474552. URL: https://doi.org/10.1145/3474552.

[109] Liang Xiao et al. “IoT Security Techniques Based on Machine Learning: How Do IoT Devices Use AI to Enhance Security?” In: *IEEE Signal Processing Magazine* 35.5 (2018), pp. 41–49. DOI: 10.1109/MSP.2018.2825478.

[110] Abderrahmane Boudi et al. “Assessing lightweight virtualization for security-as-a-service at the network edge”. English. In: *IEICE Transactions on Communications* Volume E102B, issue 5 (2019), p. 8. ISSN: 0916-8516, 1745-1345. DOI: 10.1587/transcom.2018EUI0001. URL: http://urn.fi/URN:NBN:fi:aalto-201906033400.

[111] Pasika Ranaweera et al. “Security as a Service Platform Leveraging Multi-Access Edge Computing Infrastructure Provisions”. In: *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*. 2020, pp. 1–6. DOI: 10.1109/ICC40277.2020.9148660.

[112] Maram Alsharif and Danda B. Rawat. “Study of Machine Learning for Cloud Assisted IoT Security as a Service”. In: *Sensors* 21.4 (2021). ISSN: 1424-8220. DOI: 10.3390/s21041034. URL: https://www.mdpi.com/1424-8220/21/4/1034.

[113] Hamed HaddadPajouh et al. “AI4SAFE-IoT: An AI-Powered Secure Architecture for Edge Layer of Internet of Things”. In: *Neural Comput. Appl.* 32.20 (Oct. 2020), pp. 16119–16133. ISSN: 0941-0643. DOI: 10.1007/s00521-020-04772-3.

[114] Dinesh Verma, Seraphin Calo, and Greg Cirincione. “Distributed AI and Security Issues in Federated Environments”. In: *Proceedings of the Workshop Program of the 19th International Conference on Distributed Computing and Networking*. Workshops ICDCN ’18. Varanasi, India: Association for Computing Machinery, 2018. ISBN: 9781450363976. DOI: 10.1145/3170521.3170525. URL: https://doi.org/10.1145/3170521.3170525.

[115] Aaron Yi Ding et al. “Roadmap for Edge AI: A Dagstuhl Perspective”. In: *CoRR* abs/2112.00616 (2021). arXiv: 2112.00616. URL: https://arxiv.org/abs/2112.00616.

[116] Pawani Porambage et al. “The Roadmap to 6G Security and Privacy”. In: *IEEE Open Journal of the Communications Society* 2 (2021), pp. 1094–1122. DOI: 10.1109/OJCOMS.2021.3078081.

[117] Mithun Mukherjee et al. “Intelligent Edge Computing: Security and Privacy Challenges”. In: *IEEE Communications Magazine* 58.9 (2020), pp. 26–31. DOI: 10.1109/MCOM.001.2000297.

[118] Muktar Yahuza et al. “Systematic Review on Security and Privacy Requirements in Edge Computing: State of the Art and Future Research Opportunities”. In: *IEEE Access* 8 (2020), pp. 76541–76567. DOI: 10.1109/ACCESS.2020.2989456.

[119] Raj Sachdev. “Towards Security and Privacy for Edge AI in IoT/IoE based Digital Marketing Environments”. In: *2020 Fifth International Conference on Fog and Mobile Edge Computing (FMEC)*. 2020, pp. 341–346. DOI: 10.1109/FMEC49853.2020.9144755.

[120] Miao Du et al. “Big Data Privacy Preserving in Multi-Access Edge Computing for Heterogeneous Internet of Things”. In: *IEEE Communications Magazine* 56.8

(2018), pp. 62–67. DOI: 10.1109/MCOM.2018.1701148.

[121] Gary Marcus and Ernest Davis. *Rebooting AI: Building Artificial Intelligence We Can Trust*. USA: Pantheon Books, 2019. ISBN: 1524748250.

[122] Kang Wei et al. "Federated Learning With Differential Privacy: Algorithms and Performance Analysis". In: *IEEE Transactions on Information Forensics and Security* 15 (2020), pp. 3454–3469. DOI: 10.1109/TIFS.2020.2988575.

[123] Bo Liu et al. "When Machine Learning Meets Privacy: A Survey and Outlook". In: *ACM Computing Surveys* 54.2 (2021). ISSN: 0360-0300. DOI: 10.1145/3436755. URL: https://doi.org/10.1145/3436755.

[124] Elisa Bertino et al. "AI for Security and Security for AI". In: *Proceedings of the Eleventh ACM Conference on Data and Application Security and Privacy*. CODASPY '21. Virtual Event, USA: Association for Computing Machinery, 2021, pp. 333–334. ISBN: 9781450381437. DOI: 10.1145/3422337.3450357. URL: https://doi.org/10.1145/3422337.3450357.

[125] Yushan Siriwardhana et al. "AI and 6G Security: Opportunities and Challenges". In: *2021 Joint European Conference on Networks and Communications 6G Summit (EuCNC/6G Summit)*. 2021, pp. 616–621. DOI: 10.1109/EuCNC/6GSummit51104.2021.9482503.

[126] Pawani Porambage et al. "Sec-edgeAI: a vision for using artificial intelligence for securing the edge". In: *10th Nordic Workshop on System and Network Optimization for Wireless (SNOW)*. 2019.

[127] Matthew Jagielski et al. "Manipulating Machine Learning: Poisoning Attacks and Countermeasures for Regression Learning". In: *2018 IEEE Symposium on Security and Privacy (SP)*. 2018, pp. 19–35. DOI: 10.1109/SP.2018.00057.

[128] Nikolaos Pitropakis et al. "A taxonomy and survey of attacks against machine learning". In: *Computer Science Review* 34 (2019), p. 100199. ISSN: 1574-0137. DOI: https://doi.org/10.1016/j.cosrev.2019.100199. URL: https://www.sciencedirect.com/science/article/pii/S1574013718303289.

[129] Vale Tolpegin et al. "Data Poisoning Attacks Against Federated Learning Systems". In: *Computer Security – ESORICS 2020*. Ed. by Liqun Chen et al. Cham: Springer International Publishing, 2020, pp. 480–501. ISBN: 978-3-030-58951-6.

[130] Jared Saia and Mahdi Zamani. "Recent Results in Scalable Multi-Party Computation". In: *SOFSEM 2015: Theory and Practice of Computer Science*. Ed. by Giuseppe F. Italiano et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 24–44. ISBN: 978-3-662-46078-8.

[131] Brendan McMahan et al. "Communication-Efficient Learning of Deep Networks from Decentralized Data". In: *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*. Ed. by Aarti Singh and Jerry Zhu. Vol. 54. Proceedings of Machine Learning Research. PMLR, 20–22 Apr 2017, pp. 1273–1282. URL: https://proceedings.mlr.press/v54/mcmahan17a.html.

[132] Shiqiang Wang et al. "When Edge Meets Learning: Adaptive Control for Resource-Constrained Distributed Machine Learning". In: *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*. 2018, pp. 63–71. DOI: 10.1109/INFOCOM.2018.8486403.

[133] Tian Li et al. "Federated Learning: Challenges, Methods, and Future Directions". In: *IEEE Signal Processing Magazine* 37.3 (2020), pp. 50–60. DOI: 10.1109/MSP.2020.2975749.

[134] Lingjuan Lyu et al. "Towards Fair and Privacy-Preserving Federated Deep Models". In: *IEEE Transactions on Parallel and Distributed Systems* 31.11 (2020), pp. 2524–2541. DOI: 10.1109/TPDS.2020.2996273.

[135] Dinh C. Nguyen et al. "Federated Learning Meets Blockchain in Edge Computing: Opportunities and Challenges". In: *IEEE Internet of Things Journal* (Apr. 2021). DOI: 10.1109/JIOT.2021.3072611.

[136] Yue Zhao et al. *Federated Learning with Non-IID Data*. 2018. arXiv: 1806.00582.

[137] Eunjeong Jeong et al. "Communication-Efficient On-Device Machine Learning: Federated Distillation and Augmentation under Non-IID Private Data". In: *CoRR* abs/1811.11479 (2018). arXiv: 1811.11479. URL: http://arxiv.org/abs/1811.11479.

[138] Tian Li et al. *Federated Optimization in Heterogeneous Networks*. 2020. arXiv: 1812.06127.

[139] Hyowoon Seo et al. "Federated Knowledge Distillation". In: *CoRR* abs/2011.02367 (2020). arXiv: 2011.02367. URL: https://arxiv.org/abs/2011.02367.

[140] Seungeun Oh et al. "Mix2FLD: Downlink Federated Learning After Uplink Federated Distillation With Two-Way Mixup". In: *IEEE Communications Letters* 24.10 (2020), pp. 2211–2215. DOI: 10.1109/LCOMM.2020.3003693.

[141] Seyyedali Hosseinalipour et al. "From Federated to Fog Learning: Distributed Machine Learning over Heterogeneous Wireless Networks". In: *IEEE Communications Magazine* 58 (Dec. 2020), pp. 41–47. DOI: 10.1109/MCOM.001.2000410.

[142] Seyyedali Hosseinalipour et al. *Multi-Stage Hybrid Federated Learning over Large-Scale D2D-Enabled Fog Networks*. 2020. arXiv: 2007.09511.

[143] Peter Kairouz et al. *Advances and Open Problems in Federated Learning*. 2021. arXiv: 1912.04977.

[144] Paul Vanhaesebrouck, Aurélien Bellet, and Marc Tommasi. "Decentralized Collaborative Learning of Personalized Models over Networks". In: *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*. Ed. by Aarti Singh and Jerry Zhu. Vol. 54. Proceedings of Machine Learning Research.

PMLR, 20–22 Apr 2017, pp. 509–517. URL: https://proceedings.mlr.press/v54/vanhaesebrouck17a.html.

[145] Stephen Boyd et al. "Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers". In: *Foundations and Trends in Machine Learning* 3 (2011), pp. 1–122. DOI: 10.1561/2200000016.

[146] Anis Elgabli et al. "GADMM: Fast and Communication Efficient Framework for Distributed Machine Learning". In: *Journal of Machine Learning Research* 21.76 (2020), pp. 1–39. URL: http://jmlr.org/papers/v21/19-718.html.

[147] Jeff Daily et al. "GossipGraD: Scalable Deep Learning using Gossip Communication based Asynchronous Gradient Descent". In: *CoRR* abs/1803.05880 (2018). arXiv: 1803.05880. URL: http://arxiv.org/abs/1803.05880.

[148] Stefano Savazzi, Monica Nicoli, and Vittorio Rampa. "Federated Learning With Cooperating Devices: A Consensus Approach for Massive IoT Networks". In: *IEEE Internet of Things Journal* 7.5 (2020), pp. 4641–4654. DOI: 10.1109/JIOT.2020.2964162.

[149] Chenghao Hu, Jingyan Jiang, and Zhi Wang. *Decentralized Federated Learning: A Segmented Gossip Approach*. 2019. arXiv: 1908.07782.

[150] Ali H. Sayed. Now Foundations and Trends, 2014. ISBN: 978-1-601-98851-5.

[151] Stefano Savazzi et al. "Opportunities of Federated Learning in Connected, Cooperative, and Automated Industrial Systems". In: *IEEE Communications Magazine* 59.2 (2021), pp. 16–21. DOI: 10.1109/MCOM.001.2000200.

[152] Jianshu Chen and Ali H. Sayed. "Diffusion Adaptation Strategies for Distributed Optimization and Learning Over Networks". In: *IEEE Transactions on Signal Processing* 60.8 (2012), pp. 4289–4305. DOI: 10.1109/TSP.2012.2198470.

[153] Reza Olfati-Saber, J. Alex Fax, and Richard M. Murray. "Consensus and Cooperation in Networked Multi-Agent Systems". In: *Proceedings of the IEEE* 95.1 (2007), pp. 215–233. DOI: 10.1109/JPROC.2006.887293.

[154] Muhammad Shayan et al. "Biscotti: A Blockchain System for Private and Secure Federated Learning". In: *IEEE Transactions on Parallel and Distributed Systems* 32.7 (2021), pp. 1513–1525. DOI: 10.1109/TPDS.2020.3044223.

[155] Hyesung Kim et al. "Blockchained On-Device Federated Learning". In: *IEEE Communications Letters* 24.6 (2020), pp. 1279–1283. DOI: 10.1109/LCOMM.2019.2921755.

[156] Zhilin Wang and Qin Hu. *Blockchain-based Federated Learning: A Comprehensive Survey*. 2021. arXiv: 2110.02182.

[157] Chung-Kai Yu, Mihaela van der Schaar, and Ali H. Sayed. "Reputation design for adaptive networks with selfish agents". In: *IEEE Workshop on Signal Processing Advances in Wireless Communications*. 2013, pp. 160–164. DOI: 10.1109/SPAWC.2013.6612032.

[158] Aurélien Bellet et al. "Personalized and Private Peer-to-Peer Machine Learning". In: *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*. Ed. by Amos Storkey and Fernando Perez-Cruz. Vol. 84. Proceedings of Machine Learning Research. PMLR, Sept. 2018, pp. 473–481. URL: https://proceedings.mlr.press/v84/bellet18a.html.

[159] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. "BadNets: Identifying Vulnerabilities in the Machine Learning Model Supply Chain". In: *CoRR* abs/1708.06733 (2017). arXiv: 1708.06733. URL: http://arxiv.org/abs/1708.06733.

[160] Matthew Fredrikson et al. "Privacy in Pharmacogenetics: An End-to-End Case Study of Personalized Warfarin Dosing". In: *23rd USENIX Security Symposium (USENIX Security 14)*. San Diego, CA: USENIX Association, Aug. 2014, pp. 17–32. ISBN: 978-1-931971-15-7. URL: https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/fredrikson_matthew.

[161] Reza Shokri et al. "Membership Inference Attacks Against Machine Learning Models". In: *2017 IEEE Symposium on Security and Privacy (SP)*. 2017, pp. 3–18. DOI: 10.1109/SP.2017.41.

[162] Pierangela Samarati and Latanya Sweeney. "Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression". In: (1998).

[163] Ashwin Machanavajjhala et al. "L-Diversity: Privacy beyond k-Anonymity". In: *ACM Trans. Knowl. Discov. Data* 1.1 (2007), 3–es. ISSN: 1556-4681. DOI: 10.1145/1217299.1217302. URL: https://doi.org/10.1145/1217299.1217302.

[164] Ninghui Li, Tiancheng Li, and Suresh Venkatasubramanian. "t-Closeness: Privacy Beyond k-Anonymity and l-Diversity". In: *2007 IEEE 23rd International Conference on Data Engineering*. 2007, pp. 106–115. DOI: 10.1109/ICDE.2007.367856.

[165] Cynthia Dwork et al. "Calibrating Noise to Sensitivity in Private Data Analysis". In: *Theory of Cryptography*. Ed. by Shai Halevi and Tal Rabin. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 265–284. ISBN: 978-3-540-32732-5.

[166] Cynthia Dwork. "Differential Privacy: A Survey of Results". In: *Theory and Applications of Models of Computation*. Ed. by Manindra Agrawal et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 1–19. ISBN: 978-3-540-79228-4.

[167] Ronald L Rivest, Len Adleman, Michael L Dertouzos, et al. "On data banks and privacy homomorphisms". In: *Foundations of secure computation* 4.11 (1978), pp. 169–180.

[168] Craig Gentry. “Fully Homomorphic Encryption Using Ideal Lattices”. In: *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing*. STOC ’09. Bethesda, MD, USA: Association for Computing Machinery, 2009, pp. 169–178. ISBN: 9781605585062. DOI: 10.1145/1536414.1536440.

[169] Pascal Paillier. “Public-Key Cryptosystems Based on Composite Degree Residuosity Classes”. In: *Advances in Cryptology — EUROCRYPT ’99*. Ed. by Jacques Stern. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 223–238. ISBN: 978-3-540-48910-8.

[170] Ximeng Liu et al. “Privacy-Preserving Patient-Centric Clinical Decision Support System on Naïve Bayesian Classification”. In: *IEEE Journal of Biomedical and Health Informatics* 20.2 (2016), pp. 655–668. DOI: 10.1109/JBHI.2015.2407157.

[171] Le Trieu Phong et al. “Privacy-Preserving Deep Learning via Additively Homomorphic Encryption”. In: *IEEE Transactions on Information Forensics and Security* 13.5 (2018), pp. 1333–1345. DOI: 10.1109/TIFS.2017.2787987.

[172] Ran Gilad-Bachrach et al. “CryptoNets: Applying Neural Networks to Encrypted Data with High Throughput and Accuracy”. In: *Proceedings of The 33rd International Conference on Machine Learning*. Ed. by Maria Florina Balcan and Kilian Q. Weinberger. Vol. 48. Proceedings of Machine Learning Research. New York, New York, USA: PMLR, 2016, pp. 201–210. URL: https://proceedings.mlr.press/v48/gilad-bachrach16.html.

[173] Amit Sahai and Brent Waters. “Fuzzy Identity-Based Encryption”. In: *Advances in Cryptology – EUROCRYPT 2005*. Ed. by Ronald Cramer. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 457–473. ISBN: 978-3-540-32055-5.

[174] Dan Boneh, Amit Sahai, and Brent Waters. “Functional Encryption: Definitions and Challenges”. In: *Theory of Cryptography*. Ed. by Yuval Ishai. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 253–273. ISBN: 978-3-642-19571-6.

[175] Théo Ryffel et al. “Partially Encrypted Machine Learning Using Functional Encryption”. In: *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. Red Hook, NY, USA: Curran Associates Inc., 2019.

[176] Rosario Gennaro. “Verifiable Outsourced Computation: A Survey”. In: *Proceedings of the ACM Symposium on Principles of Distributed Computing*. PODC ’17. Washington, DC, USA: Association for Computing Machinery, 2017, p. 313. ISBN: 9781450349925. DOI: 10.1145/3087801.3087872. URL: https://doi.org/10.1145/3087801.3087872.

[177] Andrew Chi-Chih Yao. “How to generate and exchange secrets”. In: *27th Annual Symposium on Foundations of Computer Science (sfcs 1986)*. 1986, pp. 162–167. DOI: 10.1109/SFCS.1986.25.

[178] Bita Darvish Rouhani, M. Sadegh Riazi, and Farinaz Koushanfar. “Deepsecure: Scalable Provably-Secure Deep Learning”. In: *Proceedings of the 55th Annual Design Automation Conference*. DAC ’18. San Francisco, California: Association for Computing Machinery, 2018. ISBN: 9781450357005. DOI: 10.1145/3195970.3196023. URL: https://doi.org/10.1145/3195970.3196023.

[179] Wenting Zheng et al. “Cerebro: A Platform for Multi-Party Cryptographic Collaborative Learning”. In: *30th USENIX Security Symposium (USENIX Security 21)*. USENIX Association, Aug. 2021, pp. 2723–2740. ISBN: 978-1-939133-24-3. URL: https://www.usenix.org/conference/usenixsecurity21/presentation/zheng.

[180] M. Sadegh Riazi et al. “Chameleon: A Hybrid Secure Computation Framework for Machine Learning Applications”. In: *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*. ASIACCS ’18. Incheon, Republic of Korea: Association for Computing Machinery, 2018, pp. 707–721. ISBN: 9781450355766. DOI: 10.1145/3196494.3196522. URL: https://doi.org/10.1145/3196494.3196522.

[181] Chiraag Juvekar, Vinod Vaikuntanathan, and Anantha Chandrakasan. “GAZELLE: A Low Latency Framework for Secure Neural Network Inference”. In: *27th USENIX Security Symposium (USENIX Security 18)*. Baltimore, MD: USENIX Association, Aug. 2018, pp. 1651–1669. ISBN: 978-1-939133-04-5. URL: https://www.usenix.org/conference/usenixsecurity18/presentation/juvekar.

[182] Karl Tuyls and Gerhard Weiss. “Multiagent Learning: Basics, Challenges, and Prospects”. In: *Ai Magazine* 33 (2012), pp. 41–52. DOI: 10.1609/aimag.v33i3.2426.

[183] Yoav Shoham and Kevin Leyton-Brown. *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge University Press, 2008. DOI: 10.1017/CBO9780511811654.

[184] Kaiqing Zhang, Zhuoran Yang, and Tamer Başar. “Multi-Agent Reinforcement Learning: A Selective Overview of Theories and Algorithms”. In: *Handbook of Reinforcement Learning and Control*. Ed. by Kyriakos G. Vamvoudakis et al. Cham: Springer International Publishing, 2021, pp. 321–384. ISBN: 978-3-030-60990-0. DOI: 10.1007/978-3-030-60990-0_12. URL: https://doi.org/10.1007/978-3-030-60990-0_12.

[185] Piotr J. Gmytrasiewicz and Prashant Doshi. “A Framework for Sequential Planning in Multi-Agent Settings”. In: *Journal of Artificial Intelligence Research* 24.1 (2005), pp. 49–79. URL: https://doi.org/10.1613/jair.1579.

[186] Pablo Hernandez-Leal et al. *A Survey of Learning in Multiagent Environments: Dealing with Non-Stationarity*. 2019. arXiv: 1707.09183.

[187] Ann Nowé, Peter Vrancx, and Yann-Michaël De Hauwere. “Game Theory and Multi-agent Reinforcement Learning”. In: *Reinforcement Learning: State-*

of-the-Art. Ed. by Marco Wiering and Martijn van Otterlo. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 441–470. ISBN: 978-3-642-27645-3. DOI: 10.1007/978-3-642-27645-3_14.

[188] Lucian Buşoniu, Robert Babuška, and Bart De Schutter. "Multi-agent Reinforcement Learning: An Overview". In: *Innovations in Multi-Agent Systems and Applications - 1*. Ed. by Dipti Srinivasan and Lakhmi C. Jain. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 183–221. ISBN: 978-3-642-14435-6. DOI: 10.1007/978-3-642-14435-6_7. URL: https://doi.org/10.1007/978-3-642-14435-6_7.

[189] Eric A. Hansen, Daniel S. Bernstein, and Shlomo Zilberstein. "Dynamic Programming for Partially Observable Stochastic Games". In: *Proceedings of the 19th National Conference on Artifical Intelligence*. AAAI'04. San Jose, California: AAAI Press, 2004, pp. 709–715. ISBN: 0262511835.

[190] Frans A. Oliehoek and Christopher Amato. *A Concise Introduction to Decentralized POMDPs*. 1st. Springer Publishing Company, Incorporated, 2016. ISBN: 9783319289298. DOI: 10.1007/978-3-319-28929-8.

[191] Daniel Bernstein et al. "The Complexity of Decentralized Control of Markov Decision Processes". In: *Mathematics of Operations Research* 27 (4 2002), pp. 819–840. DOI: 10.1287/moor.27.4.819.297.

[192] Michael Bowling and Manuela Veloso. "Multiagent learning using a variable learning rate". In: *Artificial Intelligence* 136.2 (2002), pp. 215–250. DOI: 10.1016/S0004-3702(02)00121-2.

[193] Bikramjit Banerjee and Jing Peng. "Performance Bounded Reinforcement Learning in Strategic Interactions". In: *Proceedings of the 19th National Conference on Artifical Intelligence*. AAAI'04. San Jose, California: AAAI Press, 2004, pp. 2–7. ISBN: 0262511835.

[194] Pablo Hernandez-Leal et al. "Identifying and tracking switching, non-stationary opponents: A Bayesian approach". In: *Workshops at the Thirtieth AAAI Conference on Artificial Intelligence*. 2016.

[195] Brenda Ng et al. "Bayes-Adaptive Interactive POMDPs". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 26.1 (2021), pp. 1408–1414.

[196] Ekhlas Sonu, Yingke Chen, and Prashant Doshi. *Individual Planning in Agent Populations: Exploiting Anonymity and Frame-Action Hypergraphs*. 2015. arXiv: 1503.07220.

[197] Pablo Hernandez-Leal, Bilal Kartal, and Matthew Taylor. "A survey and critique of multiagent deep reinforcement learning". In: *Autonomous Agents and Multi-Agent Systems* 33 (2019), pp. 750–797. DOI: 10.1007/s10458-019-09421-1.

[198] Volodymyr Mnih et al. "Human-level control through deep reinforcement learning". In: *Nature* 518 (2015), pp. 529–533. DOI: 10.1038/nature14236.

[199] Thanh Nguyen, Ngoc Duy Nguyen, and Saeid Nahavandi. "Deep Reinforcement Learning for Multiagent Systems: A Review of Challenges, Solutions, and Applications". In: *IEEE Transactions on Cybernetics* 50.9 (2020), pp. 3826–3839. DOI: 10.1109/TCYB.2020.2977374.

[200] Sven Gronauer and Klaus Dieopold. "Multi-agent deep reinforcement learning: a survey". In: *Artificial Intelligence Review* (2021). DOI: 10.1007/s10462-021-09996-w.

[201] Alvaro Ovalle Castaneda. "Deep Reinforcement Learning Variants of Multi-Agent Learning Algorithms". MA thesis. University of Edinburgh, 2016.

[202] Chao Yu et al. "Multiagent Learning of Coordination in Loosely Coupled Multiagent Systems". In: *IEEE Transactions on Cybernetics* 45.12 (2015), pp. 2853–2867. DOI: 10.1109/TCYB.2014.2387277.

[203] Jakob Foerster et al. "Learning with Opponent-Learning Awareness". In: *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*. AAMAS '18. Stockholm, Sweden: International Foundation for Autonomous Agents and Multiagent Systems, 2018, pp. 122–130.

[204] Ryan Lowe et al. "Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper/2017/file/68a9750337a418a86fe06c1991a1d64c-Paper.pdf.

[205] Ozsel Kilinc and Giovanni Montana. *Multi-agent Deep Reinforcement Learning with Extremely Noisy Observations*. 2018. arXiv: 1812.00922.

[206] Jiaju Qi et al. *Federated Reinforcement Learning: Techniques, Applications, and Open Challenges*. 2021. arXiv: 2108.11887.

[207] Han Cha et al. *Federated Reinforcement Distillation with Proxy Experience Memory*. DOI: 10.36227/techrxiv.12645497.v1.

[208] Han Cha et al. "Proxy Experience Replay: Federated Distillation for Distributed Reinforcement Learning". In: *IEEE Intelligent Systems* 35.4 (2020), pp. 94–101. DOI: 10.1109/MIS.2020.2994942.

[209] Kristof Van Moffaert and Ann Nowé. "Multi-Objective Reinforcement Learning using Sets of Pareto Dominating Policies". In: *Journal of Machine Learning Research* 15.107 (2014), pp. 3663–3692. URL: http://jmlr.org/papers/v15/vanmoffaert14a.html.

[210] Roxana Rădulescu et al. "Multi-Objective Multi-Agent Decision Making: A Utility-Based Analysis and Survey". In: *Autonomous Agents and Multi-Agent Systems* 34.1 (Dec. 2019). ISSN: 1387-2532. DOI: 10.1007/s10458-019-09433-x. URL: https://doi.org/10.1007/s10458-019-09433-x.

[211] Diederik M. Roijers et al. "A Survey of Multi-Objective Sequential Decision-Making". In: *Journal of*

Artificial Intelligence Research 48.1 (2013), pp. 67–113.

[212] Peter Vamplew et al. "On the Limitations of Scalarisation for Multi-objective Reinforcement Learning of Pareto Fronts". In: *AI 2008: Advances in Artificial Intelligence*. Ed. by Wayne Wobcke and Mengjie Zhang. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 372–378.

[213] Kristof Van Moffaert et al. "A novel adaptive weight selection algorithm for multi-objective multi-agent reinforcement learning". In: *2014 International Joint Conference on Neural Networks (IJCNN)*. 2014, pp. 2306–2314. DOI: 10.1109/IJCNN.2014.6889637.

[214] Felipe Silva and Anna Costa. "A Survey on Transfer Learning for Multiagent Reinforcement Learning Systems". In: *Journal of Artificial Intelligence Research* 64.1 (2019), pp. 645–703. DOI: 10.1613/jair.1.11396.

[215] Felipe Leno Da Silva and Anna Helena Reali Costa. "Object-Oriented Curriculum Generation for Reinforcement Learning". In: *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*. AAMAS '18. Stockholm, Sweden: International Foundation for Autonomous Agents and Multiagent Systems, 2018, pp. 1026–1034.

[216] Ruben Glatt, Felipe Leno Da Silva, and Anna Helena Reali Costa. "Towards Knowledge Transfer in Deep Reinforcement Learning". In: *2016 5th Brazilian Conference on Intelligent Systems (BRACIS)*. 2016, pp. 91–96. DOI: 10.1109/BRACIS.2016.027.

[217] Peter E. Caines, Minyi Huang, and Roland P. Malhamé. "Mean Field Games". In: *Handbook of Dynamic Game Theory*. Ed. by Tamer Basar and Georges Zaccour. Cham: Springer International Publishing, 2017, pp. 1–28. ISBN: 978-3-319-27335-8. DOI: 10.1007/978-3-319-27335-8_7-1.

[218] Yaodong Yang et al. *Mean Field Multi-Agent Reinforcement Learning*. 2020. arXiv: 1802.05438.

[219] KARL TUYLS and ANN NOWÉ. "Evolutionary game theory and multi-agent reinforcement learning". In: *The Knowledge Engineering Review* 20.1 (2005), pp. 63–90. DOI: 10.1017/S026988890500041X.

[220] Daan Bloembergen et al. "Evolutionary Dynamics of Multi-Agent Learning: A Survey". In: *Journal of Artificial Intelligence Research* 53 (2015), pp. 659–697. DOI: 10.1613/jair.4818.

[221] David Moriarty, Alan Schultz, and John Grefenstette. "Evolutionary Algorithms for Reinforcement Learning". In: *Journal of Artificial Intelligence Research* 11 (1999), pp. 241–276. DOI: 10.1613/jair.613.

[222] Amrita Chakraborty and Arpan Kumar Kar. "Swarm Intelligence: A Review of Algorithms". In: *Nature-Inspired Computing and Optimization: Theory and Applications*. Ed. by Srikanta Patnaik, Xin-She Yang, and Kazumi Nakamatsu. Cham: Springer International Publishing, 2017, pp. 475–494. DOI: 10.1007/978-3-319-50920-4_19.

[223] Melanie Schranz et al. "Swarm Robotic Behaviors and Current Applications". In: *Frontiers in Robotics and AI* 7 (2020). DOI: 10.3389/frobt.2020.00036.

[224] Yongkun Zhou, Bin Rao, and Wei Wang. "UAV Swarm Intelligence: Recent Advances and Future Trends". In: *IEEE Access* 8 (2020), pp. 183856–183878. DOI: 10.1109/ACCESS.2020.3028865.

[225] Aliaa F. Raslan, Ahmed F. Ali, and Ashraf Darwish. "Swarm intelligence algorithms and their applications in Internet of Things". In: *Swarm Intelligence for Resource Management in Internet of Things*. Ed. by Aboul Ella Hassanien and Ashraf Darwish. Intelligent Data-Centric Systems. Academic Press, 2020, pp. 1–19. DOI: 10.1016/B978-0-12-818287-1.00003-6.

[226] Chih-Han Yu, Justin Werfel, and Radhika Nagpal. "Collective decision-making in multi-agent systems by implicit leadership". In: *9th International Conference on Autonomous Agents and Multiagent Systems*. Vol. 2. 2010, pp. 1189–1196. DOI: 10.1145/1838186.1838192.

[227] Yingying Ding, Yan He, and Jing-Ping Jiang. "Self-organizing multi-robot system based on personality evolution". In: *IEEE International Conference on Systems, Man and Cybernetics*. Vol. 5. 2002. DOI: 10.1109/ICSMC.2002.1176414.

[228] S. N. Givigi and H. M. Schwartz. "A Game Theoretic Approach to Swarm Robotics". In: *Applied Bionics and Biomechanics* 3 (2006). DOI: 10.1533/abbi.2005.0021.

[229] Samuel T. Langlois et al. "Metareasoning Structures, Problems, and Modes for Multiagent Systems: A Survey". In: *IEEE Access* 8 (2020), pp. 183080–183089. DOI: 10.1109/ACCESS.2020.3028751.

[230] Stuart Russell and Eric Wefald. "Principles of metareasoning". In: *Artificial Intelligence* 49.1 (1991), pp. 361–395. DOI: https://doi.org/10.1016/0004-3702(91)90015-C.

[231] Justin Svegliato, Kyle Hollins Wray, and Shlomo Zilberstein. "Meta-Level Control of Anytime Algorithms with Online Performance Prediction". In: *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*. 2018, pp. 1499–1505. DOI: 10.24963/ijcai.2018/208.

[232] Sarit Kraus. "Negotiation and cooperation in multi-agent environments". In: *Artificial Intelligence* 94.1 (1997), pp. 79–97. DOI: https://doi.org/10.1016/S0004-3702(97)00025-8.

[233] N. Jennings et al. "Automated Negotiation: Prospects, Methods and Challenges". In: *Group Decision and Negotiation* 10 (Mar. 2001), pp. 199–215. DOI: 10.1023/A:1008746126376.

[234] Gheorghe Cosmin Silaghi, Liviu Dan Şerban, and Cristian Marius Litan. "A Framework for Building Intelligent SLA Negotiation Strategies under Time Constraints". In: *Economics of Grids, Clouds, Systems, and Services*. Ed. by Jörn Altmann and Omer F. Rana.

Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 48–61.

[235] Shaheen Fatima, Sarit Kraus, and Michael Wooldridge. *Principles of Automated Negotiation*. Cambridge University Press, 2014. DOI: 10 . 1017 / CBO9780511751691.

[236] Fernando Lopes and Helder Coelho, eds. *Negotiation and argumentation in multi-agent systems: fundamentals, theories, systems and applications*. Bentham Science Publishers, 2014.

[237] Tim Baarslag et al. "Learning about the Opponent in Automated Bilateral Negotiation: A Comprehensive Survey of Opponent Modeling Techniques". In: *Autonomous Agents and Multi-Agent Systems* 30.5 (2016), pp. 849–898. DOI: 10.1007/s10458-015-9309-1.

[238] Usha Kiruthika, Thamarai Selvi Somasundaram, and Kanaga Suba Subramanian. "Lifecycle Model of a Negotiation Agent: A Survey of Automated Negotiation Techniques". In: *Group Decision and Negotiation* 29 (2020), pp. 1239–1262. DOI: 10.1007/s10726-020-09704-z.

[239] Z Ren and C.J Anumba. "Multi-agent systems in construction–state of the art and prospects". In: *Automation in Construction* 13.3 (2004), pp. 421–434. DOI: https://doi.org/10.1016/j.autcon.2003.12.002.

[240] D.D.B. van Bragt and J.A. La Poutré. "Why Agents for Automated Negotiations Should Be Adaptive". In: *NETNOMICS: Economic Research and Electronic Networking* 5 (2003), pp. 101–118. DOI: 10.1023/A:1026021701904.

[241] Iyad Rahwan et al. "Argumentation-based negotiation". In: *The Knowledge Engineering Review* 18.4 (2003), pp. 343–375. DOI: 10 . 1017 / S0269888904000098.

[242] Iyad Rahwan et al. "A formal analysis of interest-based negotiation". In: *Annals of Mathematics and Artificial Intelligence* 55 (2009), pp. 253–276. DOI: 10.1007/s10472-009-9145-6.

[243] Iyad Rahwan et al. "On the benefits of exploiting underlying goals in argument-based negotiation". In: *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence*. Ed. by Robert C. Holte and Adele Howe. The AAAI Press, 2007.

[244] Iyad Rahwan and Kate Larson. "Argumentation and Game Theory". In: 2009, pp. 321–339. DOI: 10.1007/978-0-387-98197-0_16.

[245] Nicoletta Fornara and Marco Colombetti. "A commitment-based approach to agent communication". In: *Applied Artificial Intelligence* 18 (2004), pp. 853–866.

[246] Amit K. Chopra and Munindar P. Singh. "Specifying and Applying Commitment-Based Business Patterns". In: *The 10th International Conference on Autonomous Agents and Multiagent Systems - Volume 2*. AAMAS '11. Taipei, Taiwan, 2011, pp. 475–482.

[247] Ian Goodfellow. *NIPS 2016 Tutorial: Generative Adversarial Networks*. 2017. arXiv: 1701.00160.

[248] Chelsea Finn, Ian Goodfellow, and Sergey Levine. "Unsupervised Learning for Physical Interaction through Video Prediction". In: *Advances in Neural Information Processing Systems*. Ed. by D. Lee et al. Vol. 29. Curran Associates, Inc., 2016. URL: https://proceedings.neurips.cc/paper/2016/file/d9d4f495e875a2e075a1a4a6e1b9770f-Paper.pdf.

[249] Chelsea Finn et al. "A Connection between Generative Adversarial Networks, Inverse Reinforcement Learning, and Energy-Based Models". In: *CoRR* abs/1611.03852 (2016). arXiv: 1611.03852. URL: http://arxiv.org/abs/1611.03852.

[250] Konstantinos Bousmalis et al. "Unsupervised Pixel-Level Domain Adaptation With Generative Adversarial Networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. July 2017.

[251] Yong Xiao et al. "Toward Self-Learning Edge Intelligence in 6G". In: *IEEE Communications Magazine* 58.12 (2020), pp. 34–40. DOI: 10.1109/MCOM.001.2000388.

[252] Maciej Wiatrak, Stefano V. Albrecht, and Andrew Nystrom. *Stabilizing Generative Adversarial Networks: A Survey*. 2020. arXiv: 1910.00927.

[253] Zhipeng Cai et al. "Generative Adversarial Networks: A Survey Toward Private and Secure Applications". In: *ACM Comput. Surv.* 54.6 (July 2021). ISSN: 0360-0300. DOI: 10.1145/3459992. URL: https://doi.org/10.1145/3459992.

[254] Ian Goodfellow et al. "Generative Adversarial Nets". In: *Advances in Neural Information Processing Systems*. Ed. by Z. Ghahramani et al. Vol. 27. Curran Associates, Inc., 2014. URL: https://proceedings.neurips.cc/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf.

[255] Martin Arjovsky, Soumith Chintala, and Léon Bottou. *Wasserstein GAN*. 2017. arXiv: 1701.07875 [stat.ML].

[256] Yongjun Hong et al. "How Generative Adversarial Networks and Their Variants Work: An Overview". In: *ACM Comput. Surv.* 52.1 (Feb. 2019). ISSN: 0360-0300. DOI: 10.1145/3301282. URL: https://doi.org/10.1145/3301282.

[257] Mario Lucic et al. "Are GANs Created Equal? A Large-Scale Study". In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio et al. Vol. 31. Curran Associates, Inc., 2018. URL: https://proceedings.neurips.cc/paper/2018/file/e46de7e1bcaaced9a54f1e9d0d2f800d-Paper.pdf.

[258] Yaqing Wang et al. "Generalizing from a Few Examples: A Survey on Few-shot Learning". In: *ACM Computing Surveys* 53 (2020), pp. 1–34. DOI: 10.1145/3386252.

[259] Jiang Lu et al. *Learning from Very Few Samples: A Survey*. 2020. arXiv: 2009.02653.

[260] Yan Duan et al. "One-Shot Imitation Learning". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper/2017/file/ba3866600c3540f67c1e9575e213be0a-Paper.pdf.

[261] Léon Bottou and Olivier Bousquet. "The Tradeoffs of Large Scale Learning". In: *Advances in Neural Information Processing Systems*. Ed. by J. Platt et al. Vol. 20. Curran Associates, Inc., 2007.

[262] Chelsea Finn, Pieter Abbeel, and Sergey Levine. "Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks". In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, June 2017, pp. 1126–1135. URL: https://proceedings.mlr.press/v70/finn17a.html.

[263] Chelsea Finn, Kelvin Xu, and Sergey Levine. "Probabilistic Model-Agnostic Meta-Learning". In: *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. NIPS'18. Montréal, Canada, 2018, pp. 9537–9548.

[264] Zhenguo Li et al. *Meta-SGD: Learning to Learn Quickly for Few-Shot Learning*. 2017. arXiv: 1707.09835.

[265] Thomas Elsken et al. "Meta-Learning of Neural Architectures for Few-Shot Learning". In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 12362–12372. DOI: 10.1109/CVPR42600.2020.01238.

[266] Muhammad Abdullah Jamal and Guo-Jun Qi. "Task Agnostic Meta-Learning for Few-Shot Learning". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[267] Andrei A. Rusu et al. "Meta-Learning with Latent Embedding Optimization". In: *International Conference on Learning Representations*. 2019. URL: https://openreview.net/forum?id=BJgklhAcK7.

[268] Qianru Sun et al. *Meta-Transfer Learning for Few-Shot Learning*. 2018. arXiv: 1812.02391.

[269] Timothy Hospedales et al. "Meta-Learning in Neural Networks: A Survey". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PP (May 2021), pp. 1–1. DOI: 10.1109/TPAMI.2021.3079209.

[270] Hui Xu et al. "Unsupervised meta-learning for few-shot learning". In: *Pattern Recognition* 116 (2021). ISSN: 0031-3203. DOI: 10.1016/j.patcog.2021.107951.

[271] Thomas Adler et al. *Cross-Domain Few-Shot Learning by Representation Fusion*. 2020. arXiv: 2010.06498.

[272] Cong Zhao et al. "Exploration Across Small Silos: Federated Few-Shot Learning on Network Edge". In: *IEEE Network* (2021), pp. 1–7. DOI: 10.1109/MNET.111.2100329.

[273] Chenyou Fan and Jianwei Huang. "Federated Few-Shot Learning with Adversarial Learning". In: *2021 19th International Symposium on Modeling and Optimization in Mobile, Ad hoc, and Wireless Networks (WiOpt)*. 2021, pp. 1–8. DOI: 10.23919/WiOpt52861.2021.9589192.

[274] Xiao Liu et al. "Self-supervised Learning: Generative or Contrastive". In: *IEEE Transactions on Knowledge and Data Engineering* (2021), pp. 1–1. DOI: 10.1109/TKDE.2021.3090866.

[275] Yixin Liu et al. *Graph Self-Supervised Learning: A Survey*. 2022. arXiv: 2103.00111.

[276] Lirong Wu et al. "Self-supervised on Graphs: Contrastive, Generative, or Predictive". In: (2021). arXiv: 2105.07342.

[277] Yuandong Tian, Xinlei Chen, and Surya Ganguli. "Understanding self-supervised Learning Dynamics without Contrastive Pairs". In: (2021). arXiv: 2102.06810.

[278] Alireza Makhzani et al. *Adversarial Autoencoders*. 2016. arXiv: 1511.05644.

[279] Vincent Dumoulin et al. *Adversarially Learned Inference*. 2017. arXiv: 1606.00704.

[280] Satoshi Iizuka, Edgar Simo-Serra, and Hiroshi Ishikawa. "Globally and Locally Consistent Image Completion". In: *ACM Transactions on Graphics* 36.4 (2017). DOI: 10.1145/3072959.3073659.

[281] Alejandro Newell and Jia Deng. "How Useful Is Self-Supervised Pretraining for Visual Tasks?" In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 7343–7352. DOI: 10.1109/CVPR42600.2020.00737.

[282] Dan Hendrycks et al. "Using Self-Supervised Learning Can Improve Model Robustness and Uncertainty". In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach et al. Vol. 32. Curran Associates, Inc., 2019. URL: https://proceedings.neurips.cc/paper/2019/file/a2b15837edac15df90721968986f7f8e-Paper.pdf.

[283] Bram van Berlo, Aaqib Saeed, and Tanir Ozcelebi. "Towards Federated Unsupervised Representation Learning". In: *Proceedings of the Third ACM International Workshop on Edge Systems, Analytics and Networking*. 2020, pp. 31–36. DOI: 10.1145/3378679.3394530.

[284] Aaqib Saeed et al. "Federated Self-Supervised Learning of Multisensor Representations for Embedded Intelligence". In: *IEEE Internet of Things Journal* 8.2 (2021), pp. 1030–1040. DOI: 10.1109/JIOT.2020.3009358.

[285] Weiming Zhuang, Yonggang Wen, and Shuai Zhang. "Divergence-aware Federated Self-Supervised Learning". In: *International Conference on Learning Representations*. 2022. URL: https://openreview.net/forum?id=oVE1z8NlNe.