Agent-based Modeling in the Edge Continuum using Swarm Intelligence

Melanie Schranz, Kseniia Harshina, Peter Forgacs
Lakeside Labs
Klagenfurt, Austria
{schranz, harshina, forgacs}@lakeside-labs.com

Fred Buining
HIRO MicroDataCenters
Voorburg, Netherlands
fred.buining@hiro-microdatacenters.nl

***Abstract*—The edge continuum presents a dynamic and evolving paradigm in the future's world of computing, offering a versatile and efficient solution for a wide range of applications and industries. The edge infrastructure is more challenged in its stability and performance because of more stringent latency and autonomy requirements, distribution across multiple sites, their local limited size, multi-tenancy and multi-operators, local management, with components being concurrent and asynchronous. This paper introduces an innovative framework that combines agent-based modeling and swarm intelligence to address complex challenges such as resource allocation, workload scheduling, and data management in the edge continuum. This framework, at the core of the architecture, enhances edge autonomy, reduces latency, improves energy efficiency, and optimizes cloud connectivity by applying agent-based modeling. By integrating autopoietic characteristics like self-organization, regeneration, and regulation, the system dynamically adapts to changing conditions. Two candidate algorithms, the hormone algorithm and ant algorithm, emulate decentralized decision-making processes observed in nature. The paper reviews related work in swarm intelligence for network optimization and emphasizes the need for distributed, agent-based solutions. This research paves the way for robust, adaptive, and scalable systems in the complex edge environment, promising emergent behaviors and enhanced efficiency. In this position paper, we propose the edge continuum with its characteristics and limitations as a novel field of application for swarm intelligence by conceptually proposing agent-based modeling and simulation.**

***Index Terms*—Swarm Intelligence, Bio-inspired Algorithm, Edge Continuum, Agent-Based Modeling**

I. Introduction

The emergence of local processing capacity at the edge is driven by numerous advantages essential for upcoming processing tasks. These benefits encompass heightened security and reliability, alongside reduced latency and energy consumption. The management of the edge infrastructure, the so-called edge continuum, presents a dynamic computing landscape. Within the edge continuum, for which we consider a mesh of Edge Micro Data Centers (EMDCs) in this paper (see Figure 1), intelligence is spread across the edges forming a distributed environment. This will make the edge more autonomous and fine-grained in local decision making within a regional context and make it more independent from a central coordination point. This is especially necessary, if we talk about real-time applications such as autonomous driving or monitoring and control of smart grids. The edge infrastructure is more challenged in its stability and performance because of more stringent latency and autonomy requirements, distribution across multiple sites, its local limited size, multi-tenancy and multi-operators, local management, with components being concurrent and asynchronous. This challenge to edge infrastructures is growing rapidly due to the increasing i) number of connected devices and their data-producing and data-consuming capabilities, ii) intelligence embedded in edge devices, iii) atomization of monolithic applications, iv) scale, speed, and complexity of edge device interactivity in a zero-trust environment. Resource allocation, workload scheduling, and data management are challenges that increase in the complexity of the edge orchestration and edge-cloud interaction (see Figure 1 for a schematic architecture provided by the ACES project).

This position paper introduces a conceptual, but novel framework that combines agent-based modeling and swarm intelligence as an emergent orchestrating mechanism to address these complexities. Agent-based modeling and swarm intelligence are known for providing advantages in simulating complex systems with autonomous entities including adaptability, scalability and robustness. They utilize collective decision-making processes as observed in nature by swarms of insects, fish or birds [1]. Central to our approach is the integration of these autopoietic characteristics that include the emergent intelligence of self-organization, regeneration, and regulation. These characteristics enable the system to dynamically adapt and optimize in response to changing conditions. AI-driven optimization methods (including swarm intelligence) in cloud infrastructure are successfully being researched (see Section VI for more details). Among recent notable examples of utilization of swarm intelligence to optimize complex systems, is the work of Schranz et al. [2], where authors successfully utilize bottom-up job shop scheduling applying swarm intelligence algorithms for optimizing a large production plant. Thus, we propose the edge continuum with its characteristics and limitations as a novel field of application for swarm intelligence.

This framework is at the core of the architecture, required to manage the edge infrastructure, EMDCs capable of processing big data and AI at the edge-to-edge environment independent from a distant cloud. Key to our conceptual approach is the use of swarm agents, representing demand and supply entities.

 ISBN: 978-1-68558-153-4

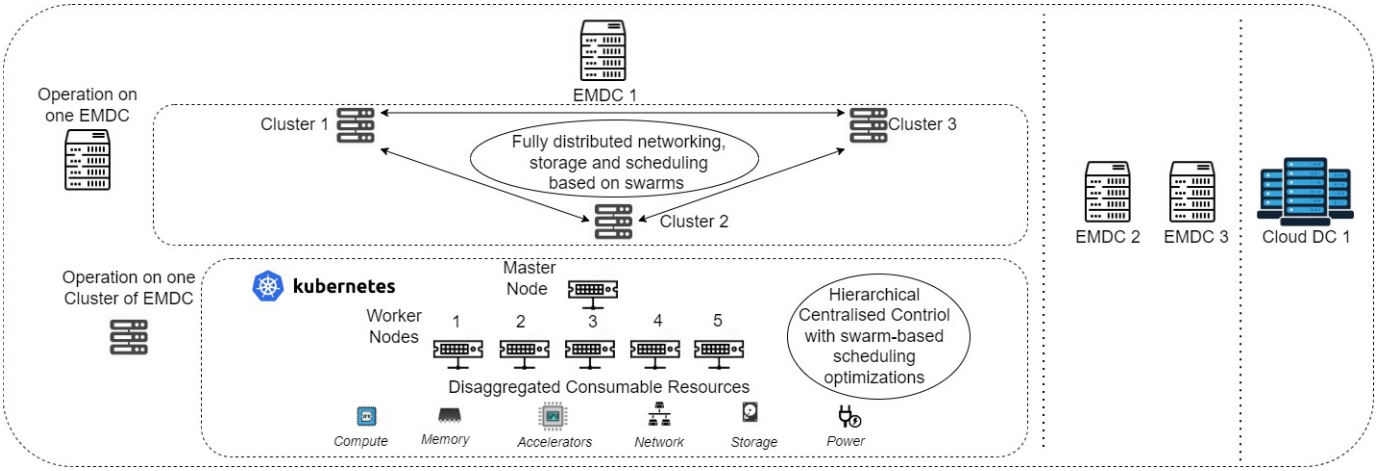

Fig. 1. Schematic architecture demonstrating the inter-edge resource allocation in clusters (nodes, pool of resources - see Section III - and the overall edge-cloud interaction.

Demand swarm agents represent workload behaviors at the pod level, ensuring pod-level optimization. On the other hand, supply swarm agents represent node dynamics. These agents collaborate within an EMDC environment, orchestrating processes such as workload placement, storage management, and caching optimization. Exemplary swarm algorithms, in this paper the hormone and ant algorithms are utilized in order to accomplish the desired functionality of the system. For example, demand swarm agents deploy synthetic hormones to communicate their requirements and priorities. Supply swarm agents, detect these hormones to make informed allocation decisions. The ant algorithm dynamically optimizes workload-node-assignments by simulating the foraging behavior of ants, depositing pheromones to guide subsequent decisions.

The paper follows the following structure: Section 1 introduces the problem and outlines specific challenges of the research process. In Section 2, we explore the edge continuum setting. In Section 3, we discuss the agent-based model, and address challenges specific to the infrastructure and interactions in the EMDCs. In Section 4, we evaluate candidate swarm algorithms, the hormone and ant algorithms on a conceptual basis. We conclude the paper with Section 5 by reviewing the related work and giving an outline on future work in Section 6.

II. The Edge Continuum

Industry surveys show that edge infrastructure is a driver for new initiatives and business operations. According to recent studies, the percentage of respondents that have or will implement edge locations within the next three years raised from 55% last year to 87% this year [3]. According to the EU, a decentralized intelligently orchestrated edge infrastructure (hardware and software) is needed to support platforms, data spaces and applications, e.g., an Industrial Metaverse utilizing a combination of cloud, edge and IoT to enable a wide range of new solutions to transform processes, automate operations, and launch new products and services. At the edge where traffic patterns are becoming sudden and unpredictable, more traffic will be handled in far edge or metro edge data centers, micro edge data centers embedded in metro networks [4].

The edge is created from a mesh of EMDCs that are completely composable, customizable, heterogeneous, and therefore different in capacity and capabilities. An EMDC will supply resources; nodes with servers, accelerators, storage, and networking capabilities. While the application owners and edge devices will demand certain resources, capabilities, and performances, the orchestration of matching demand and supply are made dynamically and decentralized in a bottom-up approach. This will allow the EMDC to be configured and sized to satisfy 1) local autonomy: the demands of the local edge clients, and 2) regional elasticity: the demands upon groups of EMDCs caused by fluctuating local demand and mobile demand traveling along several EMDCs in a region. This means that each EMDC needs to be:

- aware of its hardware configuration, hardware capabilities, software services (supply) and the local and regional requests for services (demand);
- self-intelligent and autonomous (autopoietic AI) in matching local and regional supply and demand;
- efficient in the execution of its services and self-intelligence.

Currently, hardware-based composability of an EMDC means that the nodes can be of any type, such as Central Processing Unit (CPU), Field Programmable Gate Array (FPGA), Graphical Processing Unit (GPU), or Non-Volatile Memory express (NVMe).

III. Creating an Agent-Based Model for Optimizing the Edge Continuum

When shaping the edge continuum to an agent-based system, we analyze a group of possible swarm agents and their attributes. In this context, we need to determine the eligibility of an entity to serve as a member of the EMDC swarm [5]. The swarm can exhibit homogeneity (with all agents being of the

 ISBN: 978-1-68558-153-4

same kind, like numerous pods) or heterogeneity (comprising agents of various types, such as pods and resources). For an entity to qualify as a swarm member, it should possess the capacity to effectively function within a swarm. This entails the presence of a significant number of other swarm members (for instance, a single instance of an FPGA, existing in isolation, would not make a suitable swarm member). Additionally, the entity should exhibit an appropriate degree of abstraction to facilitate modeling, possess the capability to sense dynamic information from the immediate environment, respond to information originating from the local vicinity (such as making decisions), and be logically coherent and comprehensible, fostering trust in the proposed solution [1].

A. Modeling Agents in the Edge Continuum

Our agent-based approach introduces two distinct types of swarm agents: demand swarm agents and supply swarm agents. These agents collaborate within an EMDC environment, orchestrating processes such as pod placement, storage management, and caching optimization. The model for the problem consists of an edge continuum with resources, queues, pods, and processes.

1) Demand Swarm Agents: An application is split into a set of services $\mathfrak{S}$ that are represented as a set of related pods $P^s = \{p_1^s, p_2^s, \dots\}$ with s as the specific service. Each service s is defined by a compilation of resources R^s which prescribes the processing steps necessary to compute the individual pods. The pod p_j^s can choose which of the suitable nodes N_i^n to use for each necessary process step P^r.

2) Supply Swarm Agents: The EMDC $\mathfrak{E}$ contains several sets of nodes or nodes, consisting of different types of resources $N^r = \{N_1^r, N_2^r, \dots\}$, where r is/are the resource type(s). A node with different resources presents a typical EMDC node, whereas a node with a single resource presents, e.g., a CPU that is part of a pool of resources. In the course of this work, we consider the following resources along with their respective capacities: CPU, FPGA, RAM, and NVMe. Each resource N_i^r has a queue Q_i^r.

3) Agent Collaboration and Self-Organization: The interaction between demand swarm agents and supply swarm agents is orchestrated through swarm intelligence algorithms. Demand swarm agents autonomously seek out the most suitable node for workload placement, while supply swarm agents determine the optimal workload to process based on available resources and capacity. This collaborative decision-making process enables the system to efficiently allocate workloads to nodes, optimizing processing, latency, and resource utilization.

Our agent-based model is designed to exhibit autopoietic characteristics, fostering self-organization, regeneration, and regulation within the edge continuum. As demand and supply agents interact and adapt to changing workloads and resource availability, the system as a whole displays emergent behaviors that contribute to its resilience and efficiency.

IV. Challenges in Modeling Agents for the EMDC

In the agent-based modeling of an EMDC, we face a set of challenges that need to be considered in the modeling process.

A. Pool of Resources

Additionally to the nodes in an EMDC, we consider a pool of resources that presents an innovation to the current definitions of the edge continuum. This means that besides the processing capabilities in a node (that is a constitution of multiple resources), single resources can be requested for pod processing. This pool of resources is part of the EMDC and can be consulted by the edge(-cloud) management as requested. Such a pool mainly prevents resource limits, increased latencies, and stability of the performance of other pods, as their assigned resources are not tapped. Currently, the Compute Express Link (CXL) is being implemented in CPUs (Intel, AMD), in memories and storage (Samsung) and the PCIe switches are expected in 2025. Besides the hardware development, the biggest challenge currently is how these pools of resources can be orchestrated.

B. Application Types

For the different services, we can differ between the three application types that come with diverse requirements in their response time.

- The **Long-Running Applications (LRAs)** instantiate long-standing pods to enable iterative computations in memory or unceasing request-response. LRAs include processing frameworks (e.g., Storm [6], Flink [7], Kafka streams [8]), latency-sensitive database applications (e.g., HBase [9] and MongoDB [10]), and data-intensive in-memory computing frameworks (e.g., TensorFlow [11]).
- **Batch processing** is typically used when you have a large amount of data that needs to be processed all at once, and when the results of that processing can be stored and used later. Data is typically processed on a schedule or at regular intervals. There are two types of batch processing: Regular returning requests, and opportunistic requests with little to no SLA (Service Level Agreement).
- **Stream processing** also deals with large volumes of data, but the data needs to be processed in real-time.

Future workloads will become even more complex with LRAs, batches, and stream processes being interconnected. Therefore, it will be challenging to categorize an application and tune its agents accordingly.

C. Relationships among Pods

The demand swarm agents are related pods P^s split from a specific service s. These pods can have several relations with each other. There can be different needs, e.g., that they need to be processed in parallel or that they depend on each other. Additionally, if one pod is too slow, the current system creates more pods to reach the given response times of the specific service s. Currently, these relationships are not used in the scheduler and orchestration optimization. For example, placing interacting services closer together can significantly enhance their performance, e.g., i) if there are multiple services with microservices that frequently interact, it is advisable to locate the microservices of one service within the same region to improve performance, ii) for pods that are heavily dependent

 ISBN: 978-1-68558-153-4

on a database, it is best to place them near the database to reduce latency and improve overall performance.

V. Candidate Algorithms

In this section, we introduce two candidate algorithms for the edge continuum. These algorithms adopt a bottom-up approach, by modeling real-world entities as agents, including the attributes that enable them to interact with each other and their environment. By applying swarm intelligence principles to these agents, we enhance their capabilities to manage the complexity in the edge continuum. This allows them to make context-aware decisions by drawing from both local and global information. This approach embraces decentralized decision-making, promising effective resource management in the complex edge(-cloud) continuum.

A. Hormone Algorithm

Artificial hormone systems draw inspiration from the biological endocrine system, which regulates various metabolic processes within our bodies [12], [13]. This creates a self-organizing system characterized by scalability, adaptability, and robustness. In our simulation, supply swarm agents correspond to nodes within the continuum, and demand swarm agents represent pods seeking optimal node placement.

Demand swarm agents release synthetic hormones into the environment based on their resource requirements and preferences. These hormones carry information about the demands and priorities of the pods. Supply swarm agents, representing nodes, detect these hormones and adjust their behavior accordingly. Nodes release their own hormones indicating resource availability and capacity.

The concentration of hormones guides demand swarm agents toward nodes that match their requirements, fostering autonomous and informed decision-making. The communication of synthetic hormones replaces traditional centralized control mechanisms with decentralized coordination, allowing the system to adapt to pod variations and resource fluctuations.

The underlying principle is inspired by the use of artificial hormones for reorganizing agents in self-organizing systems for technical applications [14], [15], which can be extended to the dynamic edge environment. In our framework, we will implement the artificial hormone system as a software layer distributed across the processing nodes within the edge continuum as inspired by the applications in production plants (see Elmenreich et al. [16] for more details). The hormone algorithm used for optimization in the edge continuum can be dissected into six key mechanisms:

Production: Supply swarm agents, representing nodes, produce hormones in response to the number of demand swarm agents, pods, in the EMDC. Nodes that are currently processing fewer pods produce more hormone. Each node as well as a pool of resources (e.g., CPU, storage) may produce a distinct type of hormone with

$$H^r = \frac{1}{|Q_i^r| + \beta} \tag{1}$$

where H_i^r is the hormone corresponding to the the node N_i^r, β is a smoothing factor, and $|Q_i^r|$ is the number of waiting workloads in the EMDC for the node N_i^r.

Evaporation: The hormone levels at each node gradually decrease over time through a process of evaporation, controlled by a parameter α given with

$$H_{i,t+1}^r = H_{i,t}^r \cdot (1 - \alpha) \tag{2}$$

where $H_{i,t+1}^r$ and $H_{i,t}^r$ represent the state of hormone at the node N_i^r before and after a discrete evaluation step.

Diffusion: Hormones diffuse from one node to another based on the compilation of resources per pod that also connects the resources similar to hormone propagation in biological systems. Hormones move upstream, following the reverse of this resource graph by calculating

$$\Delta H = H_i^r \cdot \gamma \tag{3}$$

$$H_i^r - = \Delta H \tag{4}$$

where ΔH is the amount of hormone moving upstream from the node N_i^r, and γ is a parameter setting the motility of hormone.

The link strength $l^{r,p}$ between two nodes N_i^r and N_j^p is equivalent to the number of compilations of resources R_t containing processes P^r and P^p in direct succession. Each node connected upstream receives a proportional part of the upstream hormone with

$$H_j^p + = \Delta H \frac{l^{r,p}}{\sum_c l^{r,c}} \tag{5}$$

where $\sum_c l^{r,c}$ represents the sum of all upstream links from P^r.

Diffusion through pod movement: When pods move between nodes within the EMDC, e.g., due to a lack or loss of resources in a node, they carry hormones with them, influencing the hormone levels at both the initial and destination nodes.

$$\Delta H = H_i^r \cdot \delta \tag{6}$$

$$H_i^r - = \delta H \tag{7}$$

$$H_j^p + = \delta H \tag{8}$$

where ΔH defines the amount of hormone that moves with the pod, calculated from the amount of available hormone H_i^r at the node N_i^r.

Attraction: Pods are attracted by the nodes whose processing capabilities match the pods' requirements from the corresponding compilation of resources. The amount of attraction decreases exponentially based on the order of the node. The attraction force is applied to pods as soon as they enter the

processing queue Q_i^r and it can make the pod move towards a distinct node.

$$attraction = \sum_{i,r} H_i^r \cdot \varepsilon^n \tag{9}$$

where H_i^r is the hormone amount at a node that is n edges away, and ε is a factor < 1 defining the degradation of the hormone attraction over edge distance in the graph G.

Each mechanism comes with a parameter indicating the strength of each part, that is evaporation rate (α), hormone production factor (β), upstream diffusion factor (γ), hormone distribution factor (δ), and attraction factor (ε). A possible configuration of these parameters is stated in [16]. Due to the interaction between each of the mechanisms forming feedback control loops, the algorithm can operate with a broad set of possible parameter settings.

B. Ant Algorithm

Ant algorithms draw inspiration from the decentralized foraging behavior of ants, a natural phenomenon where ants can efficiently find near-optimal paths to food sources without relying on global knowledge. They achieve this by leaving pheromone trails to communicate with other ants. In our simulation within the edge continuum, this concept will be applied to optimize the allocation and processing of pods by supply swarm agents, analogous to ants, representing nodes within the continuum.

Here's how the ant algorithm is adapted to the edge continuum:

Trail Following: In our context, we frame the allocation of pods as a routing problem in the edge continuum. Pods probabilistically select the next suitable node N_i^n from the set of potential nodes N^n based on both local pheromone values associated with that node and a local heuristic considering the node's current pod, which can be assessed by metrics like queue length or resource utilization. The probability $P_{i,j}$ of selecting node N_i^n is computed as shown in Equation 10.

$$P_{i,j} = \frac{\tau_{i,j,d} + \alpha\eta_{i,j}}{1 + \alpha(N_i - 1)} \tag{10}$$

with

$$\eta = 1 - \frac{q_{i,j}}{\sum q}. \tag{11}$$

In this equation, η represents the relative queue length of node with N_i as the number of possible nodes. The parameter α allows for fine-tuning the influence of pheromone τ (see update rules below) versus the local pod heuristic. In our adaptation, the destination d corresponds to the next step in the pod's compilation of resources within the EMDC, rather than a specific destination node.

Trail Laying: Pheromone values are updated after a pod has been processed on a node within the EMDC. However, unlike traditional ant algorithms where backward ants are used to update pheromone values, we utilize communication and coordination among nodes within the continuum. Each pod maintains a memory of the processing, effectively measuring the time it waited for resources. When a pod moves from one node to another, this information is used to update the pheromone values.

For a chosen node N_x^n, the pheromone value is updated as follows:

$$\tau_{x,d} \leftarrow \tau_{x,d} + r(1 - \tau_{x,d}) \tag{12}$$

For all potential nodes N_n^n that were not chosen, the pheromone values are updated according to

$$\tau_{n,d} \leftarrow \tau_{n,d} - r\tau_{n,d}. \tag{13}$$

The reinforcement r depends on the processing time of the pod, which reflects the waiting time and resource utilization. This approach ensures that nodes with shorter pod processing times and lower resource utilization become more attractive for incoming workloads.

Evaporation: Periodically, pheromone values are subject to evaporation with a rate p. This process simulates the natural fading of pheromone trails and helps remove paths that may have become less optimal due to changes in resource availability or demand (Equation 14).

$$\tau(t+1) = \tau(t)(1 - p) \tag{14}$$

This adaptation effectively models and optimizes the allocation and processing of pods within the EMDC, drawing inspiration from the decentralized behavior of ants.

VI. Related Work

Next-Generation Networks (NGN) are growing fast, and this rapid growth is becoming more and more demanding to optimize resource management in cloud computing, edge computing, and edge-cloud computing. As big data analytics is gaining size, optimization is becoming problematic, because those optimizers, which seek an exact global optimum, can have an exponentially growing complexity. Some examples of optimization problems in big data analytics that can exhibit expensive computational complexity:

Combinatorial Feature Selection: When dealing with a large number of features (variables) in a dataset, selecting the optimal subset of features for a machine learning model can be computationally intensive. The number of possible feature combinations grows exponentially with the number of features, leading to exponential complexity [17], [18].

Clustering in High-Dimensional Spaces: In high-dimensional spaces, clustering algorithms like k-means can become computationally expensive. As the number of dimensions increases, the data points tend to become more distant from each other, making it challenging for clustering algorithms to identify meaningful clusters. This phenomenon is often referred to as the curse of dimensionality [19], [20].

Optimizing Distributed Systems: Optimizing the allocation of computing resources in distributed machine learning systems for big data analytics can be computationally expensive. These systems often involve multiple nodes and parallel processing

of large data sets. Ensuring efficient resource allocation to reduce training time and resource waste is a challenging optimization problem [21], [22].

Pham et al. [23] do an in-depth review of the implementation of swarm intelligence for NGN, and state the advantage of swarm intelligence in guaranteed convergence, robustness, near-optimal solution, and computationally-traceability. There are several research works addressing the major issues in NGN using swarm intelligence. The way they are approaching is by creating a random set of solutions. This set of candidate solutions is improved iterations by iterations optimizing the objective function, which quantifies the goodness of a solution. The review also mentions possible swarm intelligence implementations for spectrum management and resource allocation, wireless caching and edge computing, network security, and miscellaneous issues. The following lines of this section present some even more recent research done in the optimization of task offloading in edge computation than the ones presented in the review.

In smart homes, to minimize the energy consumption of a residential consumer-centric load-scheduling, Lin & Hu [24] proposed a constrained Particle Swarm Optimization (PSO) algorithm, where the possible solutions are modeled as agents. Feng et al. [25] also describe a task offloading strategy, which is able to reduce the energy consumption, the time latency and the service price in mobile edge computing, however, the strategy is to use a Grey Wolf Optimizer (GWO), Whale Optimization Algorithm (WOA) and GWO – WOA where the agents are the percentages of how much of a mobile device's task is computed locally on the mobile device (because a task can also be partially offloaded to the edge server). This means that whenever there is a change in the tasks or the mobile devices, the algorithm needs to compute the optimal solution with a new set of input. Lee et al. [26] provides a swarm intelligence algorithm, an Artificial Bee Colony (ABC) algorithm for the allocation of a given task set to a given edge server set and a cloud, where the solutions from the solution population are interacting with each other. A relatively recent work, Mahenge & Sanga [27] presents a strategy to offload resource-intensive tasks in mobile edge computing energy-efficiently using a hybrid approach (PSO and GWO), where the algorithm gathers the information about the tasks and servers and then calculates the optimal offloading strategy. Bacanin et al. [28] perform energy optimization in 5G-enabled edge nodes using PSO which first has to obtain as input all the data about the tasks and edges. Attiya et al. [29] aims to tackle the problem of IoT application task scheduling. It uses the Manta Ray Foraging Optimization (MRFO) combined with Salp Swarm Algorithm (SSA) which is also initialized with a set of N solutions. In Singh et al. [30], the authors write all available resources into an availability list. On this list, a swarm algorithm (Ant Colony Optimization, ACO) is executed for searching an optimized (centralized) solution for resource allocation and scheduling. Another approach is presented in de Melo et al. [31]. Here, the focus is on a review of several methodologies to parallelize swarm algorithms on parallel hardware to increase execution performance. The aim is to accelerate finding an optimal solution to a problem which is then mostly applied in a centralized manner. No decentralized agent-based approach is revealed in this work.

Although the proposed solutions in the literature apply different swarm intelligence algorithms, they are executed centrally. Typical problems that arise from this approach are single point of failure, higher computational effort, and lack of dynamicity to occurring changes in the environment or incoming demands. Therefore we propose an optimization from the bottom-up, using swarm intelligence literally with interacting embodied agents that make decisions based on local information. These are then the algorithms that are robust, adaptive, and scale due to their distributed characteristic leading to a real emergent behavior of a complex system.

To the best of our knowledge, no one has ever tried an agent-based approach in the edge-cloud domain where resources and requests are regarded as agents, and scheduling along with relevant objectives (utilization, low latency, energy efficiency, etc.) are considered emergent properties of the agent's local decision making and interaction (autopoiesis). This will direct the NGN of EMDCs on the edge to a powerful, self-organized network, where we can generate a main contribution towards the scheduling of a resource pool and the dynamics of pod arrivals.

VII. Conclusion and Future Work

The management of the edge continuum presents a multi-faceted computing landscape that continues to grow in complexity. Our paper introduces a novel conceptual framework that leverages agent-based modeling and swarm intelligence to address these complexities, focusing on enhancing edge autonomy, reducing latency, improving energy efficiency, and optimizing cloud connectivity.

We propose utilizing swarm agents to represent demand and supply entities within an Edge Micro Data Center (EMDC) environment. Demand swarm agents optimize at the pod level, while supply swarm agents manage node dynamics. The application of swarm algorithms, including hormone and ant algorithms, facilitates intelligent workload placement, storage management, and caching optimization.

As a next step, we will elaborate on the efficiency of the proposed candidate algorithms using a simulation approach based on an abstracted version of the edge continuum using SwarmFabSim [32], a NetLogo implementation, as inspiration. Additionally, real-world implementation and validation of the framework will be essential to demonstrate its practical effectiveness in managing the dynamic edge(-cloud) landscape. This is a step that will need some workarounds first, as the hardware to realize resource pools on the edge is still in the development phase.

Acknowledgement

This work was performed in the course of the EU-project ACES[1] supported by European Union's Horizon Europe re-

[1] https://www.aces-edge.eu/

search and innovation programme under the grant agreement No. 101093126 (HORIZON-CL4-2022-DATA-01-02).

REFERENCES

[1] M. Schranz, G. A. Di Caro, T. Schmickl, W. Elmenreich, F. Arvin, A. Şekercioğlu, and M. Sende, "Swarm intelligence and cyber-physical systems: concepts, challenges and future trends," *Swarm and Evolutionary Computation*, vol. 60, p. 100762, 2021.

[2] M. Schranz., M. Umlauft., and W. Elmenreich., "Bottom-up job shop scheduling with swarm intelligence in large production plants," in *Proceedings of the 11th International Conference on Simulation and Modeling Methodologies, Technologies and Applications*, pp. 327–334, INSTICC, SciTePress, 2021.

[3] B. Kleyman, B. Gillooly, and K. Letourneau, "The 2021 state of the data center report," *AFCOM*, p. 20, 2021. https://datacenterworld.com/sites/default/files/AFCOM_State%20of%20the%20Data%20Center_FINAL_2021_5-10-21.pdf [Online; accessed 8-March-2024].

[4] P. Fetterolf, "Next-generation metro network and edge computing architectures," *ACG Research*, p. 10, 2021. https://www.acgcc.com/media/reports/files/ACG_Research_Next-Generation_Metro_Network_and_Edge_Computing_Architectures_2021_1.pdf [Online; accessed 8-March-2024].

[5] M. Schranz, M. Umlauft, M. Sende, and W. Elmenreich, "Swarm robotic behaviors and current applications," *Frontiers in Robotics and AI*, vol. 7, p. 36, 2020.

[6] A. Storm. https://storm.apache.org/. [Online; accessed 8-March-2024].

[7] A. Flink. https://flink.apache.org/. [Online; accessed 8-March-2024].

[8] A. Kafka. https://kafka.apache.org. [Online; accessed 8-March-2024].

[9] A. HBase. https://hbase.apache.org/. [Online; accessed 8-March-2024].

[10] MongoDB. https://www.mongodb.com/. [Online; accessed 8-March-2024].

[11] TensorFlow. https://www.tensorflow.org/. [Online; accessed 8-March-2024].

[12] A. Sobe, W. Elmenreich, T. Szkaliczki, and L. Böszörmenyi, "Seahorse: Generalizing an artificial hormone system algorithm to a middleware for search and delivery of information units," *Computer Networks*, 2015.

[13] A. Turing, "The chemical basis of morphogenesis," *Philosophical Transactions of the Royal Society of London, Series B, Biological Sciences*, vol. 237, no. 641, pp. 37–72, 1952.

[14] W. Elmenreich, R. D'Souza, C. Bettstetter, and H. de Meer, "A survey of models and design methods for self-organizing networked systems," in *International Workshop on Self-Organizing Systems*, pp. 37–49, Springer, 2009.

[15] W. Elmenreich and H. de Meer, "Self-organizing networked systems for technical applications: A discussion on open issues," in *International Workshop on Self-Organizing Systems*, pp. 1–9, Springer, 2008.

[16] W. Elmenreich, A. Schnabl, and M. Schranz, "An artificial hormone-based algorithm for production scheduling from the bottom-up.," in *International Conference on Agents and Artificial Intelligence*, pp. 296–303, 2021.

[17] L. Yu and H. Liu, "Feature selection for high-dimensional data: A fast correlation-based filter solution," in *Proceedings of the 20th international conference on machine learning*, pp. 856–863, 2003.

[18] U. M. Khaire and R. Dhanalakshmi, "Stability of feature selection algorithm: A review," *Journal of King Saud University-Computer and Information Sciences*, vol. 34, no. 4, pp. 1060–1073, 2022.

[19] C. C. Aggarwal, A. Hinneburg, and D. A. Keim, "On the surprising behavior of distance metrics in high dimensional space," in *Proceedings of the 8th International Conference on Database Theory*, pp. 420–434, Springer, 2001.

[20] A. C. Benabdellah, A. Benghabrit, and I. Bouhaddou, "A survey of clustering algorithms for an industrial context," *Procedia computer science*, vol. 148, pp. 291–302, 2019.

[21] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: Cluster computing with working sets," in *2nd USENIX Workshop on Hot Topics in Cloud Computing*, 2010.

[22] J. Verbraeken, M. Wolting, J. Katzy, J. Kloppenburg, T. Verbelen, and J. S. Rellermeyer, "A survey on distributed machine learning," *ACM Computing Surveys*, vol. 53, no. 2, pp. 1–33, 2020.

[23] Q. Pham, D. Nguyen, S. Mirjalili, D. Hoang, D. Nguyen, P. Pathirana, and W.-J. Hwang, "Swarm intelligence for next-generation networks: Recent advances and applications," *Journal of Network and Computer Applications*, vol. 191, p. 103141, 2021.

[24] Y.-H. Lin and Y.-C. Hu, "Residential consumer-centric demand-side management based on energy disaggregation-piloting constrained swarm intelligence: Towards edge computing," *Sensors*, vol. 18, no. 5, p. 1365, 2018.

[25] S. Feng, Y. Chen, Q. Zhai, M. Huang, and F. Shu, "Optimizing computation offloading strategy in mobile edge computing based on swarm intelligence algorithms," *EURASIP Journal on Advances in Signal Processing*, vol. 7, no. 36, pp. 1–24, 2021.

[26] C. Lee, Y. Huo, S. Zhang, and K. Ng, "Design of a smart manufacturing system with the application of multi-access edge computing and blockchain technology," *IEEE Access*, vol. 8, pp. 28659–28667, 2020.

[27] M. Mahenge, C. Li, and C. Sanga, "Energy-efficient task offloading strategy in mobile edge computing for resource-intensive mobile applications," *Digital Communications and Networks*, vol. 8, no. 6, pp. 1048–1058, 2022.

[28] N. Bacanin, M. Antonijevic, T. Bezdan, M. Zivkovic, K. Venkatachalam, and S. Malebary, "Energy efficient offloading mechanism using particle swarm optimization in 5g enabled edge nodes," *Cluster Computing*, vol. 26, pp. 587–598, 2023.

[29] I. Attiya, M. Elaziz, L. Abualigah, T. Nguyen, and A. El-Latif, "An improved hybrid swarm intelligence for scheduling iot application tasks in the cloud," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 9, pp. 6264–6272, 2022.

[30] H. Singh, A. Bhasin, and P. R. Kaveri, "Qras: Efficient resource allocation for task scheduling in cloud computing," *SN Applied Sciences*, vol. 3, pp. 1–7, 2021.

[31] B. A. de Melo Menezes, H. Kuchen, and F. Buarque de Lima Neto, "Parallelization of swarm intelligence algorithms: Literature review," *International Journal of Parallel Programming*, vol. 50, pp. 1–29, 2022.

[32] M. Umlauft, M. Schranz, and W. Elmenreich, "Swarmfabsim: A simulation framework for bottom-up optimization in flexible job-shop scheduling using netlogo," in *Proceedings of the 12th International Conference on Simulation and Modeling Methodologies, Technologies and Applications*, pp. 271–279, 2022.

 ISBN: 978-1-68558-153-4