Computation Offloading in SAGIN: Architecture, Optimization, and DRL Approaches

IM TAEUK

Abstract

Space–Air–Ground Integrated Network (SAGIN) is considered one of the most promising architectures for next-generation networks. SAGIN enables distributed computing in global areas, making it particularly useful for computation offloading in remote or restricted areas. Despite this potential, existing research on SAGIN remains limited and most prior studies focus on specific aspects of SAGIN. This paper provides a comprehensive analysis of recent SAGIN architectures and computation offloading strategies, in recent research, with respect to optimization objectives such as latency and power consumption, as well as the approaches used to address them. In particular, we examine how deep reinforcement learning (DRL)-based approaches, which have recently gained significant attention, are applied to optimize offloading strategies of SAGIN in highly dynamic environments. Through this comprehensive survey, this work offers overview of a deep understanding of SAGIN and emerging optimization approaches to computation offloading.

I. Introduction

SAGIN is widely regarded as one of the highly attractive architectures for next-generation networks. By integrating ground, aerial, and space components, SAGIN augments conventional integrated satellite–terrestrial networks (ISTNs) with an additional aerial layer. This aerial layer can be realized by various platforms, such as unmanned aerial vehicles (UAVs), civil aircrafts (CAs), or balloons depending on system requirements and deployment conditions. Through this multi-layer integration, SAGIN can provide global coverage, distributed computing capability, and improved energy efficiency for computation while maintaining low-latency. Owing to these characteristics, SAGIN is particularly suitable for computation offloading scenarios, enabling ground users to access computing resources even in remote or restricted areas where conventional terrestrial infrastructure is unavailable. SAGIN differs from ISTNs in several key aspects. First, the introduction of aerial components between ground users and satellites enables signal enhancement and more stable communication links. Since satellites are located far from the ground, direct communication between users and satellites may suffer from severe signal attenuation and long propagation delays. In SAGIN, aerial components can relay and amplify signals between the ground and space components, thereby improving quality of signals. Second, SAGIN allows computation tasks to be partially or fully processed at the aerial layer.

In conventional ISTNs, when computations locally on user devices are unavailable, even small computation tasks must be transmitted to satellites, which is energy-inefficient with respect to of the transmission power consumption. In contrast, SAGIN enables small or partitioned tasks to be processed by aerial components, achieving lower latency and reduced energy consumption compared to space-based processing. Finally, aerial components in SAGIN offer higher mobility and deployment flexibility than satellites. While satellites have limited accessibility to obstructed environments, such as areas shielded by buildings or rooftops, aerial platforms can dynamically adjust their positions, making SAGIN more adaptable to complex and constrained environments.

Despite its potential, research on SAGIN remains relatively limited. To the best of our knowledge, although several studies have investigated SAGIN, most of them focus on specific use cases or aspects of the system, and there is a lack of recent comprehensive reviews. In particular, existing literature rarely provides a comprehensive analysis that jointly considers SAGIN architectures, computation offloading strategies, and the optimization approaches exploited to support such networks. This insufficiency makes it difficult to systematically understand how promising SAGIN can be efficiently designed and optimized. In this paper, we conduct a detailed analysis of SAGIN architectures by examining selected recent studies. We also reviewed computation offloading strategies adopted in SAGIN and analyze how these strategies are optimized with respect to representative objectives, such as latency and energy consumption. Moreover, this paper places particular emphasis on DRL-based approaches, which are applicable for highly dynamic and complex network environments that can demonstrate real world well. This comprehensive overview of recent works in SAGIN will provide the foundation for the further research.

A. contribution

- We conducted a detailed and organized analysis of the SAGIN architectures proposed in recent studies by decomposing them into three components : ground, aerial and space.
- We compared the computation offloading strategies adopted in each study and, in particular, provided an overview of computation execution placements, data splitting strategies, and processing paths.
- We analyzed the problem formulations and solution approaches of two representative studies that employ traditional optimization techniques and deep reinforcement learning methods, respectively.

II. Architecture of SAGIN

SAGIN integrates space, aerial and ground components to provide global and seamless connectivity. Although ground base station has been considered as main component for communication, space networks and aerial networks are assumed to be essential components for the next-generation network due to their global coverage. Through SAGIN,

we expect to democratize accessible to networks even for people in remote area with no ground base station.

The space component is commonly composed of low Earth orbit (LEO) satellites in most of SAGIN models due to their ability to ensure low-latency. Geostationary Earth orbit (GEO) satellites or medium Earth orbit (MEO) satellites are occasionally used in SAGIN [1] because of long latencies they make, despite their ability to provide wider coverage. For LEO satellites, some papers assume that LEO satellites provide continuous connection while others don't. In the SAGIN model employing a seamless LEO satellites constellation, the space component can provide services without outage [2]. In [3], multiple LEO satellites with a fixed, predetermined and stable orbit constellation are adopted for delivering cloud computing services to the targeted area all the time. Otherwise, the number of satellites is limited and aerial components or users may not be accessible to satellites from time to time [4], [5]. Meanwhile, the satellites can either provide edge computing itself or relay the tasks to cloud server [5] and most of the papers assume that the satellites can communicate with each other using Inter satellite link (ISL) [4], [5]

The aerial component of SAGINs commonly relies on UAVs to provide edge computing and offfload tasks from ground users to satellites or cloud servers by relaying data across heterogeneous network segments. Among the various design considerations for UAV used in SAGINs, energy availability is a main issue, since it directly constrains the flying and operational duration and continuity of providing services in the network. In existing studies, they adopt different assumptions in their work regarding UAV energy models. Some works suggest prolonged-life UAVs equipped with solar panels [6] that provide energy consistently for operation while others assume that UAVs have limited life time. These assumptions significantly affect how UAVs are positioned and operate within the system architecture. To be more specific, most of the papers that adopt UAVs as aerial components assume that UAVs have unlimited energy while papers that exploit other aerial vehicles such as airplanes argue that prolonged life of UAVs is unrealistic. In addition to energy-related considerations, UAV mobility and trajectory design are also recognized as important factors, as they affect communication quality, resource allocation efficiency, and overall system performance in dynamic SAGIN environments.

For instance, UAVs are introduced as a critical control layer in [3], where they are equipped with solar panels to support extended operation durations, along with vision sensors and millimeter-wave (mmWave) communication modules. In this work, UAVs actively collect perception data, such as the velocity and relative distance of mobile users, and utilize this information to make strategic task offloading decisions. Beyond serving as relaying nodes, UAVs can also provide edge computing and caching services, which have been widely exploited to reduce content retrieval latency and alleviate backhaul congestion [6]. These different roles highlight the usage of UAVs in SAGINs, while their abilities to amplify the signals from ground base station are also important.

Meanwhile, several studies explicitly consider UAV mobility and trajectory design as part of the system optimization. In particular, some works jointly optimize UAV trajectories to improve communication quality, energy efficiency, or delay in task completion [5], [7]. However, majority of the literature assume that UAVs follow pre-optimized trajectories, thereby separate trajectory planning from strategy of resource allocation and offloading decisions. While this assumption reduce the complexity of model and optimization, they may overlook the inter-wined relations between UAV mobility, energy consumption, and network performance in highly dynamic SAGIN environments in real world.

However, civil aircrafts (CAs) are used to enhance the Inter Satellite Terrestrial Networks (ISTNs) in terms of coverage, capacity and task scheduling [4] and this network is called CA-augmented space-air-ground integrated networks (CAA-SAGIN) [8]. In the work [9], SAGIN adopts CAs for delay-aware applications in robust task scheduling. This paper argues that UAVs are energy-constrained platforms and can serve tasks only for limited areas, while almost-always-present CAs distributed throughout the world make wide-area data collection and ubiquitous sky access platforms possible. The paper [4] also incorporates CAs, characterized by high velocity and rapidly changing flight directions, to sky access platforms (SAPs) in there work. Due to the dynamic changes of the aircrafts' locations, the aerial component can't work as the control layer. Instead, satellites are utilized for relaying the data from device to ground station (GS) or Cloud server in this paper. Although not common, balloons and drones located in the air are adopted in Google's Project Loon and Facebook's Aquila to provide Internet access even for the restricted areas with many people [10].

For IoT devices, most of the works assume that they have limited power and computation capability. Regarding to offloading strategy, they can offload tasks directly to space components as well as aerial components [2]–[4], while the paper [5] suggests that users can only access to UAVs. Ground users are usually suggested as quasi-static [4], but in the paper [3], they suggest a scenario where a large number of ground devices are moving at different speeds and they navigate by using a random walk model.

III. Comparison of Computation-Offloading Strategies

A. Ground user computing capability

Some IoT devices or users can process tasks locally while others have to offload them entirely for edge computing. For example, tasks can be processed only by being offloaded to SAPs or cloud servers in some SAGIN networks. The work in [3] proposes the computation model which doesn't allow users to process tasks locally. Instead, users offload tasks to either UAVs or satellites for computation. On the other hand, SAPs are considered as vsupplementary components for extensive computation-intensive applications in [4]. Although local computing is possible, SAPs and cloud computing share the burdens by parallel processing. In the paper [5], all ground users are responsible for handling multiple dependent computing tasks. Though they can relay part of tasks to UAV, they can still

handle them locally. To sum up, they have flexibility to perform computations either locally or by offloading.

B. Task offloading structure

1) Data splitting: Two principal task-offloading models are widely adopted in current systems: binary offloading and partial offloading. In binary offloading, a task is treated as an indivisible unit and must be executed entirely at either the local device or the remote server, but never across both simultaneously. This strategy is commonly used for simple or integrated tasks. Conversely, some papers assume that complicated and complex applications are splitted into separate parts for parallel processing in partial offloading. Decision of splitting the data can be made by users or other parts of network. In [3], UAVs whose set works as a control layer decide whether to process some tasks locally or split them and offload some parts to BS or satellites. To guarantee the latency requirements, parallel processing in different platforms which is decided by users is also suggested in [4]. In [5], they suggested partial offloading protocols for their work, where each task is bit-wise independent and can be arbitrarily allocated to other MEC units.

2) Processing Path (Multi-hop or multi-layer processing): In paper [3], tasks made by users can be sent to the UAV layer or directly to satellites. If the tasks are hosted by UAVs, they can be processed locally or resent to either BS or satellites. Similarly, in paper [4], tasks generated by IoT devices can be processed locally or sent to either CAs or satellites. In this work, they assume CAs and satellites as a set, and ground users can choose only one component among them to offload the tasks although CAs and satellites have different characteristics. When satellites are allocated with tasks, they can again forward the data to Ground stations(GSs) for Cloud Computing(CC) while CAs can't due to their rapid change of speed and direction. On the other hand, in the paper [5], when the user decided to offload the tasks, they can send the tasks to only one UAV but not directly to satellites. In other words, UAV is a key layer that relays data between space components and ground users. In this work, UAVs can also offload the data to one satellite at most. For satellites, they can either relay the data to another satellite using ISL or to cloud server when they decided to offload.

3) Computation execution placement: For computation ability of each component, some works assume that local computation of ground devices is unavailable. For example, in the work [3], ground devices can only generate tasks but cannot process them locally. Tasks should be sent to either UAVs or satellites for process. UAVs adopt a frequency scaling technique which allows UAVs to exploit the dynamic adjustment of the CPU cycle frequency. BS also have its own CPU resources to processes. Finally, satellites are assumed that they have multi-core CPU and can process the tasks instantly without any queue backlogs and queueing delays. However, most of the works argue that ground

devices have capabilities to process tasks locally. For instance, users can process the tasks locally in the work [4] as well as edge computing ability of SAPs. Meanwhile, cloud Computing at Ground Stations are assumed to have multi-core high-speed CPUs which has so strong computation capability that the computing latency can be ignored. This assumption is widely adopted since we can ignore the processing latency occurs in cloud servers. Likewise, the papaer [5] which adopt the network with users, UAVs, satellites and cloud servers, also assume that each layer has an ability to compute the task.

C. Offloading Architecture

In this section, we outline the complete offloading architecture adopted in three selected papers, integrating the strategies discussed above on a paper-by-paper basis. Each paper is remarkable for UAVs as a control layer, the adoption of CAs, and a cloud server with no processing delay, respectively.

First paper [3] is notable for their using UAV layer as a control layer. In this work, they divide the computation offloading procedure into two phases which reflect the computational scheduling process. In phase 1, the UAV layer which is designated as a control layer decides whether to host tasks generated from ground devices or not. Here, if the UAV choose not to host tasks, those service demands will be sent to satellites for computation which means computing abilities of ground devices are unavailable. When the UAV is allocated tasks, it makes a decision about where to process the tasks during phase 2. It utilizes a frequency scaling technique and queuing dynamics to make a fairly optimized choice and enhance overall system performance. Tasks that are hosted by UAVs can be processed locally or sent to either BS or satellites. To sum up, UAV-centric control architecture with multi-tier offloading chain and parallel offloading strategy is adopted here.

Second paper [4] can be regarded as a representative example of studies that adopt CAs as aerial components. The network model is composed of quasi-static user devices, SAPs and GSs for CC. Users can compute tasks by themselves but also can utilize SAPs for assistance since they have limited computing capabilities. Once they decide to distribute tasks partially to SAPs, they can offload tasks to either satellites or CAs. While CAs can't relay the data to another due to their highly unstable motion, satellites can relay the data to other satellites using ISL or to GSs for CCs.

Last paper [5] draws a attention since it introduce a clear view of cloud servers and augment them with SAPs. The network in this paper is composed of ground users, UAVs, LEO satellites, and especially cloud servers. The task offloading strategy here is as follows. While all UAVs can serve multiple users at a time, each user can only access to one UAV to upload its tasks for edge computing at a given time slot. Similarly, each satellite can serve many UAVs at a given time slot, each UAV can only access to one LEO satellite at a given time slot. Finally, satellites can also access to one cloud server at a time. Cloud servers are unique components in this paper since when the tasks are

offloaded to cloud servers they are processed immediately, which makes us consider only propagation delay not processing delay.

IV. Problem Formulation Frameworks

A. Problem formulation in [3]

This paper introduced the DRL-and-Perception-aided approach designed to jointly optimize task hosting between ground mobile devices and UAVs, computation offloading, association control between UAVs and BSs, and computing resource allocation in real-time within SAGIN. The optimization can be achieved by minimizing the network's operational cost. Cost model in this paper is defined as combination of both energy consumption at each node to process tasks and energy used in offloading tasks from one node to another node.

1) Cost of computation offloading model: The cost of computation offloading models is the summation of the components below

a) The energy consumed by UAVs, BSs and satellites to process tasks b) The energy required for UAV to offload tasks to nearby BSs and satellites c) Energy consumed to offload tasks directly from ground users to satellites d) Energy consumed by UAV m for collecting tasks via the device-UAV link

2) Joint optimization startegy: Joint optimization with respect to components as follow

a) Computation offloading b) Association control c) Computing resource allocation of UAVs and BSs.

Here, optimization function can be defined as minimization of cost function.

3) Constraints: We have to consider the constraints as below

(a) Limitation of the task hosting for each UAV (b) The maximum computational capacity of a UAV (c) Computational capability of BS regarding available computing resource. (e) The network stability.

B. Problem formulation in [4]

1) Object function: Jointly minimization of the weighted sum of E2E delay and energy consumption. Energy and latency are defined as follows

Energy: a) Energy consumption in local computing, edge computing in SAPs, CC in GSs. b) Transmission energy between users and satellites, between satellites, satellites and ground users, aerial components and users

Latency : Local execution latency, offloading latency, edge computing latency at SAPs, propagation delay, transmission time. Since they adopt parallel computing in the paper, maximum between local execution latency and offloading latency is adopted as total latency.

2) Constraints: We have to consider the constraints as below

a) User association : association variable is binary and user can offload up to one of SAPs b) Transmission power is limited c) Link data rate is limited d) Computation resources

are limited and non-positive. e) Partial offloading strategy f) Maximum tolerance of E2E delay

V. Solution Methodologies and Learning-Based Approaches

A. Solution for problem in [3]

1. Problem Transformation by Lyapunov Optimization

It proceeds to reformulate the problem using Lyapunov optimization for easier resolution. The problem is a naturally stochastic optimization problem. We can break down multi-slot stochastic problem into tractable one-slot problems, tackled sequentially through Lyapunov. Through this work, we can define Lyapunov function as the representation of the task queue backlog and Lyapunov drift function as the representation of the network stability.

2. DRL-and-Perception-Aided Approach

We can solve the joint the optimization problem by deviding them into small subproblems.

1) Reduce the problem into subproblems :

P1 : Joint optimization of computation offloading and local computing resource allocation from the UAVs' view point

P2 : Optimization of the association control between UAVs and BSs.

P3 : Optimization of the allocation of computing resources at ground BSs.

2) Solution for each problem:

P1: We adopt DDPG(Deep Deterministic Policy Gradient) to solve this problem. The action space in P1 is continuous and high-dimensional, since the offloading tasks and CPU frequencies are real-valued variables rather than discrete values. This makes conventional value-based methods not be adopted such as Q-learning or DQN since they are designed for discrete action space. Moreover, the considered environment is highly dynamic: queue states, mmWave-based perception results, user mobility, and time-varying wireless channels make it difficult to derive an accurate analytical model for optimal control. For the dynamic environments where analytical models are not aplicable, a model-free reinforcement learning approach is more appropriate. DDPG is an off-policy actor–critic algorithm so it can learn a deterministic policy for continuous control while efficiently reusing past experiences. This enables fine-grained optimization of task offloading decisions and computing resource allocation, including CPU frequency scaling.

P2: UAVs and ground BSs association control is optimized through solving P2. This problem is a mixed-integer nonlinear problem (MINP) and as widely known, can be solved using a Deep Q-Network (DQN). The DQN algorithm is particularly notable for handling high-dimensional state spaces, which makes it suitable for complex decision-making tasks characterized by substantial state representations.

P3: The objective of this problem is to optimize BS computing resource allocation. This problem adopts the Self-adaptive Global-best Harmony Search (SGHS) algorithm. SGHS,

heuristic method, can handle non-convex, multi-variable problems without requiring gradient information or convexity assumptions. It performs a global search to avoid being stuck in poor local minima, flexibly handles continuous CPU-allocation variables and constraints. So we can achieve near-optimal cost with reasonable computational complexity.

B. Solution for problem in [4]

1) Set the upper bound of the locally processed ratio rho which means how much of the tasks will be served locally. 2) We can linearize the total delay term for simplicity while we cannot linearize energy terms. The transformed formula is still a mixed-integer nonlinear programming (MINLP) probelm which is an NP-hard problem due to the coupling relationship among different variables. 3) Devide the problem into two subproblems, primal problem and master problem to use GBD. In GBD, primal problem and master problem are solved iteratively. Primal problem is solved first and the result is added to the master problem. Then the master problem is solved. Upper bound is obtained by the primal problem and the lower bound obtained by the master problem. 4) Primary problem is reduced to convex problem and the optimal solution can be obtained by applying Karush-Kuhn-Tucker (KKT conditions). 5) Master problem is NP-hard problem due to multiple coupling parameters. The parallel successive convex approximation (SCA) algorithm is used in here to transform the multi-variable NP-hard problem into a convex one. 6) Based on SCA theory, we can extend the existing SCA approximation methods and transform the master problem into a tractable convex problem. 7) Finally, we can solve the problem using Multi-tier partial task offloading (MPTO) algorithm suggested in the paper.

VI. Conclusion

Through an analysis of ten selected SAGIN papers, we examined their network architectures, computation offloading strategies, as well as the corresponding problem formulations and solution approaches. This paper is meaningful in that it provides a detailed and systematic analysis of architectures adopted in recent SAGIN research and investigates computation offloading strategies from various perspectives. In addition, we reviewed DRL-based approaches, which have recently considered as an important optimization technique in SAGIN studies. Given the inherent complexity and multi-component nature of SAGIN, the design of accurate system models and effective solutions for dynamic optimization problems is expected to be increasingly important. We believe that this work can serve as a foundation for future research on SAGIN and its optimization methodologies.

References

[1] J. Liu, Y. Shi, Z. M. Fadlullah, and N. Kato, "Space-air-ground integrated network: A survey," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 4, pp. 2714–2741, 2018.

[2] N. Cheng, F. Lyu, W. Quan, C. Zhou, H. He, W. Shi, and X. Shen, "Space/aerial-assisted computing offloading for iot applications: A learning-based approach," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 5, pp. 1117–1129, 2019.

[3] Y. Gao, Z. Ye, and H. Yu, "Cost-efficient computation offloading in sagin: A deep reinforcement learning and perception-aided approach," *IEEE J. Sel. Areas Commun.*, vol. 42, no. 12, pp. 3462–3476, 2024.

[4] Q. Chen, W. Meng, T. Q. S. Quek, and S. Chen, "Multi-tier hybrid offloading for computation-aware iot applications in civil aircraft-augmented sagin," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 2, pp. 399–417, 2023.

[5] C. Huang, G. Chen, P. Xiao, Y. Xiao, Z. Han, and J. A. Chambers, "Joint offloading and resource allocation for hybrid cloud and edge computing in sagins: A decision assisted hybrid action space deep reinforcement learning approach," *IEEE J. Sel. Areas Commun.*, vol. 42, no. 5, pp. 1029–1043, 2024.

[6] N. Zhang, S. Zhang, P. Yang, O. Alhussein, W. Zhuang, and X. S. Shen, "Software defined space-air-ground integrated vehicular networks: Challenges and solutions," *IEEE Commun. Mag.*, vol. 55, no. 7, pp. 101–109, 2017.

[7] Y. Dai, Y. Li, and T. Lyu, "Task offloading based on multi-agent reinforcement learning for uav-assisted edge computing," in *2024 Asia-Pacific Conference on Image Processing, Electronics and Computers (IPEC)*, 2024, pp. 426–430.

[8] Q. Chen, W. Meng, S. Li, C. Li, and H.-H. Chen, "Civil aircrafts augmented space–air–ground-integrated vehicular networks: Motivation, breakthrough, and challenges," *IEEE Internet of Things Journal*, vol. 9, no. 8, pp. 5670–5683, 2022.

[9] Q. Chen, W. Meng, S. Han, C. Li, and H.-H. Chen, "Robust task scheduling for delay-aware iot applications in civil aircraft-augmented sagin," *IEEE Trans. Commun.*, vol. 70, no. 8, pp. 5368–5385, 2022.

[10] N. Kato, Z. M. Fadlullah, F. Tang, B. Mao, S. Tani, A. Okamura, and J. Liu, "Optimizing space-air-ground integrated networks by artificial intelligence," *IEEE Trans. Wireless Commun.*, vol. 26, no. 4, pp. 140–147, 2019.