frontiers | Frontiers in Communications and Networks

TYPE Original Research
PUBLISHED 22 November 2024
DOI 10.3389/frcmn.2024.1467812

OPEN ACCESS

EDITED BY
Tawfik Al-Hadhrami,
Nottingham Trent University, United Kingdom

REVIEWED BY
Kwame Assa-Agyei,
Nottingham Trent University, United Kingdom
Umar Yahya,
Islamic University in Uganda, Uganda
Shangshi Huang,
Tsinghua University, China

*CORRESPONDENCE
Mokhtar Harrabi,
harrabim@yahoo.fr

RECEIVED 20 July 2024
ACCEPTED 05 November 2024
PUBLISHED 22 November 2024

CITATION
Harrabi M, Hamdi A and Bel Hadj Tahar J (2024) Optimizing service caching in smart buildings: a dynamic approach for responsive IoT and edge computing integration in smart cities. *Front. Comms. Net* 5:1467812. doi: 10.3389/frcmn.2024.1467812

Optimizing service caching in smart buildings: a dynamic approach for responsive IoT and edge computing integration in smart cities

Mokhtar Harrabi[1,2]*, Abdelaziz Hamdi[2] and Jamel Bel Hadj Tahar[2]

[1]Department of Computer Engeering, Higher Institute of Computer Science and Communication Technologies (ISITCOM) University of Sousse, Sousse, Tunisia, [2]Networked Objects, Control and Communication Systems (NOCCS) Research Lab, ENISO University of Sousse, Sousse, Tunisia

Introduction: This paper introduces a novel approach for optimizing service caching in smart buildings through the integration of Internet of Things (IoT) and edge computing technologies. Traditional cloud-based solutions suffer from high latency and resource consumption, which limits the performance of smart city applications.

Methods: The proposed solution involves a dynamic crowdsourcing and caching algorithm that leverages IoT gateways and edge servers. This algorithm reduces latency and enhances responsiveness by prioritizing services for caching based on a newly developed efficiency metric. The metric takes into account cloud and edge-computed response times, memory usage, and service popularity.

Results: Experimental results show a reduction in average response time (ART) by up to 25% and a 15% improvement in resource utilization compared to traditional cloud-based methods.

Discussion: These findings underscore the potential of the proposed approach for resource-constrained environments and its suitability for smart city infrastructures. The results provide a foundation for further advancements in edge-based service optimization in smart cities.

KEYWORDS

smart buildings, IoT, edge computing, service caching, optimization, gateway, cloud computing

1 Introduction

In the dynamic landscape of smart cities, the orchestration of resources within smart buildings has emerged as a critical area of research and innovation (Harnal et al., 2022). The deployment of edge servers within this urban framework, as illustrated in Figure 1, offers a strategic approach to minimizing latency and enhancing user experience by reducing dependence on distant cloud servers (Suhag and Jha, 2023a; Hasan et al., 2024). However, the finite nature of edge server resources requires a meticulous optimization strategy within each smart building to ensure efficient resource utilization (Nauman et al., 2023; Aljubayrin et al., 2023). This study explores the intricate relationship between IoT gateways, edge servers, and cloud resources in the context of smart buildings within smart cities. As

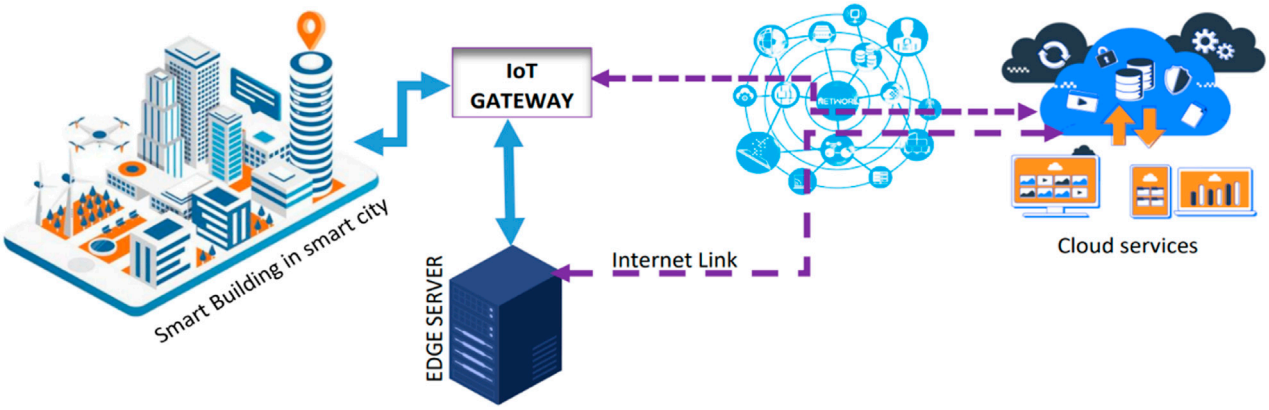

FIGURE 1
Integrated data flow architecture for smart buildings: Device to cloud connectivity.

buildings are allocated fixed resources on edge servers, a major challenge arises in developing intelligent algorithms that optimize resource usage and ensure the efficient functioning of home applications. Optimization becomes particularly crucial as services, initially deployed in the cloud, are dynamically crowdsourced and cached at the edge to improve the average response time (ART) of user applications (Nguyen et al., 2023; Suhag and Jha, 2023b; Zhou et al., 2019). The pivotal role of the IoT gateway, acting as a resource orchestrator, is highlighted as it balances the use of allocated resources while navigating the constraints of edge servers (Pallewatta et al., 2023). This paper introduces a smart algorithm implemented within IoT gateways, designed to dynamically crowdsource and cache services based on real-time demands, service popularity, and response times. The algorithm aims to create a responsive and efficient smart building ecosystem. The strategic deployment of edge servers, along with this intelligent algorithm, aims to reduce cloud reliance and enhance service delivery performance. The algorithm's novelty lies in its ability to prioritize services for caching, weighing response time improvements against memory usage.

In light of the increasing demand for low-latency, high-bandwidth services driven by the proliferation of internet-connected devices, this study proposes a new approach to optimize service caching in edge servers. The proposed solution addresses the limitations of conventional cloud-based approaches, which often result in high latency and resource consumption. By introducing a smart metric that evaluates the trade-offs between response time gains and memory costs, the optimization problem is tackled in the specific context of smart buildings and smart cities. Simulations based on real-world scenarios demonstrate the superiority of the proposed approach, showcasing improvements in average response time and cache hit ratio compared to existing caching strategies.

This research makes significant contributions to the field of IoT and edge computing, offering a practical solution for optimizing service caching in smart buildings. The results not only validate the effectiveness of the approach in resource-constrained environments but also add valuable insights into the growing body of research in edge computing. The study highlights the potential for further advancements in the optimization of edge resources, aligning with the demands for efficient service delivery in smart city infrastructures.

The main contributions of this paper are as follows:

1. A new dynamic crowdsourcing and caching algorithm that optimizes service delivery in smart buildings is introduced.
2. A novel efficiency metric for caching services, which considers both cloud-computed and edge-computed response times, is proposed.
3. The effectiveness of the approach is demonstrated through extensive experimental evaluations, showing significant improvements in average response time (ART) and resource usage compared to traditional methods.

The remainder of the paper is organized as follows: Section 2 reviews related work on service caching and IoT integration in smart buildings. Section 3 describes the proposed approach and the new efficiency metric. Section 4 presents the experimental setup and results. Section 5 discusses the implications of the findings and compares the proposed approach with existing methods. Finally, Section 6 concludes the paper and suggests directions for future research.

2 Related work

The evolution of Mobile Edge Computing (MEC) has ushered in a paradigm shift in the realm of smart buildings, where the orchestration of caching strategies and optimization techniques is pivotal for seamless and responsive operations. In this context, numerous studies have explored various methodologies to enhance the performance of edge servers, particularly addressing the unique challenges posed by the dynamic nature of smart building environments.

Among the plethora of caching strategies, the Least Recently Used (LRU) algorithm has garnered widespread recognition. Its efficacy in reducing cache miss rates has made it a staple choice in MEC environments (Mulero-Palencia and Monzon Baeza, 2023; Oliveira et al., 2023). In smart buildings, where edge servers serve as

local caches, the LRU algorithm strategically manages data by evicting the least recently accessed content, thereby maintaining a compact cache size. This approach not only aids in reducing the likelihood of overloading edge servers but also aligns with the resource constraints often associated with smart building deployments (Alzakari et al., 2020). Similarly, the Least Frequently Used (LFU) algorithm has found application in scenarios where services exhibit varying usage frequencies. Particularly effective for services with low usage frequency, LFU minimizes cache miss rates by removing the least frequently accessed data (Haraty and Nancy, 2023). This nuanced approach caters to the diverse usage patterns within smart buildings, ensuring optimal resource utilization.

Beyond traditional caching methods, the integration of meta-heuristic techniques has been explored to optimize resource allocation and enhance the performance of IoT applications within smart buildings. Genetic algorithms have been employed to dynamically allocate resources between IoT devices and edge servers (Abbas et al., 2021; Natesha and Guddeti, 2022; Deng et al., 2018), showcasing adaptability in resource-intensive environments. Particle swarm optimization has emerged as a valuable tool for load distribution across edge servers (Wu et al., 2019; Zhang et al., 2023; Busetti et al., 2022), ensuring balanced workloads and efficient utilization of computational resources within the smart building ecosystem. The realm of swarm intelligence techniques has also made a significant impact on dynamic optimization in edge computing environments (Hua et al., 2023; Sudha et al., 2023). Techniques such as particle swarm optimization, ant colony optimization, and artificial bee colony have been scrutinized for their applicability in smart buildings (Zulfa et al., 2022). These algorithms, designed to adapt to dynamic environments, offer promising avenues for addressing constraints inherent in the smart building ecosystem (Bibri et al., 2024; Bouramdane, 2023; Walia et al., 2023). Recent advancements in edge computing also emphasize proactive content caching strategies. Aghazadeh et al. (2023) conducted a systematic literature review on proactive content caching in edge computing environments, highlighting the crucial role these techniques play in improving traffic management and response times. The study categorizes proactive caching mechanisms into model-based, machine-learning-based, and heuristic-based approaches, addressing open issues and challenges that need to be explored in future research. Furthermore, Gupta et al. (2023) introduced an Information Centric Networking (ICN)-based edge caching scheme for managing multimedia big data traffic in smart cities. Their proposed architecture combines edge computing and ICN to enhance content dissemination near the end user. By incorporating caching attributes and employing proactive caching mechanisms, their approach significantly improves performance metrics such as cache hit ratio and content retrieval delay. Gou and Wu (2024) focused on optimizing the collaboration strategy among edge servers in IoT applications within smart cities. They proposed an edge server group collaboration architecture (ESGCA) to mitigate the challenges posed by insufficient cache space and urgent transmission tasks. Their optimization method effectively reduces message transmission delays and energy consumption, demonstrating the need for collaborative strategies in resource-constrained environments. Additionally, Songhorabadi et al. (2023) examined fog computing approaches in IoT-enabled smart cities, emphasizing their role in latency-sensitive and security-critical applications. Their study classified fog computing methods into service-based, resource-based, and application-based categories, addressing the limitations of traditional cloud-based approaches in smart city scenarios. Resource management techniques further contribute to the optimization of edge computing in smart buildings. The survey conducted by Khanh et al. (2023) delves into resource allocation, load balancing, and task scheduling strategies. Employing heuristic algorithms, game theory, and machine learning, these approaches ensure that smart buildings efficiently distribute tasks among edge servers, overcoming challenges posed by fluctuating workloads. The insights provided in these works highlight the multifaceted nature of caching and optimization within the evolving landscape of smart buildings, illustrating the necessity for innovative approaches to meet the growing demands for efficient service delivery in smart city infrastructures.

3 Problem definition

The rapid advancement of Internet of Things (IoT) technologies has transformed smart buildings into complex ecosystems that require efficient resource management and service delivery. As the demand for low-latency services continues to rise, traditional cloud-based solutions have proven inadequate due to their reliance on centralized data processing, resulting in high latency and increased resource consumption. These challenges significantly hamper the performance of smart applications.

This study proposes a novel approach to optimizing service caching in smart buildings through the integration of edge computing technologies and a dynamic crowdsourcing technique. The innovative smart metric developed in this research takes into account multiple factors, balancing the gain in response time when a service is cached against the loss of memory consumption. This dual-focus approach not only addresses the limitations of existing caching strategies, such as Least Recently Used (LRU) and Least Frequently Used (LFU), but also enhances resource utilization within constrained environments. The uniqueness of this approach lies in its ability to reduce the complexity of the caching algorithm, making it feasible for real-time implementation in dynamic ecosystems. Unlike traditional enumeration methods, which are computationally expensive and time-consuming, the proposed algorithm can efficiently operate in real-time scenarios. Additionally, while many existing solutions utilize AI approaches like Genetic Algorithms (GA) and Swarm Intelligence, they often struggle with complexity and scalability in resource-constrained settings. The smart metric facilitates a simpler yet effective optimization process that is both practical and efficient.

By optimizing resource allocation and reducing average response times (ART), the proposed method directly contributes to the overarching goal of creating intelligent environments capable of handling increasing data traffic and diverse user requirements. The necessity for such advancements is underscored by the rapid proliferation of IoT devices, projected to reach 29.3 billion connected devices by 2023, resulting in immense data demands that require innovative solutions for effective management (Gupta et al., 2023; Gou and Wu, 2024).

TABLE 1 Symbols list.

Symbols	Description
cs	Cloud server
es	Edge server
Vu---->cs	Data transmission rate between device and cloud
Vcs---->es	Data transmission rate between edge server and cloud
Vu---->es	Data transmission rate between device and edge server
s_i	Service that fulfill specific task
S	Service set
Gsc	Service composition graph
Sc	Set of composite service
Sa	Set of atomic service
r_i	Resource consumption of service si
R	Resource limitation of edge server
f_i	Popularity (frequency)of service si
t_i^e	Service response time when s_i placed on e_s
T_i^c	Service response time when s_i placed on c_s
MS_i	Set of member service s_i
y_i	Selection indicator for s_i

Moreover, this research addresses the challenges posed by high traffic volumes and resource constraints in smart cities. It builds on the findings of Gupta et al. (2023), who highlight the importance of proactive content caching strategies, and Gou and Wu (2024), who propose edge caching schemes for multimedia big data traffic. By focusing on dynamic adaptability and collaboration among edge servers, this study presents a comprehensive solution that ensures efficient service delivery while maintaining a focus on user experience.

3.1 Symbols used for describing the optimization problem and caching strategy

These symbols play a pivotal role throughout the entire paper, serving as the cornerstone for articulating both the optimization problem and our proposed caching strategy. The utilization of boldface symbols is intentional, emphasizing their significance and aiding in clarity. Within the provided Table 1, you will discover a comprehensive list of symbols used to represent various elements and concepts in the context of our discussio. For a more detailed understanding, here's a breakdown of the symbols and their respective meanings.

The amount of resources utilized by a service is specified by: $R = \left(R_{max}^{res1}, R_{max}^{res2}, \ldots, R_{max}^{resn}\right)$

The average service response time (ASRT) is a crucial metric for evaluating the efficiency of a service delivery system. For each service Si, there are typically two different service response times t_i^e and T_i^c, Therefore, the latency can be determined by Equation 1:

$$\begin{cases} t_i^e = \frac{in_i}{Vu \rightarrow es} + C_{exec_{time}}^{edge} + \frac{out_i}{vu \rightarrow es} \\ T_i^c = \frac{in_i}{Vu \rightarrow Cs} + C_{exec_{time}}^{cloud} + \frac{out_i}{vu \rightarrow Cs} \end{cases} \quad (1)$$

were.

$\left(C_{\text{exec}_{\text{time}}}^{\text{edge}}\right)$:computing time of service Si under Edge server.

$\left(C_{\text{exec}_{\text{time}}}^{\text{cloud}}\right)$:computing time of service Si under Cloud server.

$\frac{\text{in}_i}{\text{Vu} \rightarrow \text{es}}$: time to transfer data from user application to Edge.

$\frac{\text{out}_i}{\text{vu} \rightarrow \text{es}}$: time to transfer response from Edge server to the user application.

$\frac{\text{in}_i}{\text{Vu} \rightarrow \text{Cs}}$: time to transfer data from the user application to Cloud server.

$\frac{\text{out}_i}{\text{vu} \rightarrow \text{Cs}}$: time to transfer response from Cloud server to the user application.

$\frac{\text{out}_i}{\text{vu} \rightarrow \text{Cs}}$: time to transfer response from Cloud server to the user application.

When a service Si is a composite service made up of other services, $MS_i = \{S_{i1}, S_{i2}, \ldots, S_{ik}\}$, its response time $t_i^*(y)$ can be derived from the response times of its constituent services, taking into account the values of the vector <y>, where yi = 1 indicates that Si is in cache and $y_i = 0$ signifies that Si is not in cache as shown in Equation 2:

$$t_i^*(y) = \begin{cases} Min\left(t_i^e, \sum_{sj \epsilon MSi} t_j^*\right), yi = 1, si \epsilon Sc \\ Min\left(T_i^c, \sum_{sj \epsilon MSi} t_j^*\right), yi = 1, si \epsilon Sc \\ t_i^e, yi = 1, si \notin Sc \\ T_i^e, yi = 0, si \notin Sc \end{cases} \quad (2)$$

Therefore, the average response time is expressed as follows as shown in Equations 3, 4.

$$\tilde{rt}(y) = f1 \times t_1^*(y) + f2 \times t_2^*(y) + \ldots + fn \times t_n^*(y) \quad (3)$$

$$\tilde{rt}(y) = \sum_{i=1}^{n} fi \times t_i^*(y) \quad (4)$$

In conclusion, the problem can be stated as follows: Given a composition graph (Gsc), the objective is to find the cache policy vector (Y) that results in the minimum average service response time (Min-ASRT) as shown in Equations 5, 6, respectively.

$$\min \tilde{rt}(y) = \sum_{i=1}^{n} fi \times t_i^*(y) \quad (5)$$

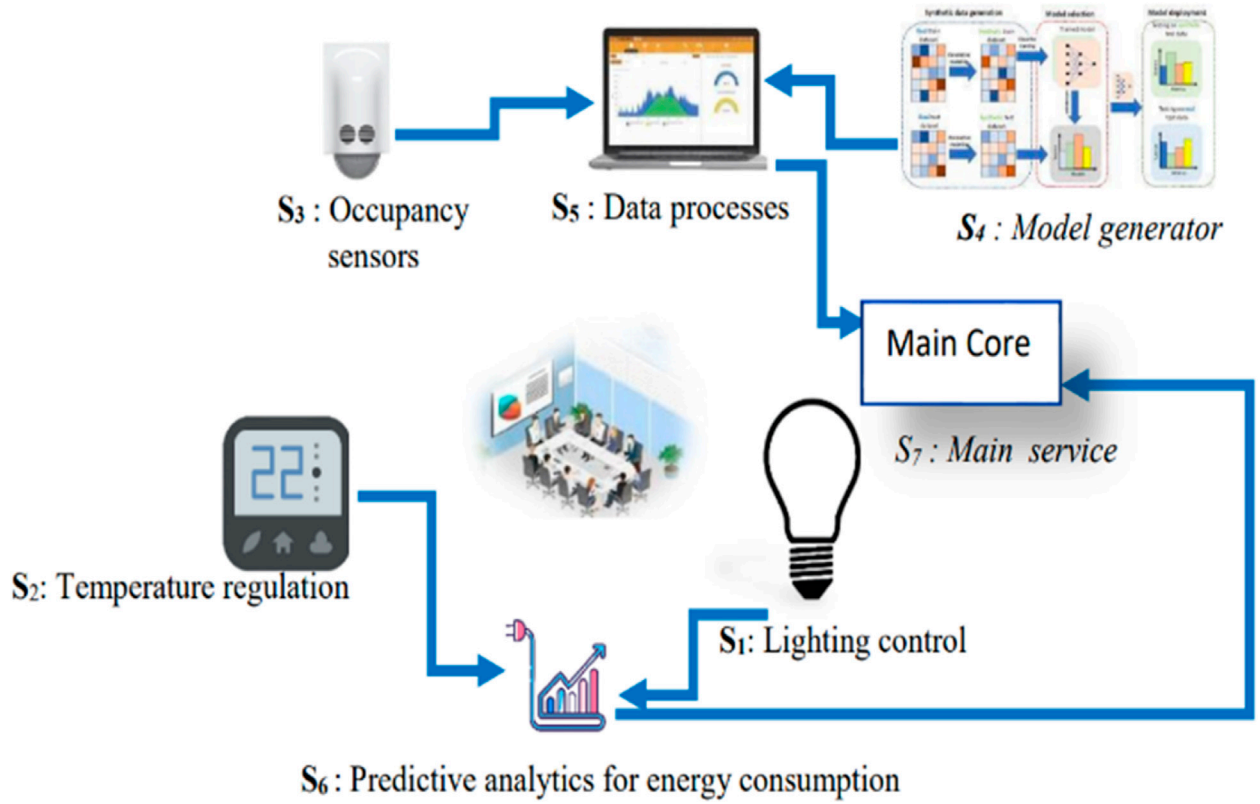

FIGURE 2
Intelligent automation in smart buildings: A practical scenario illustrating the application of advanced caching strategies.

TABLE 2 Running config.

Service	Ti [ms]	Ti [ms]	Memory [Mb]	Frequency [%]
S_1	2.5	4.0	45	7
S_2	2.0	3.0	80	8
S_3	3.5	4.5	50	11
S_4	4.0	5.5	40	14
S_5	7.0	8.5	70	9
S_6	4.0	6.0	100	16
S_7	10.5	1.0	190	35

$$\text{Subject to s.t} \begin{cases} \sum_{i=1}^{m} \text{yiri} \leq R \\ y_i \in \{0, 1\} \end{cases} \quad (6)$$

3.2 Optimization techniques and algorithm

In the realm of optimization methodologies, the Enumeration Method serves as a rigorous yet computationally demanding approach. This method exhaustively traverses the entire solution space in pursuit of the optimal response time, involving the generation of all conceivable cache policy vector combinations (y) and subsequent calculation of the average service response time (ARST). The Algorithm 1 illustrate all the step of enumeration method. However, its accuracy comes at the cost of significant computational resources, rendering it less viable for large-scale systems due to the NP-complete nature of the problem. In contrast, heuristic algorithms offer a pragmatic alternative for timely solutions. Algorithm 1, inspired by the Enumeration Method, adopts a numeric approach to explore the solution space iteratively, efficiently selecting cache policies that minimize the average service response time. These heuristic algorithms include the Greedy Algorithm, which strategically selects and adjusts services based on popularity and response time, and the Genetic Algorithm, inspired by natural selection,

which evolves candidate solutions through crossover and mutation operations. Other heuristic algorithms like Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), and Simulated Annealing (SA) contribute to the repertoire of optimization strategies. The choice among these methods hinges on factors like problem size, optimization objectives, and available computational resources, with performance evaluations considering solution quality and efficiency relative to optimal solutions obtained through techniques such as Integer Linear Programming (ILP) or Mixed-Integer Programming (MIP).

```
Input:
  SCG (S, E): the service composition graph;
  T = {Ti} for i = 1 to n: the response time on the cloud;
  t = {ti} for i = 1 to n: the response time on the edge
  if cached;
  y: the cache policy;
Output:
  {t*1, t*2, ..., t*n}: response time with cache policy y;
    1:  Stack S† ← ∅
    2:  while not all computed do
    3:    Szero ← {si | si ∈ S, indegree (si) = 0}
    4:    for sk ∈ Szero do
    5:      if outdegree (sk) = 0 then
    6:        if yk = 1 then
    7:          t*k = te*k
    8:        else
    9:          t*k = Tc*k
   10:      else
   11:          push (S†, sk)
   12:      for sk ∈ Szero do
   13:        remove (G, sk)
   14:        update G
   15:      while S† ≠ ∅ do
   16:        Sk ← pop (S†)
   17:        for sk ∈ Sk do
   18:          if yk = 1 then
   19:            t*k = min{ Σsj∈MSk t*j, te*k }
   20:          else
   21:            t*k = min{ Σsj∈MSk t*j, Tc*k }
   22: return {t*1, t*2, ..., t*n}n}
```

Algorithm 1. CSRT(Computing Service Response Time).

3.3 Proposed algorithm

In the realm of Mobile Edge Computing (MEC), the optimization of service response time is a critical consideration, given the inherent constraints of edge servers. Our proposed caching strategy introduces a smart metric, E_i, designed to enhance the average service response time while acknowledging resource limitations. E_i is pivotal, as it gauges the potential improvement in response time against the associated memory cost of caching a service in the edge. The implementation of our approach involves a structured series of steps. Initially, we create a service composition graph (Gsc) to elucidate the intricate relationships between atomic and composite services. The focus lies on caching atomic services, considered as fundamental building blocks. Subsequently, we categorize services into two sets: atomic services (Sa) and composite services (Sc). The Ei metric is then evaluated for each atomic service, leading to a prioritized list for caching based on their calculated importance. The highest priority services are cached until resource limitations are reached. Importantly, our method maintains a moderate level of complexity, relying on tasks such as graph construction, metric evaluation, and prioritization, without necessitating elaborate optimization techniques or extensive computational resources. It is noteworthy that the approach relies on trial-and-error processes and a set of rules rather than a deterministic algorithm. The Ei metric is defined by Equation 7:

$$E_i = \left[(T_i - t_i)/r_i \right] * f_i \tag{7}$$

where Ti is the cloud-computed service response time, ti is the edge-computed response time, ri is the memory required for edge storage, and fi denotes service popularity. The metric provides an efficiency measure for caching services in the edge, with higher Ei values indicating a favorable trade-off between response time improvement and memory usage. Services with the highest Ei values are given priority for caching, leading to optimal resource allocation. This method, encapsulated in Algorithm 2, facilitates informed decisions on service caching in the edge, ensuring effective optimization of limited edge server resources. Through these steps, our approach contributes to the advancement of efficient strategies for Mobile Edge Computing environments.

```
Input:
  SCG (S, E): the service composition graph;
  {Ti} for i = 1 to n: the response time on the cloud;
  {ti} for i = 1 to n: the response time on the edge
  if cached;
  {fi} for i = 1 to n: the popularity of service i
  {ri} for i = 1 to n: the resource consumptions
  of services
Output:
  y = {y1, y2, ..., yn}: the cache policy of services;
begin:
  1:  for each atomic service
  2:    Ei = [(Ti-ti)/ri]*fi
  3:      Rank the atomic services in descending order of
          priority for caching based on their Ei
          metric value.
  4.      Cache the atomic services with the highest
          priority in the edge server until the
          resource limitations of the edge server
          are reached
  5:  return y = {y1, y2, ..., yn}://yi = 1 if service i in
      cache 0 if not
end
```

Algorithm 2. proposed Algorithm.

To further illustrate our approach, we present a practical scenario that showcases the application of our proposed caching strategy in a smart building context. In this study, we envision a smart office building equipped with an intelligent automation system, as depicted in Figure 2. The system is designed to

TABLE 3 Services score Using Equation 7.

Service	Gt, [ms]	Score, Gi, [ms/Mb]
S1	1.5	0.230
S2	1.0	0.100
S3	1.0	0.220
S4	1.5	0.525
S5	1.5	0.192
S6	2	0.320
S7	0.5	0.092

TABLE 4 Ranking based on score.

Rank	1	2	3	4	5	6	7
Services	S3	S4	S2	S1	S7	S6	S5
Memory, [Mb]	50	40	80	45	190	100	70
Edge capacity, [Mb]	50	90	170	170	170	170	170

automate various aspects of building management, including lighting control, temperature regulation, occupancy sensing, and more. The building is equipped with sensors and actuators, and data from these devices are processed and managed through a set of cloud services (s_1, s_2, . . . , s_7).

For instance, the first service (S_1: lighting_control) manages the lighting in specific areas, while the second service (S_2: temperature_regulation) handles the temperature control. The S6 service is a composite of S_1 and S_2, providing predictive analytics for energy consumption optimization. The S_3 service stores data from occupancy sensors, while the S_5 service processes this data using anomaly detection models generated by S_4. Finally, the S_7 service analyzes the overall building state and makes decisions to optimize energy usage and comfort, which are then communicated to the local building management system.

Table 2 presents the operational parameters for our services configuration. Where

T_i: service response time when Si placed on the cloud server.

Frequency: popularity of service S_i.

Memory: resource consumption of service S_i

In this scenario, our objective is to find the optimal cache policy that minimizes the average service response time (ASRT) while adhering to the edge server's capacity limit of 200 Mb. Initially, it may seem logical to start caching the most frequently requested service. However, in this case, the most popular service could consume nearly all of the available edge resources, for instance, in this configuration S7 is the most popular service (35%) and consumes 190Mb, thus resulting in a higher ASRT: $\sum_{i=1}^{6} \mathrm{Ti} \times \mathrm{fi} + \mathrm{t7} \times \mathrm{f7} = 718.5$ m. This would result in a higher ASRT compared to the average response time (704.0 m) achieved by choosing less popular services (S1, S2, S5). Calculating the ASRT for this configuration of services is a complex operation that necessitates a specialized algorithm, taking into consideration all its properties.

The comprehensive implementation of our method on the dataset provided in Table 2 has led to the generation of results that are meticulously detailed and thoughtfully presented in the subsequent Table 3. This iterative process encapsulates the transformative impact of our methodology on the given data, offering a deeper insight into the outcomes derived from the applied approach.

The histogram in Figure 3 displays the gain in time per bit (Score) for each service. Based on these results, the services are ranked and stored in the edge cache until its memory capacity is fully utilized, as indicated in Table 4.

Table 4 allows us to easily deduce the cache retention policy, y = {0, 0, 0, 1, 1, 1, 0}, which results in an ASRT calculation of 643.5 M using Equation 4. These results are consistent with those obtained through the enumeration method. It's crucial to note that this is a specific scenario, and the proposed method needs evaluation in more generic and automatic scenarios. The next section aims to compare the effectiveness of our method with literature-based methods in a broader context, providing a comprehensive

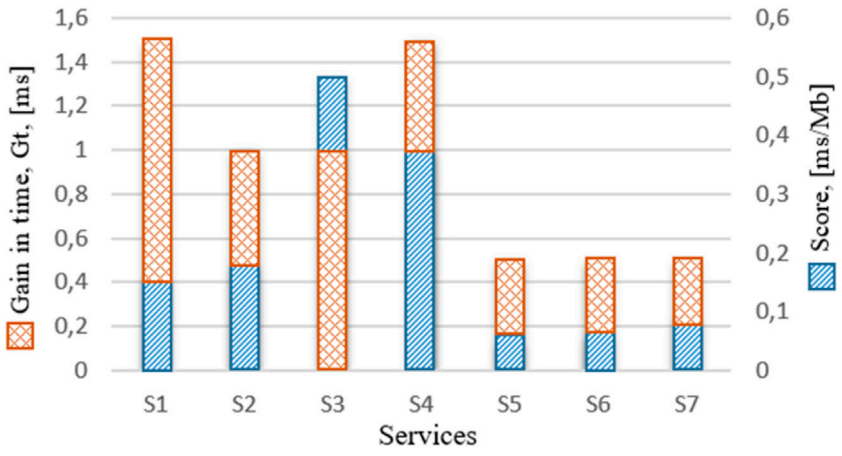

FIGURE 3
Simulation results of proposed method, as in First Step.

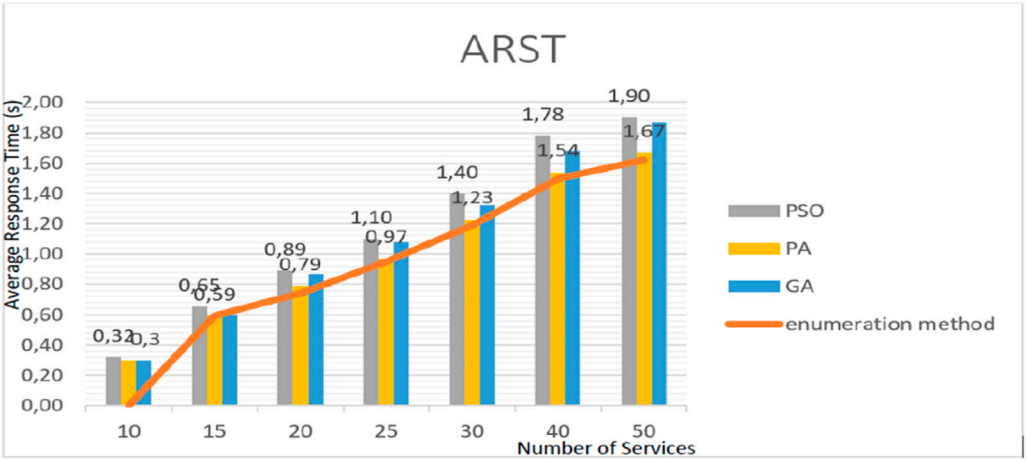

FIGURE 4
The performance evaluation results of various caching algorithms.

assessment of its applicability and performance. The results of this comparative analysis will be presented in the subsequent section.

4 Experimental results and discussion

In this section, we present the experimental results and discussion of our proposed caching strategy and compare it with the literature algorithm. We first describe the experimental setup and the datasets used in our experiments. Then, we present the performance evaluation results of our proposed method and the literature algorithm in terms of response time, hit ratio, and cache utilization. Finally, we discuss the results and draw conclusions regarding the effectiveness and efficiency of our proposed method compared to the literature algorithm.

4.1 Experimental setup and limitations

The first round of tests was conducted on a computer equipped with a 13th Gen Intel(R) Core(TM) i5-1335U CPU running at 1.30 GHz, with 24 GB of memory. This environment allowed for the simulation of various caching scenarios, where key metrics such as Average Response Time (ART), Resource Utilization, and Computational Complexity were measured. These tests provided valuable insights into how the proposed caching algorithm performs in an environment with ample resources.

After finalizing the simulation environment, the algorithm was deployed on a Raspberry Pi as the final target platform to simulate its performance in a real-world, resource-constrained environment like a home gateway. This testing ensured that the algorithm could handle service caching effectively in a constrained setting.

However, a limitation of our solution is related to the resource constraints of home gateways, particularly when the set of services to be managed becomes very large. If the number of services exceeds a certain threshold (e.g., more than 50 services), the system may face difficulties in maintaining optimal performance due to limited memory and computational power. Nonetheless, in typical real-world scenarios, the number of services on a home gateway is likely to be fewer than 50, ensuring that the system will function correctly within these bounds.

4.2 Comparison of service response time between proposed method, GA, and swarm

Average Response Time (ART) is a metric that is widely used to evaluate the performance of caching algorithms in Mobile Edge Computing (MEC) environments. This metric measures the average time taken to respond to service requests, and lower values indicate better performance. In MEC environments, where edge servers are closer to end-users, reducing ART is of great importance as it enhances the user experience and satisfaction. Moreover, the performance of caching algorithms can be evaluated using other metrics, such as cache hit ratio, resource utilization, and energy consumption, but ART is often considered the most critical metric because it directly reflects the quality of service provided to end-users. Therefore, reducing ART has become one of the primary objectives of caching algorithms and methods in MEC environments.

To ensure a fair comparison of the different caching algorithms, all algorithms, including the proposed approach, were implemented using Python and executed under the same conditions on the same machine. This consistent execution environment minimizes variability in performance results, allowing for a direct and equitable evaluation of each algorithm's effectiveness.

The performance evaluation results of various caching algorithms in a Mobile Edge Computing (MEC) environment are shown in Figure 4 using the average response time (ART) metric. The histogram displays the ART values for each algorithm at different levels of service requests, while the proposed caching approach is denoted as "PA" on the histogram. It can be observed that "PA" consistently outperforms or performs comparably to the other caching algorithms across all levels of service requests, with lower or very close ART values. The Particle Swarm Optimization (PSO) algorithm generally performs worse than the other algorithms, while the Genetic Algorithm (GA) performs relatively well compared to PSO

but has higher ART values than "PA" and the reference "enumeration method." These results emphasize the significance of reducing ART in MEC environments, as it directly impacts the quality of service provided to end-users. Therefore, the proposed caching approach, "PA", appears to be a promising solution to achieve this goal.

In this study, the proposed caching algorithm employs a trial-and-error approach, which significantly reduces algorithmic complexity. This design choice facilitates real-time decision-making, allowing the algorithm to adapt quickly to changes in service demand and user behavior, which is critical in practical applications within smart buildings. While this method may not always yield the globally optimal solution, it provides flexibility in responding to fluctuating service requests. Such adaptability is especially important in environments where the popularity of services can vary greatly. Furthermore, it is essential to acknowledge potential challenges associated with this approach, particularly in highly volatile scenarios where user preferences and service demands change rapidly. Addressing these challenges ensures that our proposed algorithm remains practical and relevant for real-world service caching applications.

5 Conclusion

This paper introduced a novel dynamic crowdsourcing and caching algorithm for optimizing service delivery in smart buildings, leveraging IoT gateways and edge servers to reduce latency and enhance the responsiveness of smart city applications. The proposed efficiency metric (Ei) balances cloud-computed and edge-computed response times, facilitating optimal caching decisions. Experimental results demonstrated significant improvements in average response time (ART) and resource usage compared to traditional methods.

One of the limitations of our approach is the potential performance challenge in resource-constrained environments when dealing with a large number of services. However, in real-world scenarios, where the number of services is typically fewer than 50, the system performs effectively. Future research could focus on refining the efficiency metric to incorporate additional factors such as security and data privacy, addressing the increasing concerns in IoT networks. Further exploration of scalability in larger smart city environments will also be valuable.

The potential applications of this algorithm span several areas within smart city infrastructure. Smart homes and buildings can benefit from improved service delivery for applications like energy management, security systems, and automated maintenance. Healthcare systems, particularly in remote patient monitoring, could leverage edge caching to reduce latency in data transmission from IoT sensors. Traffic management and environmental monitoring are other critical domains where latency reduction and resource optimization are essential for real-time decision-making in smart cities.

In terms of next steps, further optimization of the algorithm is needed to handle more complex service environments with an increasing number of services and dependencies. Additionally, integration with advanced technologies like machine learning and predictive analytics could enhance the algorithm's adaptability, allowing it to predict future service demand and optimize caching decisions accordingly. Investigating the deployment of this algorithm in diverse IoT environments, such as industrial IoT and agricultural IoT, could broaden its applicability and demonstrate its versatility across various sectors.

Our research outcomes demonstrate significant improvements in average response time (ART) and resource utilization compared to existing algorithms, including Particle Swarm Optimization (PSO) and Genetic Algorithms (GA). While our approach effectively balances performance and resource consumption, we acknowledge that it may not yield the globally optimal solution in all scenarios, particularly when dealing with larger sets of services.

By situating our research within the broader landscape of IoT service provisioning, we aim to contribute meaningfully to the ongoing discourse in this evolving field. This work provides a strong foundation for future studies in mobile edge computing and IoT service provisioning. By extending the current solution and addressing the identified limitations, the proposed approach could play a pivotal role in improving the performance of smart city infrastructures and other resource-constrained IoT ecosystems.

Data availability statement

The original contributions presented in the study are included in the article/supplementary material, further inquiries can be directed to the corresponding author.

Author contributions

MH: Conceptualization, Data curation, Formal Analysis, Funding acquisition, Investigation, Methodology, Project administration, Resources, Software, Supervision, Validation, Visualization, Writing–original draft, Writing–review and editing. AH: Investigation, Methodology, Project administration, Supervision, Writing–review and editing. JB: Conceptualization, Data curation, Formal Analysis, Funding acquisition, Investigation, Methodology, Project administration, Resources, Software, Supervision, Validation, Visualization, Writing–review and editing.

Funding

The author(s) declare that no financial support was received for the research, authorship, and/or publication of this article.

Conflict of interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's note

References

Abbas, A., Raza, A., Aadil, F., and Maqsood, M. (2021). Meta-heuristic-based offloading task optimization in mobile edge computing. *Int. J. Distributed Sens. Netw.* 17 (6), 155014772110230. doi:10.1177/15501477211023021

Aghazadeh, R., Ali, S., and Mostafa, G.-A. (2023). Proactive content caching in edge computing environment: a review. *Softw. Pract. Exp.* 53 (3), 811–855. doi:10.1002/spe.3033

Aljubayrin, S., Aldehim, G., Alruwais, N., Mahmood, K., Al Duhayyim, M., Min, H., et al. (2023). Dynamic offloading strategy for computational energy efficiency of wireless power transfer based MEC networks in industry 5.0. *J. King Saud University-Computer Inf. Sci.* 35 (10), 101841. doi:10.1016/j.jksuci.2023.101841

Alzakari, N., Dris, A. B., and Alahmadi, S. (2020). "Randomized least frequently used cache replacement strategy for named data networking," in 2020 3rd International Conference on Computer Applications and Information Security (ICCAIS), Riyadh, Saudi Arabia, 19-21 March 2020 (IEEE), 1–6.

Bibri, S. E., Krogstie, J., Kaboli, A., and Alahi, A. (2024). Smarter eco-cities and their leading-edge artificial intelligence of things solutions for environmental sustainability: a comprehensive systematic review. *Environ. Sci. Ecotechnology* 19, 100330. doi:10.1016/j.ese.2023.100330

Bouramdane, A.-A. (2023). Optimal water management strategies: paving the way for sustainability in smart cities. *Smart Cities* 6 (5), 2849–2882. doi:10.3390/smartcities6050128

Busetti, R., El Ioini, N., Barzegar, H. R., and Pahl, C. (2022). "Distributed synchronous particle swarm optimization for edge computing," in 2022 9th International Conference on Future Internet of Things and Cloud (FiCloud), Rome, Italy, 22-24 August 2022 (IEEE), 145–152.

Deng, S., Xiang, Z., Yin, J., Taheri, J., and Zomaya, A. Y. (2018). Composition-driven IoT service provisioning in distributed edges. *IEEE Access* 6, 54258–54269. doi:10.1109/access.2018.2871475

Gou, F., and Wu, J. (2024). *Optimization of edge server group collaboration architecture strategy in IoT smart cities application*. Peer-to-Peer Networking and Applications, 1–23.

Gupta, D., Rani, S., and Ahmed, S. H. (2023). ICN-edge caching scheme for handling multimedia big data traffic in smart cities. *Multimedia Tools Appl.* 82 (25), 39697–39717. doi:10.1007/s11042-022-13518-3

Haraty, R. A., and Nancy, R. (2023). Network traffic analysis as a strategy for cache management. *Int. J. Comput. Digital Syst.* 14 (1), 10347–10359. doi:10.12785/ijcds/1401106

Harnal, S., Sharma, G., Malik, S., Kaur, G., Khurana, S., Kaur, P., et al. (2022). Bibliometric mapping of trends, applications and challenges of artificial intelligence in smart cities. *EAI Endorsed Scal Inf. Syst.* 9 (4), e8. doi:10.4108/eetsis.vi.489

Hasan, M. K., Jahan, N., Nazri, M. Z. A., Islam, S., Khan, M. A., Alzahrani, A. I., et al. (2024). Federated learning for computational offloading and resource management of vehicular edge computing in 6G-V2X network. *IEEE Trans. Consumer Electron.* 70, 3827–3847. doi:10.1109/tce.2024.3357530

Hua, H., Li, Y., Wang, T., Dong, N., Li, W., and Cao, J. (2023). Edge computing with artificial intelligence: a machine learning perspective. *ACM Comput. Surv.* 55 (9), 1–35. doi:10.1145/3555802

Khanh, Q.Vu, Nguyen, V. H., Minh, Q. N., Dang Van, A., Le Anh, N., and Chehri, A. (2023). An efficient edge computing management mechanism for sustainable smart cities. *Sustain. Comput. Inf. Syst.* 38, 100867. doi:10.1016/j.suscom.2023.100867

Mulero-Palencia, S., and Monzon Baeza, V. (2023). Detection of vulnerabilities in smart buildings using the shodan tool. *Electronics* 12 (23), 4815. doi:10.3390/electronics12234815

Natesha, B. V., and Guddeti, R. M. R. (2022). Meta-heuristic based hybrid service placement strategies for two-level fog computing architecture. *J. Netw. Syst. Manag.* 30 (3), 47. doi:10.1007/s10922-022-09660-w

Nauman, A., Alruwais, N., Alabdulkreem, E., Nemri, N., Aljehane, N. O., Dutta, A. K., et al. (2023). Empowering smart cities: high-altitude platforms based mobile edge computing and wireless power transfer for efficient IoT data processing. *Internet Things* 24, 100986. doi:10.1016/j.iot.2023.100986

Nguyen, T. V., Tran, A. T., Dao, N. N., Moon, H., and Cho, S. (2023). Information fusion on delivery: a survey on the roles of mobile edge caching systems. *Inf. Fusion* 89, 486–509. doi:10.1016/j.inffus.2022.08.029

Oliveira, F., Pereira, P., Dantas, J., Araujo, J., and Maciel, P. (2023). Dependability evaluation of a smart poultry house: addressing availability issues through the edge, fog, and cloud computing. *IEEE Trans. Industrial Inf.* 20, 1304–1312. doi:10.1109/tii.2023.3275656

Pallewatta, S., Kostakos, V., and Buyya, R. (2023). Placement of microservices-based iot applications in fog computing: a taxonomy and future directions. *ACM Comput. Surv.* 55 (14s), 1–43. doi:10.1145/3592598

Songhorabadi, M., Rahimi, M., MoghadamFarid, A., and Haghi Kashani, M. (2023). Fog computing approaches in IoT-enabled smart cities. *J. Netw. Comput. Appl.* 211, 103557. doi:10.1016/j.jnca.2022.103557

Sudha, I., Mustafa, M. A., Suguna, R., Karupusamy, S., Ammisetty, V., Shavkatovich, S. N., et al. (2023). Pulse jamming attack detection using swarm intelligence in wireless sensor networks. *Optik* 272, 170251. doi:10.1016/j.ijleo.2022.170251

Suhag, D., and Jha, V. (2023a). A comprehensive survey on mobile crowdsensing systems. *J. Syst. Archit.* 142, 102952. doi:10.1016/j.sysarc.2023.102952

Suhag, D., and Jha, V. (2023b). A comprehensive survey on mobile crowdsensing systems. *J. Syst. Archit.* 142, 102952. doi:10.1016/j.sysarc.2023.102952

Walia, G. K., Kumar, M., and Gill, S. S. (2023). AI-empowered fog/edge resource management for IoT applications: a comprehensive review, research challenges and future perspectives. *IEEE Commun. Surv. and Tutorials* 26, 619–669. doi:10.1109/comst.2023.3338015

Wu, J., Cao, Z., Zhang, Y., and Zhang, X. (2019). "Edge-cloud collaborative computation offloading model based on improved partical swarm optimization in MEC," in 2019 IEEE 25th International Conference on Parallel and Distributed Systems (ICPADS), Tianjin, China, 04-06 December 2019 (IEEE), 959–962.

Zhang, D., Sun, G., Zhang, J., Zhang, T., and Yang, P. (2023). Offloading approach for mobile edge computing based on chaotic quantum particle swarm optimization strategy. *J. Ambient Intell. Humaniz. Comput.* 14 (10), 14333–14347. doi:10.1007/s12652-023-04672-z

Zhou, Z., Wu, Q., and Chen, X. (2019). Online orchestration of cross-edge service function chaining for cost-efficient edge computing. *IEEE J. Sel. Areas Commun.* 37 (8), 1866–1880. doi:10.1109/jsac.2019.2927070

Zulfa, M. I., Hartanto, R., Permanasari, A. E., and Ali, W. (2022). LRU-GENACO: a hybrid cached data optimization based on the least used method improved using ant colony and genetic algorithms. *Electronics* 11 (19), 2978. doi:10.3390/electronics11192978