A Fog Computing-Based Collaborative Information Resource System for Smart Cities: TSAS and KLED Algorithms for Data Transmission and Service Deployment

Ya Xiao[1*], Jinyeol Jung[2]
[1]Academy of Art and Design, Yancheng Teachers Unversity, Yancheng, 224007, China
[2]Smart Experience Design Department, TED, Kookmin University, Seoul, 02707, Korea
E-mail: 13401780510@163.com, 18862009559@163.com
[*]Corresponding author

Keywords: collaborative prediction mechanism, data transmission volume compression, fog computing, smart cities, service edge deployment

Received: Januar 15, 2025

Effective integration and scheduling of information resources play a pivotal role in realizing intelligent management within the framework of smart city development. This research endeavors to overcome two significant hurdles: the redundancy in data transmission during the acquisition phase at the network's edge and the inefficiencies encountered when deploying analytical services across diverse edge devices. To address these challenges, a collaborative system architecture rooted in fog computing is introduced. A prediction mechanism, driven by spatiotemporal correlations, is incorporated to dynamically modulate data transmission intervals. This adjustment effectively curtails unnecessary synchronization, thereby enhancing the efficiency of data acquisition. Moreover, a deployment strategy based on multi-objective optimization is devised to allocate analytical tasks among edge devices constrained by limited resources, aiming to minimize the overall execution time. Experimental evaluations carried out on a real-world dataset encompassing 54 sensing terminals reveal that the proposed synchronization mechanism outperforms two traditional methods, reducing the false alarm rate by 58.90% and 31.35%, respectively, with a minimum mean absolute error of 2.6×10^{-5}. Additionally, the deployment strategy achieves an average reduction of 13.12% in service completion time across four standard scientific workflow structures. The system adeptly alleviates bandwidth constraints and computational limitations inherent in edge networks, providing a practical and effective solution for efficient data transmission and task scheduling in extensive smart city environments.

Povzetek: Opisan je nov sistem pametnih mest, ki z algoritmoma TSAS in KLED na osnovi megličnega računalništva zmanjša prenose podatkov in optimira razmestitev storitev znotraj robnega omrežja.

1 Introduction

Smart cities leverage information technologies and intelligent methodologies to facilitate real-time awareness of urban operations, seamless data integration, and informed decision-making, thereby enhancing urban governance and the delivery of public services [1]. In practical implementations, a vast array of intelligent devices is extensively deployed in public infrastructure scenarios, such as transportation, energy, and environmental monitoring, culminating in a comprehensive urban sensing network [2]. These devices continuously produce multi-source, heterogeneous inspection data, offering abundant informational resources for urban management. However, the sensing data in smart city environments is marked by its widespread distribution, substantial volume, and high diversity, presenting challenges for existing systems in data collection, synchronized transmission, and unified management. On one hand, various types of terminal devices display discrepancies in data formats, upload frequencies, and interface protocols, leading to inefficiencies in data integration [3-4]. On the other hand, the conventional cloud-centric architecture is plagued by issues like centralized computational burdens and elevated response latencies, potentially impeding its capacity to satisfy the real-time and region-specific processing requirements of city-level applications [5-6]. In recent years, edge computing has surfaced as an extension of cloud computing, offloading a portion of the computational workload to devices situated closer to the data source, thereby effectively diminishing latency [7]. Nevertheless, edge computing typically conducts data processing locally at individual terminal nodes, lacking coordination mechanisms and intermediate-layer management among nodes. Conversely, fog computing introduces an intermediary "fog node" layer between the cloud and the edge, enabling regional collaboration, task offloading, and resource-aware scheduling among

distributed nodes. This architecture is more adeptly suited for smart city scenarios characterized by heterogeneous sensor distributions, diverse services, and region-based collaborative prerequisites [8]. Consequently, this study introduces a collaborative information resource system tailored for smart cities, grounded in fog computing, which decentralizes computing power from centralized cloud centers to fog edge devices. An innovative approach is proposed, integrating the K-Means clustering algorithm with Gaussian distribution, to be incorporated into the system's sensing devices. This integration aims to minimize errors in the cooperative prediction mechanism. Furthermore, the enhanced algorithm and system generate increased broadband capacity, thereby expanding both the number of fog end edge devices and network broadband within smart city environments. The objective of this study is to tackle challenges including data redundancy, inefficient synchronization, and prolonged response latency in service deployment within fog-edge environments, with the ultimate goal of improving the real-time performance and deployment efficiency of information resource systems in smart cities. To achieve this, the research zeroes in on two primary issues: (1) devising effective methods for synchronizing and compressing multi-source sensing data in edge networks, and (2) developing strategies for optimal service deployment and scheduling on fog computing nodes with limited resources.

2 Related works

The collaborative information resource system designed for smart cities centers on two key challenges: the collection of perceptual data and the deployment strategy for edge networks. The temporal data collection process of sensing terminals is marked by data redundancy and the transmission of large volumes of data. Fei and Ma introduced the throughput rate constraint and sensing strategy in fog computing to the fog access point, and established a prediction model using multi-information sensing to infer the degree distribution of fog nodes in the network, and obtained a network transmission efficiency and environmental sensing accuracy close to 90% [9]. Ali et al. targeted at smart public safety and traffic management in cities, proposed a dynamic deep hybrid spatio-temporal neural network for high-precision traffic flow prediction, and introduced an exaggerated method of attention mechanism for city-wide short-time crowd flow prediction. The latter improved the accuracy by about 20.8% and 8.8% over the former on two traffic datasets [10]. Zhang and Li addressed the existence of data transmission mechanism in vehicular network Real-time, efficient computational tasks, and vehicle data privacy problems, and designed a data transmission mechanism for Telematics in fog computing environment. The mechanism proposed privacy-preserving task assignment and data aggregation mechanisms in a fog node-assisted crowd-sensing model for hilarious and secure data transmission [11]. Qiao et al. proposed a tool wear monitoring and prediction system based on a deep learning model and fog computing. The architecture included edge computing layer, fog computing layer and cloud computing layer, and the accuracy, response time and bandwidth consumption performance of the system were verified by prediction experiments [12]. Yang et al. used a numerical gradient-based approach to process raw data collected from marine survey work and designed an improved algorithm for multi-sensor information fusion, which effectively improved the quality of marine data and information utilization efficiency [13].

Rational deployment of analytic services in the fog edge network is the key to provide high performance and low latency computing services. Núñez-Gómez et al. proposed an architecture based on the Ethernet blockchain to address the resource coordination problem due to the distributed nature of fog computing. The experimental results verified its low overhead and efficient resource coordination when introduced into the system [14]. Huang et al. designed a multi-objective model including deployment cost and service latency for the deployment of services among nodes in fog computing, and proposed a multi-copy ant colony optimization algorithm for model solving to obtain a solution strategy with qualified diversity and accuracy [15]. Ayoubi et al. designed an autonomous IoT service placement method including four phases: monitoring, analysis, decision making and execution, which prioritizes requested services based on service deadlines and the state of available resources [16]. Ning et al. constructed a three-layer in-vehicle fog computing model to experiment distributed traffic management for the real-time traffic management problem in smart cities and validated the model based on performance analysis of real skid trajectories [17]. A decentralized optimization strategy for service layout in fog computing was proposed by Guerrero et al. The core idea of the strategy was to place the most popular services close to the users. The experimental results showed that this decentralized algorithm improved the net clusters usage and service latency. To facilitate a structured comparison, Table 1 summarizes the key techniques, outcomes, and limitations of recent fog computing-based approaches for smart city applications.

Table 1: Work comparison table

Author(s)	Method / Application area	Key techniques	Results	Limitations
Fei & Ma [9]	Fog-based network prediction	Throughput constraint; multi-source sensing; degree distribution inference	Achieved ~90% network transmission efficiency and environmental	Did not address adaptive synchronization or real-time transmission optimization

			sensing accuracy	
Ali et al. [10]	Urban traffic & crowd flow prediction	Hybrid spatio-temporal neural network; attention mechanism	Accuracy improved by 20.8% and 8.8% on two datasets	Focused only on prediction, lacks deployment or transmission considerations
Zhang & Li [11]	Telematics data privacy in vehicular fog networks	Privacy-preserving task assignment; fog-assisted crowdsensing	Secure and efficient data transmission mechanism	No dynamic scheduling or real-time model adaptation
Qiao et al. [12]	Tool wear monitoring with deep learning and fog layers	Layered architecture (edge–fog–cloud); deep prediction model	Improved accuracy, reduced latency and bandwidth use	Application-specific; lacks general deployment strategy
Yang et al. [13]	Marine survey data processing	Numerical gradient; multi-sensor data fusion algorithm	Enhanced marine data quality and utilization	Not designed for dynamic or scalable urban edge scenarios
Núñez-Gómez et al. [14]	Fog resource coordination via blockchain	Ethernet-based blockchain; resource negotiation model	Low system overhead, efficient resource coordination	High complexity of blockchain deployment; lacks scheduling optimization
Huang et al. [15]	Multi-objective fog service deployment	Ant colony optimization; latency-cost tradeoff model	Balanced service diversity and accuracy	No integration of prediction-based edge traffic regulation
Ayoubi et al. [16]	Autonomous IoT service placement	Monitor-analyze-decide-execute model; resource-aware prioritization	Adaptive task allocation based on service deadlines	Static scheduling model; lacks time-sensitive transmission modeling
Ning et al. [17]	Vehicular fog computing for traffic management	Three-layer fog structure; real trajectory-based validation	Verified effective for distributed real-time traffic control	Focused on traffic domain; lacks general resource service framework

In summary, although existing studies have explored aspects of data compression, prediction modeling, and service deployment in fog computing environments, there remains a lack of unified solutions addressing the combined challenges of data redundancy and large-scale transmission, particularly under heterogeneous resource constraints. Furthermore, while some works consider the distribution of sensing and computing tasks, few explicitly address the synchronization and coordination between them. These overlapping challenges constitute the primary focus of this study. To address these limitations, this study proposes a collaborative information resource system tailored to fog computing environments. It introduces a spatiotemporal synchronization mechanism to reduce redundant data transmission and a multi-objective deployment strategy to optimize task scheduling across edge nodes. Together, these innovations enable dynamic, efficient, and scalable service coordination for smart city scenarios.

3 Materials and methods

3.1 Fog-end collaborative prediction mechanism for transmission volume compression

The relocation of system services from centralized cloud computing centers to decentralized fog computing edge networks, with the aim of delivering high-quality services with minimal latency, has emerged as a prominent research focus and formidable challenge [18-19]. In this context, the study proposes a collaborative prediction-based transmission compression scheme tailored for fog computing environments, and introduces a spatiotemporal-aware adaptive synchronization algorithm (TSAS). Rather than only adjusting synchronization waiting time, TSAS jointly utilizes residual modeling and trend prediction to dynamically determine the optimal synchronization interval and prediction window, thereby improving the timeliness and accuracy of data uploads under bandwidth-constrained conditions. In addition, a service edge deployment algorithm based on improved Kernighan-Lin algorithm (KLED) is proposed to further optimize the system performance. KLED is designed to deploy

computationally intensive services such as traffic flow prediction and anomaly detection in environmental data—key analytical tasks in smart city applications—onto resource-constrained edge devices. These services typically require low-latency response and distributed coordination, which KLED optimizes through adaptive resource-aware placement strategies. In the collaborative prediction mechanism, the set of n data continuously collected by the sensing terminal at a specific sampling frequency is defined as shown in Equation (1).

$$D=\left\{\left(t_1,x_1\right),\left(t_2,x_2\right),\cdots,\left(t_i,x_i\right),\cdots\left(t_n,x_n\right)\right\} \quad (1)$$

In Equation (1), t_i represents the data acquisition time and x_i represents the data measurement value at that moment. Assuming that the predicted value of the data at that moment is $\hat{x}_t$, then the fitting error $E_f=\left|\hat{x}_t-x_t\right|$ at the moment of t can be calculated, and the size of the fitting error is compared with the error threshold set by the system to determine whether to synchronize the actual measured data to the fog node [20]. It should be noted that the fitting error here refers to the instantaneous residual at each time point, used solely to describe the pointwise prediction deviation, and should not be interpreted as a measure of the overall model fitting performance.

The data acquired through continuous sampling utilizing the sensing terminal is organized in chronological sequence, signifying that it constitutes a time series. Consequently, this collected data can be leveraged to forecast the measured values for an upcoming timeframe. Considering the limited computational resources of edge nodes, an autoregressive model is adopted in this study for short-term sensing data prediction. The model offers a simple structure and fast inference, enabling stable predictions without the need for large-scale training or complex prior assumptions, making it well-suited for heterogeneous and low-power fog-edge environments. In contrast, models such as LSTMs require extensive training and high memory usage, while Kalman filters depend on accurate prior modeling of system dynamics and noise, which are difficult to define in heterogeneous edge environments.

To characterize the linear correlation between the current data and its historical states, this study adopts an autoregressive model to predict the sensing data, as shown in Equation (2).

$$\hat{x}_t=\rho_1x_{t-1}+\rho_2x_{t-2}+\cdots+\rho_ix_{t-i}+\cdots+\rho_nx_{t-n}+\varepsilon_t \quad (2)$$

In Equation (2), $\left\{\rho_1,\rho_2,\cdots,\rho_n\right\}$ is the sequence of model parameters to be calculated. Considering that there are often some difficult-to-observe errors in the actual measurement process, the study also introduces white noise $\varepsilon_t\sim N\left(0,\sigma^2\right)$, ρ_i refers to the i th autoregressive parameter. Based on the idea of multiple regression theory, least squares estimation is used to calculate the parameters of the prediction model [21-22]. After substituting the sequence of the collected data, the matrix expression of the time-series linear relationship between the measured and predicted values of the data is $y=x\rho+\varepsilon$, and the estimation of the parameter ρ can be obtained as shown in Equation (3).

$$\hat{\rho}=\left(x^Tx\right)^{-1}x^Ty \quad (3)$$

When the fog edge device finishes model modeling, it synchronizes the latest model parameters to the corresponding sensing terminal, and then the sensing terminal completes data collection, and the detailed interaction between the two is shown in Figure 1.

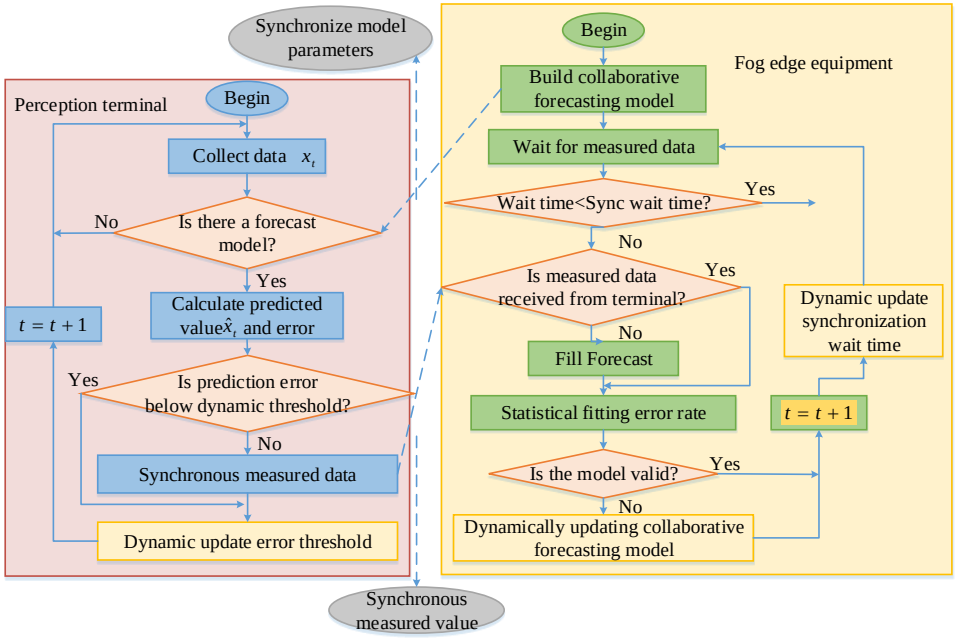


Figure 1: Flow chart of data acquisition by fog terminal synchronous prediction mechanism

In Figure 1, "synchronization wait time" refers to the minimum interval required after the last synchronization during which the sensing terminal refrains from uploading data unless the prediction error exceeds the threshold. This mechanism helps reduce data transmission frequency when prediction accuracy remains acceptable. "Synchronous measured value" denotes the original data point identified for upload, triggered either by the prediction error exceeding the threshold or the wait time reaching its upper limit. Figure 1 illustrates the collaborative synchronization process between the perception terminal and the fog computing node. The orange elements indicate three key parameters that are dynamically updated: the maximum allowable error threshold, the prediction model parameters, and the synchronization wait time. The error threshold is adaptively adjusted based on trends in the measured data to enhance the responsiveness of the synchronization trigger; the prediction model is optimized using residual statistics within a sliding window to improve adaptability to data pattern changes; and the synchronization wait time is updated according to data validity and response frequency to reduce unnecessary transmissions. These three mechanisms work collaboratively to significantly improve the system's performance in terms of data compression and transmission efficiency.

3.2 Fog-end adaptive synchronization algorithm based on spatio-temporal correlation

To address the real-time demands of the collaborative prediction mechanism, an analysis of the adaptive synchronization algorithm for fog-end edge devices is conducted. Additionally, in order to overcome spatial limitations, the deployment of a communication transmission scheme for fog-end devices within the edge network is investigated. Consequently, this study introduces the TSAS algorithm. TSAS is well-suited for high-frequency urban sensing scenarios, such as traffic flow monitoring or environmental noise detection, where sensing terminals are densely distributed. Given the temporal and spatial continuity of such data, TSAS leverages historical measurements to predict the next value and only triggers data synchronization when the prediction error exceeds a predefined threshold or the waiting time reaches a limit. This mechanism significantly reduces network transmission load and extends the operational life of sensing devices.

Assuming that a fog edge device is responsible for maintaining the data collection process of the prediction mechanism of sensing terminals, the N N terminals are divided into multiple clusters, the set of nodes of the m th cluster is defined as $G^m = \{p_1, p_2, \cdots, p_i, \cdots, p_k\}$, and a cluster contains k sensing devices. The synchronization waits time threshold and the communication link transmission delay of a sensing terminal p_i at t are defined as $v_i(t)$ and $s_i(t)$, respectively [23]. The synchronization wait time needs to be greater than the transmission delay to correctly trigger the subsequent data processing process, i.e., if $v_i(t) < s_i(t)$ at time t is defined as a false alarm. Therefore, the value $\omega(i,t)$ represents the deviation between the expected synchronization waiting time and the observed transmission delay for sensing device p_i at time t as defined in Equation (4).

$$\omega(i,t) = v_i(t) - s_i(t) \tag{4}$$

Also, $M_{\omega(i,t)}$ is defined as whether a miscue occurs when $M_{\omega(i,t)} = 0$ then $v_i(t) - s_i(t) \geq 0$, indicating that the synchronization wait time at moment t predicts that no miscue occurs [24-25]. Thus, the miscalculation rate is calculated as shown in Equation (5).

$$ER(i) = \frac{\sum_t M_{\omega(i,t)}}{|s_i|} \tag{5}$$

In Eq. (5), $ER(i)$ denotes the false positive rate during the observed time period, which is the ratio of the synchronization wait time not larger than the actual link transmission delay. $ER(i) \in [0,1]$. The smaller its value, the more accurate the prediction of the synchronous wait time threshold. the flow of the TSAS algorithm is shown in Figure 2.

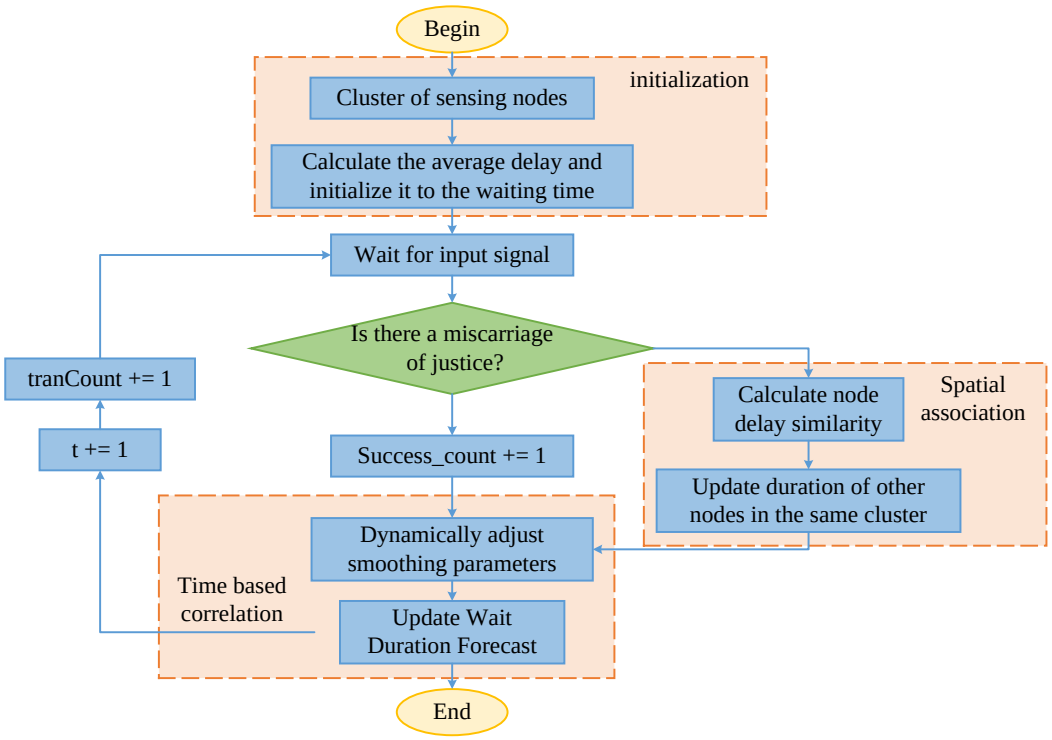


Figure 2: Adaptive synchronization algorithm of fog terminal based on spatio-temporal correlation

In the initial stage of the fog-end collaborative prediction model, each sensing terminal needs to send k real sensing data to the fog edge device as the data base for model training. Therefore, the transmission delay sequence of message sensing devices is defined as shown in Equation (6).

$$s_i(t)=\left\{s_i(t_1),s_i(t_2),\cdots,s_i(t_k)\right\},p_i\in G^m \quad (6)$$

Based on Equation (6), the message transmission delay sequences of all sensing devices in the cluster can be further obtained, and the average value of all transmission delay sequences is employed as the initial value of the waiting time threshold. Based on the assumption that synchronization wait time exhibits temporal continuity, this study employs the Exponential Smoothing Model (ESM) to fit its variation trend during the data acquisition process in the fog-edge network. The synchronization wait time prediction model is defined in Equation (7) [26–27].

$$v_i(t+1)=a_i(t)\times v_i(t)+\left(1-a_i(t)\right)\times s_i(t) \quad (7)$$

In Equation (7), $t+1$ represents the next prediction moment, and $v_i(t+1)$ denotes the predicted synchronization wait time for sensing terminal p_i at the moment of $t+1$. $a_i(t)$ is the adaptive smoothing factor computed based on the ratio between the cumulative smoothing error and the cumulative absolute error. This adaptive update enables the system to dynamically adjust scheduling intervals in response to varying data transmission conditions [28-29]. The expressions of the cumulative smoothing error and the cumulative smoothing absolute error are shown in Equation (8).

$$\begin{cases}E_s(i,t)=\gamma*E_s(i,t-1)+(1-\gamma)*ER(i,t)\\E_{as}(i,t)=\gamma*E_{as}(i,t-1)+(1-\gamma)*\left|ER(i,t)\right|\end{cases} \quad (8)$$

In Eq. (8), γ is a weighting factor with a value between 0 and 1, which generally takes the value of $[0.1,0.2]$. The magnitude and frequency of change of the prediction error of the sensing device $ER(i,t)$ can reflect the fluctuation of the network state, so the smoothing coefficient of the delay prediction model is implemented by $ER(i,t)$. Based on the assumption that synchronization wait times among neighboring terminals exhibit spatial correlation, the study introduces a similarity function $SS(j,t_i)$ to quantify the temporal alignment between two sensing terminals i and j within the same spatial cluster. The definition is provided in Equation (9) [30].

$$SS(j,t_i)=1-\frac{\left|v_i(t)-v_j(t)\right|}{v_j(t)},i\in G^m,j\in G^m \quad (9)$$

In Equation (9), v_i and v_j represent the synchronization wait times of nodes i and j, respectively. A higher value of $SS(j,t_i)$ indicates a stronger spatial similarity in synchronization behavior. Instead of relying on physical distance or transmission delay, this method uses temporal similarity to approximate the overlap in communication conditions between terminals and fog nodes. When a false alarm or abnormal update occurs at node j, the spatial

correlation is leveraged to adjust the synchronization wait time prediction of node i, as defined in Equation (10).

$$v_j(t) = a_j(t) * v_j(t) + (1 - a_j(t)) * SS(j, t|i) * s_i(t), j \in N(i) \quad (10)$$

In Eq. (10), $a_j(t)$ is the ESM parameter of the sensing terminal v_j, and $N(i)$ represents the set of sensing terminals in the cluster excluding v_i. This collaborative update mechanism enhances robustness in synchronization within the fog-edge network by enabling terminals to modify their behavior in accordance with spatially correlated neighbors, thereby mitigating the risk of misclassification.

The core computational operation of TSAS lies in the autoregressive prediction based on a sliding window, where the window length is n. This results in both time and space complexity of O(n). The dynamic error threshold is determined by fitting historical residuals, which incurs negligible computational cost compared to the prediction process and can be considered constant time. Therefore, TSAS maintains low resource consumption while ensuring synchronization accuracy, making it suitable for real-time prediction tasks on edge devices.

To enhance the real-time performance of the synchronization mechanism, the TSAS algorithm incorporates sliding window error statistics and dynamic threshold adjustment. The sliding window mechanism dynamically captures data fluctuations and adjusts model parameters in real-time. Simultaneously, the error threshold is calibrated—either tightened or relaxed—based on historical residual trends, to preclude excessive synchronization triggered by localized anomalies. This approach effectively balances the frequency of synchronization with the precision of predictions. In addition, to address external factors such as network latency and data fluctuations, TSAS employs an exponential smoothing model to dynamically estimate the waiting time, and integrates a spatial correlation mechanism that adapts local strategies based on the synchronization results of neighboring nodes, effectively improving system stability and robustness in complex environments.

3.3 Collaborative analysis service edge deployment and system design for smart city information resources

In the fog computing network scenario, the adjustment of the edge network deployment scheme mainly focuses on the real-time performance of computational operations and the robustness of business operations. In the communication delay between the sensor terminal and the fog edge device, the TSAS algorithm improves the spatial otemporal continuity and correlation of communication. The study improves the service deployment algorithm KLED for smart city information resource collaboration system in order to achieve high performance and low latency information resource analysis service. KLED targets distributed intelligent service deployment scenarios in fog computing environments, such as urban video surveillance analysis or emergency response coordination. In cases where services exhibit dependency relationships and edge nodes possess heterogeneous resource capabilities, KLED utilizes K-means clustering to group nodes based on their capacities and deploys services by considering both task dependencies and communication costs. This approach effectively shortens service makespan and enhances parallel execution efficiency, making it suitable for dynamic multi-task scheduling in smart city applications. The flowchart of the KLED algorithm is shown in Figure 3.

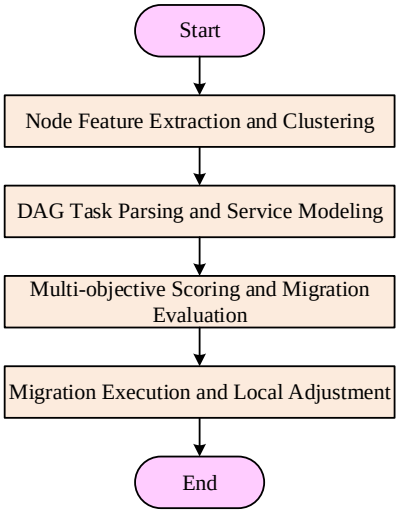


Figure 3: Flowchart of the KLED algorithm

The KLED algorithm consists of four stages: node clustering, DAG-based task modeling, heuristic scoring for migration evaluation, and iterative deployment adjustment. It enables efficient service placement by balancing latency, resource load, and execution time in fog computing environments.

The algorithm abstracts the analysis service modeling of the system into a Direct Acyclic Graph (DAG), where each vertex represents a discrete service component or task. Each vertex is associated with a set of attributes including resource requirements (e.g., CPU, memory), estimated execution time, and service type (e.g., sensing, processing, aggregation). The directed edges in the DAG represent strict dependency constraints between tasks, indicating that the source node must complete before the target node begins. This structure allows the system to capture execution order, support parallelism where applicable, and optimize service placement under resource constraints.

The DAG is represented as $G_k = \langle F, E \rangle$, where represents the set of $F = \{f_1, f_2, \cdots f_i, \cdots, f_N\}$ N subservices, and $E = \{h(f_i, f_j) \mid f_i, f_j \in F\}$ represents the set of directed edges in the DAG. $h(f_i, f_j)$ represents the size of the amount of data transmitted by the service f_i to f_j, and if $h(f_i, f_j) > 0$ represents a dependency between two subservices. The existence of

M heterogeneous machines in the fog computing edge network is defined, denoted as $R=\{r_1,r_2,\cdots,r_p,\cdots,r_M\}$. Also define $G=\{g(r_p,r_q)|r_p,r_q\in R\}$ as the link communication relationship between different devices. The data transfer time of two services is determined by both the inter-device transfer bandwidth and the amount of data transferred. Assuming that the service is deployed on the machine f_i r_p and the service f_j is deployed on the machine r_q, the data transfer time between the two services is calculated as shown in Equation (11).

$$TT(f_i,f_j)=\frac{h(f_i,f_j)}{g(r_p,r_q)},r_p\neq r_q\cap g(r_p,r_q)>0 \quad (11)$$

In equation (11), $g(r_p,r_q)$ indicates the bandwidth of the transmission link between machines r_p and r_q, and if $g(r_p,r_q)=0$ indicates that there is no available communication link between the two devices. Due to the DAG data flow constraint, the service f_i must wait for the execution of the higher priority service on the machine r_p to complete before it can start execution. The study focuses on two improvements to the original Kernighan-Lin algorithm: service division policy change and multi-objective heuristic-based implementation of the node movement policy function. The service partitioning strategy uses "self-service migration" instead of the traditional graph 2 partitioning algorithm. The heuristic multi-objective movement decision refers to the movement of nodes based on four heuristic rules: distance gain, additional communication gain $cG(f_i,r_p,r_q)$, parallelism gain $pG(f_i,r_p,r_q)$ and execution time gain $eG(f_i,r_p,r_q)$. The node movement decision function is shown in Equation (12).

$$MD(f_i,r_p,r_q)=dG(r_p,r_q)\times cG(f_i,r_p,r_q)\times pG(f_i,r_p,r_q)\times eG(f_i,r_p,r_q) \quad (12)$$

In Eq. (12), $dG(r_p,r_q)$ represents the distance gain. Among the four heuristic metrics in the scoring function, distance gain evaluates the proximity of the migrated service to the data source and the receiving endpoint, prioritizing deployment closer to user-side edge nodes. Communication gain assesses the expected reduction in data transmission delay after migration. Parallelism gain reflects the availability of the target node, facilitating concurrent task execution. Execution time saving indicates the anticipated decrease in task runtime on the new node. These metrics are combined with assigned weights to guide the optimal service migration path selection.

In fog computing environments, both the input and output data for services originate from sensing endpoints. Therefore, deploying services in close proximity to user-side edge devices can significantly reduce service feedback latency. The distance gain and execution time gain are defined as shown in Equation (13).

$$\begin{cases} dG(r_p,r_q)=\dfrac{dC(r_{src},r_{des},r_q)}{dC(r_{src},r_{des},r_p)} \\ eG(f_i,r_p,r_q)=\dfrac{EF(f_i,r_q)}{EF(f_i,r_p)} \end{cases} \quad (13)$$

In Equation (13), $dC(r_{src},r_{des},r_q)$ represents the sum of the distance of the computing device r_q from the data source device r_{src} and the final output receiving device. r_{des} $EF(f_i,r_q)$ denotes the execution time of the subservice on the device f r_q. The additional communication gain and parallelism gain are defined as shown in Equation (14).

$$\begin{cases} cG(f_i,r_p,r_q)=\dfrac{cC(f_i,r_q)}{cC(f_i,r_p)} \\ pG(f_i,r_p,r_q)=\dfrac{wC(f_i,r_q)}{wC(f_i,r_p)} \end{cases} \quad (14)$$

In Equation (14), $cC(f_i,r_q)$ represents the sum of the transmission time required to compute the data exchange of the subservice f_i with all the previous services and the successor services when it is deployed on the device r_q, while $wC(f_i,r_q)$ represents the sum of the running time of the sibling services of the subservice f_i when the subservice f_i is deployed on the device r_q. The overall architecture of the smart city information resource collaboration system based on the fog-end collaborative prediction mechanism and the improved KLED algorithm is shown in Figure 4.

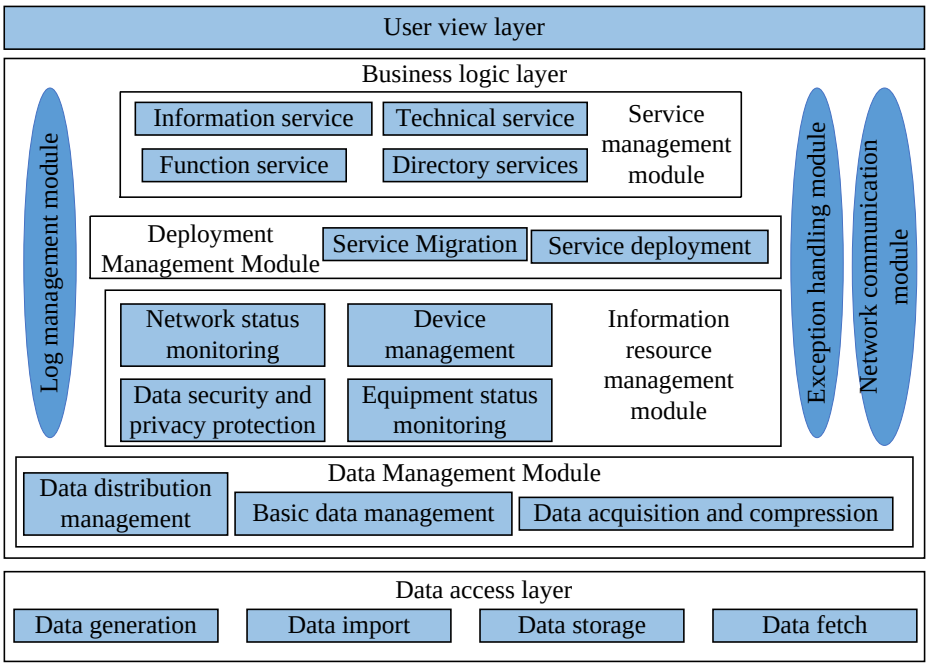


Figure 4: The overall architecture of smart city information resources collaboration system based on fog computing

As illustrated in Figure 4, the internal architecture of the smart city information resource collaboration system, which is based on fog computing, is primarily structured into three tiers: user interface, system logic, and database. Users engage with the backend system through the user interface, where the frontend is tasked with receiving user operation commands and translating them into backend operational logic. Following computation by the backend system logic, the outcome of the user's operation is rendered and presented back to the user via the user interface. Consequently, this demonstrates that the collaborative information resource system, designed on the foundation of fog computing, not only addresses issues related to data acquisition redundancy and transmission but also significantly enhances the operational efficiency and management capabilities of the system's analytical services.

The primary computational load of KLED is concentrated in the K-means clustering process, where N sensing nodes are partitioned into k clusters based on d-dimensional features over t iterations, resulting in a time complexity of O(Ndkt). The subsequent bandwidth estimation and service deployment scoring processes exhibit linear or logarithmic complexity. Overall, the algorithm balances deployment efficiency and scalability, making it well-suited for fog computing environments.

3.4 Parameter settings and tuning

To ensure the robustness and practicality of the proposed system, key parameters in both the TSAS and KLED algorithms were carefully tuned based on statistical principles and experimental validation: For the TSAS algorithm, the order of the autoregressive model was selected using the Akaike Information Criterion (AIC) to achieve optimal model parsimony. The prediction error threshold and waiting time limits were determined through preliminary experiments, aiming to minimize unnecessary synchronization while preserving data accuracy.

For the KLED algorithm, the number of clusters k was chosen using the elbow method based on node capability distribution. The weights of the deployment cost function were fine-tuned via grid search across a range of workloads, ensuring adaptability across heterogeneous fog environments. All parameters were validated through repeated trials under varying task loads and network conditions to confirm generalizability.

4 Results

4.1 Results of the fog-end collaborative prediction mechanism for transmission volume compression

In order to verify the operational performance of the smart city information resource collaboration system under fog computing, the effectiveness and accuracy of the proposed algorithms are verified respectively. The experiments were conducted on a representative fog-edge computing platform with the following hardware specifications: Intel Core i5-8265U processor (4 cores, 8 threads, 1.6GHz), 8GB DDR4 RAM, 256GB SSD storage, and Ubuntu 20.04 LTS operating system. No dedicated GPU was used, and all computations were performed on the CPU. This setup closely reflects the capabilities of typical low-power IoT gateways and fog nodes used in real-world deployments under constrained resource conditions. The hyperparameters in the experiments were determined through preliminary testing. The order of the autoregressive model was selected based on the Akaike Information Criterion (AIC), the number of clusters k in K-means was chosen

using the elbow method, and the cost function weights in the KLED algorithm were fine-tuned through multiple trial experiments. All parameters were validated on separate test sets to ensure the robustness of the results. The study first used K-means clustering algorithm to divide the sensing terminals into 7 clusters, and generated bandwidth capacity based on Gaussian distribution $Wb_i = N(\mu, \sigma^2)$, where μ is the mean value of link i bandwidth capacity and σ represents the variance of link bandwidth variance. To reduce the chance error, the mean value of five independent replicate experiments was used as the experimental result for each experiment. The experimental results of the data fit validity of the prediction mechanism are shown in Figure 5.

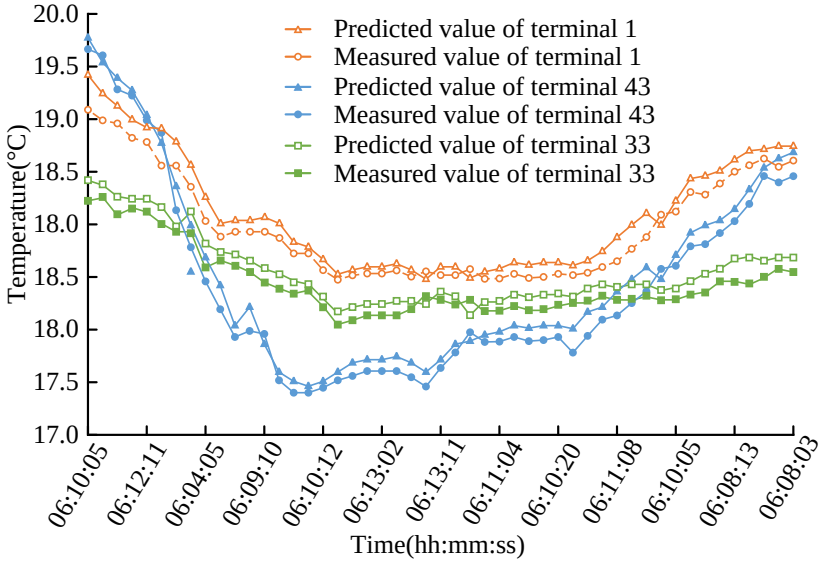
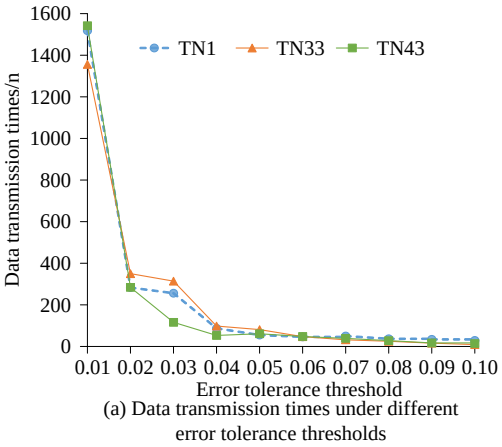
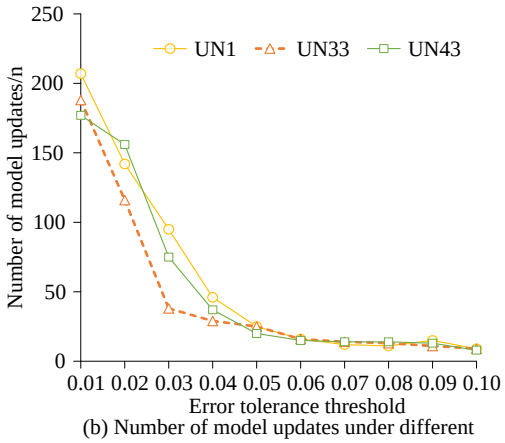


Figure 5: Fitting of measured value and predicted value in prediction mechanism

From the experimental results of fitting the temperature data in Figure 5, the predicted values obtained from the autoregressive model were highly fitted to the measured values in terms of variation trends and values. That is, the fog-end prediction mechanism could effectively compress the redundant data transmission volume under the edge network, which helped to reduce the pressure on the edge bandwidth. The validation results of the effectiveness of data transmission volume compression are shown in Figure 6.



(a) Data transmission times under different error tolerance thresholds



(b) Number of model updates under different error tolerance thresholds

Figure 6: Experimental results of the effectiveness of fog-end collaborative prediction mechanism to compress data transmission

As shown in Figure 6, a test range of 0.01 to 0.10 was selected based on preliminary experiments. When the threshold was set below 0.01, excessive synchronization requests were triggered, increasing communication overhead; when it exceeded 0.10, prediction accuracy dropped significantly. The selected range provided a well-balanced trade-off between transmission cost and prediction performance, making it suitable for edge computing environments. In the experiments on the effectiveness of data transmission volume compression, the reduction rate of data traffic within the edge network was used to focus on the reduction of data compression

rate. Experiments with different error tolerance thresholds on the number of data transmissions and the number of model updates were designed to verify the effect of the maximum error tolerance threshold on the number of communication interactions. From Figure 6(a), as the threshold value increased, the number of data transmissions decreased for all three terminals, especially in the interval from 0.01 to 0.04, while the trend was more stable in the interval from 0.04 to 0.10. Therefore, the higher the fitting accuracy of the prediction model, the less sensitive the fog-end collaborative prediction mechanism was to the error tolerance threshold. Figure 6(b) shows that as the threshold value decreased, the number of model updates gradually increased, i.e., the requirement for higher data accuracy, and thus the prediction model was required to provide more accurate data fit and prediction accuracy.

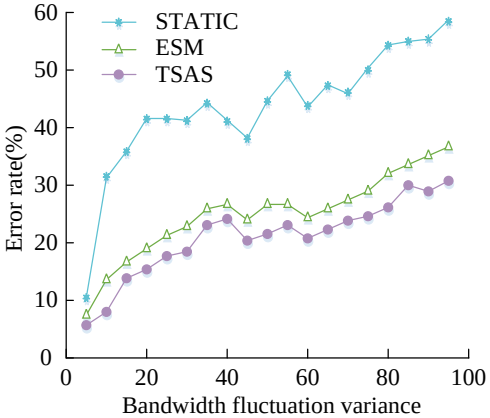
4.2 Performance of fog-end synchronization duration adaptive algorithm based on spatio-temporal correlation

Under the fog-end collaborative prediction mechanism, an effective and accurate synchronization wait time is required not only to reduce the chance of false positives, but also to shorten the waiting time of the device as much as possible without false positives, thus improving the efficiency of data acquisition. Therefore, the study used two metrics, namely, the misjudgment rate and mean absolute error (MAE), to verify the performance of the adaptive algorithm, and the experimental results are shown in Table 2.

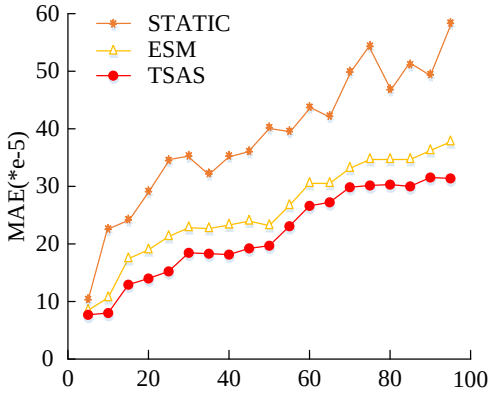
Table 2: Misjudgment rate and MAE of different algorithms

Algorithm	Misjudgment rate (%)			MAE (*e-5)		
	Maximum value	Average value	Minimum value	Maximum value	Average value	Minimum value
STATIC	52.0	30.9	13.7	5.52	3.62	1.53
ESM	39.9	18.5	15.8	5.06	3.04	1.39
TSAS	18.8	12.7	4.6	4.19	2.60	1.34

In Table 2, two static threshold schemes, STATIC algorithm and ESM, were selected experimentally as a comparison to verify the effectiveness of the TSAS algorithm. In terms of the average false alarm rate, the TSAS algorithm reduced the false alarm rate by 31.35% and 58.90% compared to the ESM and STATIC algorithms. Considering that the fog edge network had the characteristics of narrow bandwidth and weak connection, the study further analyzed the effect of bandwidth fluctuation of the fog edge network link on the false positive rate and MAE, and the experimental results are shown in Figure 7.



(a) Influence of bandwidth fluctuation misjudgment rate

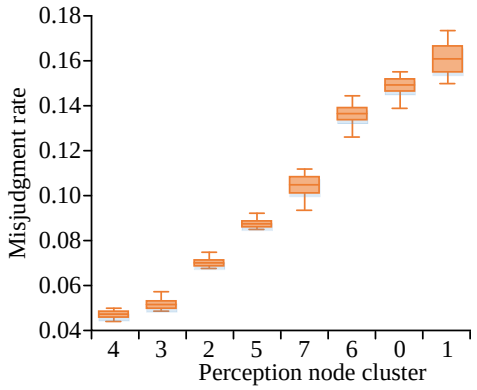


(b) Impact of bandwidth fluctuation on MAE

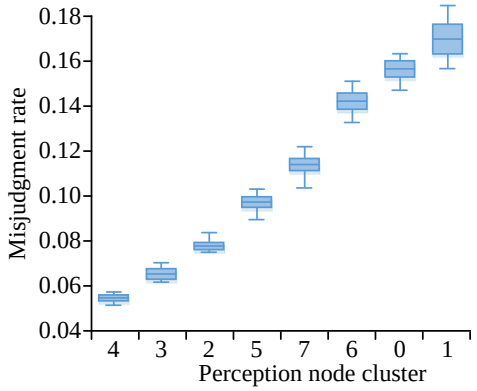
Figure 7: The impact of bandwidth fluctuation of fog edge network link on the error rate and MAE

As shown in Fig. 7(a) and Fig. 7(b), with the increase of the fluctuation variance of the link bandwidth, the false positive rate and MAE of the synchronous waiting time both increased. The comparison of the three algorithms showed that the false alarm rate and MAE of the TSAS algorithm were significantly lower than those of the STATIC algorithm and ESM. At the wave variance of 100, the false positive rate and MAE of TSAS algorithm were 30.5% and 31*e^{-5}, indicating that making full use of the spatial association information of the sensing terminal could effectively reduce the false alarm rate in the cooperative prediction mechanism. The results of further analysis of the influence of the distance between the terminal and the fog edge device on the false positive rate

based on the STAS algorithm and the ESM algorithm are shown in Figure 8.



(a) Error rate statistics of different sensing clusters based on STAS algorithm



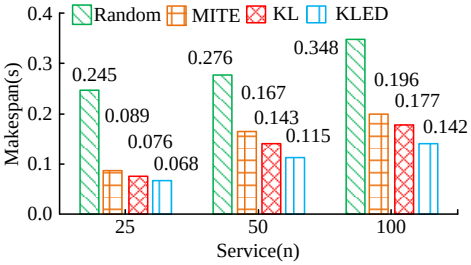
(b) Error rate statistics of different sensing clusters based on ESM algorithm

Figure 8: Experimental results of the influence of the distance between the terminal and the fog edge equipment on the error rate

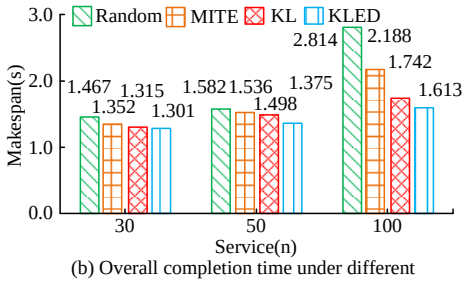
The fluctuation variance of each link was set to 10 in the experiment, and the horizontal coordinates were the different clusters ranked by the distance between the cluster center and the data aggregation node from smallest to largest. From Figure 8, both the mean and variance of the false positive rate increased with the increase of data transmission distance. When the transmission link was longer, the highest false positive rate of STAS algorithm and ESM algorithm was 16% and 17%, respectively. That is, the longer the transmission link is and the more complex the bandwidth variance is, the greater the impact on the false positive rate of the fog-end synchronization prediction mechanism.

4.3 Performance of collaborative information resource system for smart cities under fog computing

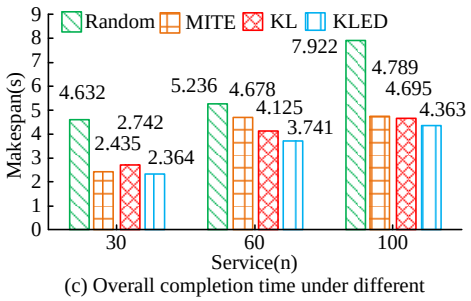
To verify the effectiveness of the KLED algorithm proposed in the study and the performance of the information resource collaboration system, the scientific workflow in Workflow Generator was selected as the service flow structure for the benchmark test. The workflow included four service structures, Montage, CyberShake, SIPHT, and LIGO Inspiral Analysis. Three comparative deployment strategies were used as baselines: Random Mapping, the Multi-Index Task-aware Edge scheduling algorithm (MITE), and the Key-Link algorithm (KL) implemented in this study. The KL algorithm followed a critical-chain-first placement strategy, assigning services along the shortest computing paths in the DAG while ignoring real-time communication or bandwidth dynamics. The experiments were repeated five times independently for each test, and the average value was taken as the final experimental result. The comparison of the overall service completion time of different algorithms under the four service structures is shown in Figure 9.



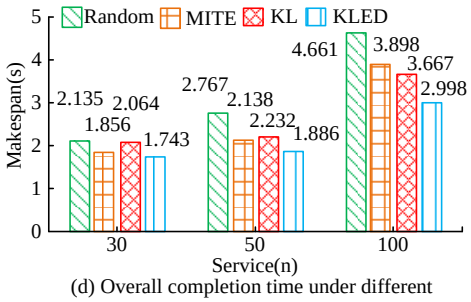
(a) Overall completion time under different scales of Montage service workflow



(b) Overall completion time under different scales of CyberShake service workflow



(c) Overall completion time under different scales of SIPHT service workflow



(d) Overall completion time under different scales of LIGO Inspiral Analysis service workflow

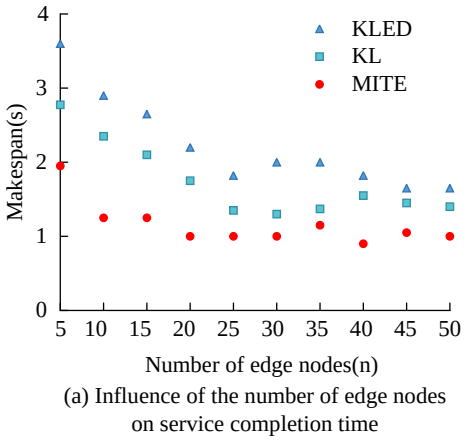
Figure 9: Comparison of overall service completion time of different algorithms under four service structures

Makespan refers to the total time required to complete all tasks within a workflow, measured from the start of the first task to the completion of the last one. It serves as a key metric to evaluate the overall scheduling efficiency and system responsiveness of service deployment strategies. In Figure 9, the makespan of the KLED algorithm proposed in the study substantially outperformed the Random deployment algorithm under four different service structures, and both outperformed the MITE algorithm and the KL algorithm. In the setting of service size of 100, the KLED algorithm reduced makespan by 19.77%, 7.41%, 7.07%, and 18.24%, respectively, compared to the KL algorithm for the four service structures of Montage, CyberShake, SIPHT, and LIGO Inspiral Analysis, i.e., the KLED algorithm outperformed the the other three algorithms. As the service size increased from 25 to 50 and 50 to 100, the makespan of KLED in the Montage structure increased by 40.87% and 19.01%, respectively. That is, the more complex the service process is, the operational performance requirements of the system increase as the communication relationships between subservices in the DAG increase. The study conducted impact proportional analysis experiments on the computation time, queuing waiting time and communication time that affect makespan, and the results are shown in Table 3.

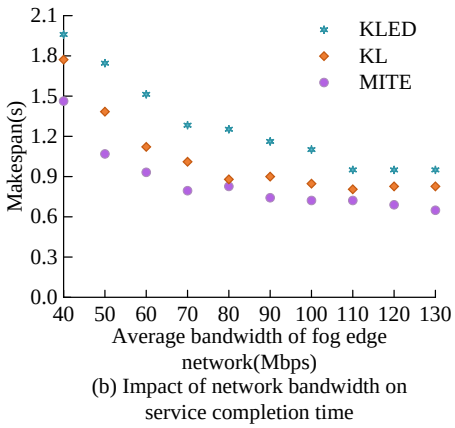
Table 3: Proportion of influence of calculation time, queue waiting time and communication time on makespan

Service size	Time type	Random	MITE	KL	KLED
	Waiting time	0.08	0.42	0.39	0.14
25	Communication time	0.73	0.38	0.35	0.43
	Calculated time	0.21	0.22	0.27	0.42
	Waiting time	0.12	0.47	0.41	0.22
50	Communication time	0.66	0.32	0.34	0.34
	Calculated time	0.24	0.23	0.25	0.43
	Waiting time	0.19	0.45	0.40	0.18
100	Communication time	0.63	0.38	0.40	0.40
	Calculated time	0.18	0.26	0.19	0.41
	Waiting time	0.75	0.78	0.77	0.73
1000	Communication time	0.20	0.14	0.15	0.11
	Calculated time	0.04	0.07	0.06	0.15

Table 3 presents the proportion of three-time components contributing to the overall makespan. Calculated time reflected the actual execution duration of tasks, communication time indicated the delay caused by data transmission between services, and waiting time represented the efficiency of resource scheduling. The results showed that when the service size was small, the Random algorithm exhibited the highest proportion of communication time, suggesting that its deployment strategy failed to minimize the transmission distance between services. MITE and KL showed a higher proportion of waiting time, indicating bottlenecks in task scheduling. In contrast, the KLED algorithm significantly increased the proportion of calculated time across different service sizes, demonstrating that system resources were more effectively utilized for execution after optimizing scheduling and transmission. As the service size grew to 1000, all four algorithms exhibited a sharp increase in waiting time proportion, primarily due to the limited parallelism, making queue delays the dominant factor in the overall makespan. The results of the KLED algorithm, MITE algorithm, and KL algorithm on the number of fog edge devices and network bandwidth on system performance for the Montage service with a service size of 100 are shown in Figure 10.



(a) Influence of the number of edge nodes on service completion time



(b) Impact of network bandwidth on service completion time

Figure 10: Effects of the number of fog edge devices and network bandwidth on system performance

In Fig. 10(a), the makespan of all three algorithms decreased gradually as the number of fog edge devices increased from 5 to 50, and the KLED algorithm outperformed the KL algorithm and the KL algorithm outperformed the MITE algorithm. The decrease of makespan was more obvious when the number of fog edge devices was increased from 5 to 15 than when the number of devices was increased from 15 to 50. Fig. 10(b) shows that as the available bandwidth increased from 10 Mbps to 150 Mbps, overall system performance improved accordingly. The most significant reduction in makespan was observed in the range of 40 to 100 Mbps. However, when the bandwidth exceeded 100 Mbps, the performance gains became marginal and the curve began to flatten, indicating that the communication bottleneck was largely eliminated. At this stage, computational and scheduling capacities became the primary limiting factors. Therefore, it is important to balance resource investment and performance improvement to avoid unnecessary resource overprovisioning.

To further validate the effectiveness of the proposed method, several representative approaches were selected as baseline methods, and their core mechanisms were reimplemented under the same experimental platform and dataset for quantitative comparison. The selected methods included: the sensing prediction mechanism based on throughput constraints proposed by J. Fei et al. [9], the data transmission optimization mechanism for vehicular networks proposed by W. Zhang et al. [11], and the multi-objective ant colony optimization–based service deployment strategy proposed by T. Huang et al. [15]. The evaluation metrics included false alarm rate, mean absolute error, and service completion time. The comparison results are presented in Table 4.

Table 4: Results of comparison with the existing methods

Method Source	False Alarm Rate (%)	MAE ($\times 10^{-5}$)	Service Completion Time (s)
Ref. [9] (Reproduced)	30.9	3.62	26.7
Ref. [11] (Reproduced)	27.4	3.20	26.5
Ref. [15] (Reproduced)	26.8	3.10	25.1
Proposed Method	12.7	2.60	21.8

In the table, the proposed method outperformed the three baseline methods in terms of false alarm rate, mean absolute error, and service completion time. Compared to Ref. [9] and Ref. [11], it demonstrated significantly better performance in prediction accuracy and synchronization. Compared to Ref. [15], it also achieved shorter completion time, indicating superior overall efficiency in both data transmission and task scheduling.

5 Discussion

To enhance the real-time processing of sensing data and improve service deployment efficiency in smart city scenarios, this study constructed a fog computing-based collaborative information resource system. It integrated sensing data, computational resources, and service dependency structures, and achieved intelligent scheduling and adaptive optimization at the edge layer through two core algorithms: TSAS and KLED. Specifically, the TSAS algorithm introduced a spatiotemporal correlation mechanism to dynamically adjust synchronization waiting times and prediction errors, thereby optimizing data compression and synchronization in high-frequency acquisition environments. The KLED algorithm combined K-means clustering with an improved scheduling strategy, taking into account resource heterogeneity and task dependencies among edge nodes, which effectively improved task parallelism and overall service execution efficiency.

Experimental results demonstrated that the TSAS algorithm reduced the false alarm rate to 18.8% and maintained the mean absolute error at 2.6×10^{-5}, thereby reducing unnecessary communication overhead while ensuring prediction accuracy. Meanwhile, the KLED algorithm achieved reductions in makespan of 18.24% and 7.41% compared to traditional KL and MITE deployment strategies across multiple workflow structures, and showed better scheduling stability and resource utilization under high-concurrency conditions.

Building on the above experimental results, the proposed TSAS and KLED algorithms also demonstrated clear advantages in terms of computational efficiency and deployment feasibility. The TSAS synchronization mechanism was based on an autoregressive model combined with exponential smoothing, involving only low-order matrix operations and sliding window updates. This resulted in significantly lower computational complexity compared to traditional deep learning-based prediction methods, making it more suitable for resource-constrained edge nodes. The KLED deployment strategy required maintaining only local service dependencies and scheduling states, with low memory overhead and structural stability, enabling efficient execution in heterogeneous devices and bandwidth-limited environments. The overall system adopted a modular design that facilitated integration into existing smart city edge infrastructures and supports good scalability. Therefore, TSAS and KLED not only outperformed traditional methods in terms of prediction accuracy and service latency, but also offered comprehensive advantages in computational efficiency and practical deployment adaptability.

While the experiments in this study focused on moderate-scale scenarios, the proposed system was

designed to support scalability in real-world deployments. TSAS executed independently on each edge device, requiring no global coordination, which enabled linear scalability as the number of devices increased. KLED, although involving clustering across multiple nodes, could be extended using hierarchical or region-based clustering frameworks to accommodate large-scale smart city environments with thousands of sensing points. Moreover, the modular architecture of both algorithms allowed parallel deployment and distributed processing, reducing the risk of performance bottlenecks. These design features made the system adaptable to future urban-scale deployments. Although security was not the primary focus of this study, data transmission was assumed to be protected by standard encryption, and basic access control could be applied to prevent unauthorized synchronization. The modular design of TSAS and KLED also allowed future integration of security mechanisms such as authentication and anomaly detection.

6 Conclusion

The convergence and development of scientific technologies such as artificial intelligence, the Internet of Things, and big data provide the basis for developing many intelligent analytical services, such as smart cities and intelligent transportation. The study combined fog computing, proposed a fog-end collaborative prediction mechanism to compress the amount of data transmission with high information redundancy in the edge network, and designed a KLED algorithm to reduce the overall computing time of the service and the corresponding delay of the service. The experimental results showed that the autoregressive model's prediction mechanism could effectively fit the changes of the temporal sensing data, and the TSAS algorithm could reduce the misclassification rate by about 31.35% and 58.90% compared with the STATIC algorithm and the ESM algorithm. The experimental performance results of the KLED algorithm showed that the highest waiting time sharing rate reached 73% when the service scale was 1000, and reduced the total calculation time of the service by approximately 13.12%, which effectively improved the service performance of the information. The service performance of the resource collaboration system was effectively improved. It is worth noting that the current experiments were conducted using a real-world dataset collected from 54 sensing devices in a specific urban scenario to verify the feasibility and effectiveness of the proposed approach. Although multiple datasets were not tested in this study, the core design of the TSAS and KLED algorithms was modular and adaptable to other fog computing environments. Future work will focus on validating the method across diverse datasets and deployment settings.

References

[1] A. Camero, and E. Alba, “Smart city and information technology: a review,” Cities, vol. 93, pp. 84-94, 2019. https://doi.org/10.1016/j.cities.2019.04.014

[2] A. Kirimtat, O. Krejcar, A. Kertesz, and M. F. Tasgetiren, “Future trends and current state of smart city concepts: a survey,” IEEE Access, vol. 8, pp. 86448-86467, 2020. https://doi.org/10.1109/ACCESS.2020.2992441

[3] Y. Shen, "Integration of IoT and Digital Twin for Intelligent Management of Urban Underground Pipe Galleries in Smart Cities," Informatica, vol. 49, no. 15, 2025. https://doi.org/10.31449/inf.v49i15.7903

[4] B. P. L. Lau, S. H. Marakkalage, Y. Zhou, N. U. I. Hassan, C. Yuen, M. Zhang, and U. Tan, “A survey of data fusion in smart city applications,” Information Fusion, vol. 52, pp. 357-374, 2019. https://doi.org/10.1016/j.inffus.2019.05.004

[5] M. Cetin, “Using GIS analysis to assess urban green space in terms of accessibility: case study in Kutahya,” International Journal of Sustainable Development & World Ecology, vol. 22, no. 5, pp. 1-5, 2015. https://doi.org/10.1080/13504509.2015.1061066

[6] E. Kaya, M. Agca, F. Adiguzel, and M. Cetin, “Spatial data analysis with R programming for environment,” Taylor & Francis, vol. 25, no. 6, pp. 1521-1530, 2019. https://doi.org/10.1080/10807039.2018.1470896

[7] M. Cetin, T. Aksoy, S. N. Cabuk, M. A. S. Kurkcuoglu, and A. Cabuk, “Employing remote sensing technique to monitor the influence of newly established universities in creating an urban development process on the respective cities,” Land Use Policy, vol. 109, pp. 109, 2021. https://doi.org/10.1016/j.landusepol.2021.105705

[8] J. Zhou, P. Wang, and L. Xie, “Research on resource allocation optimization of smart city based on big data,” IEEE Access, vol. 8, pp. 158852-158861, 2020. https://doi.org/10.1109/ACCESS.2020.3017765

[9] J. Fei, and X. Ma, “Fog computing perception mechanism based on throughput rate constraint in intelligent Internet of Things,” Personal and Ubiquitous Computing, vol. 23, no. 3, pp. 563-571, 2019. https://doi.org/10.1007/s00779-019-01200-9

[10] A. Ali, Y. Zhu, and M. Zakarya, “A data aggregation-based approach to exploit dynamic spatio-temporal correlations for citywide crowd flows prediction in fog computing,” Multimedia Tools and Applications, vol. 80, no. 20, pp. 31401-31433, 2021. https://doi.org/10.1007/s11042-020-10486-4

[11] W. Zhang, and G. Li, “An efficient and secure data transmission mechanism for Internet of vehicles considering privacy protection in fog computing environment,” IEEE Access, vol. 8, pp. 64461-64474, 2020. https://doi.org/10.1109/ACCESS.2020.2983994

[12] H. Qiao, T. Wang, and P. Wang, “A tool wear monitoring and prediction system based on multiscale deep learning models and fog computing,”

The International Journal of Advanced Manufacturing Technology, vol. 108, no. 7, pp. 2367-2384, 2020. https://doi.org/10.1007/s00170-020-05548-8

[13] J. Yang, J. Wen, Y. Wang, B. Jiang, H. Wang, and H. Song, "Fog-based marine environmental information monitoring toward ocean of things," IEEE Internet of Things Journal, vol. 7, no. 5, pp. 4238-4247, 2019. https://doi.org/10.1109/JIOT.2019.2946269

[14] C. Núñez-Gómez, B. Caminero, and C. Carrión, "HIDRA: a distributed blockchain-based architecture for Fog/Edge computing environments," IEEE Access, vol. 9, pp. 75231-75251, 2021. https://doi.org/10.1109/ACCESS.2021.3082197

[15] T. Huang, W. Lin, C. Xiong, R. Pan, and J. Huang, "An ant colony optimization-based multiobjective service replicas placement strategy for fog computing," IEEE Transactions on Cybernetics, vol. 51, no. 11, pp. 5595-5608, 2020. https://doi.org/10.1109/TCYB.2020.2989309

[16] M. Ayoubi, M. Ramezanpour, and R. Khorsand, "An autonomous IoT service placement methodology in fog computing," Software: Practice and Experience, vol. 51, no. 5, pp. 1097-1120, 2021. https://doi.org/10.1002/spe.2939

[17] Z. Ning, J. Huang, and X. Wang, "Vehicular fog computing: enabling real-time traffic management for smart cities," IEEE Wireless Communications, vol. 26, no. 1, pp. 87-93, 2019. https://doi.org/10.1109/MWC.2019.1700441

[18] I. Afzal, N. ul Amin, Z. Ahmad, and A. Algarni, "A latency-aware and fault-tolerant framework for resource scheduling and data management in fog-enabled smart city transportation systems," Computers, Materials & Continua, vol. 82, no. 1, pp. 1377 – 1399, 2025. https://doi.org/10.32604/cmc.2024.057755

[19] K. D. Singh, P. D. Singh, R. Verma, and S. Maurya, "Optimizing urban resource management through cloud and fog computing in smart cities," AIP Conference Proceedings, vol. 3121, no. 1, 030002, 2024. https://doi.org/10.1063/5.0222009

[20] T. P. da Silva, T. V. Batista, F. Lopes, and A. Rocha Neto, "Fog computing platforms for smart city applications: a survey," ACM Transactions on Internet Technology, vol. 22, no. 4, Article 1, 2022. https://doi.org/10.1145/3488585

[21] S. Rajagopal, P. K. Tripathi, M. Deshmukh, S. Choudari, and A. Kumar, "Edge computing–smart cities: optimizing data processing & resource management in urban environments," Journal of Information Systems Engineering and Management, vol. 10, no. 5s, 2025. https://doi.org/10.52783/jisem.v10i5s.667

[22] A. Elsayed, K. Mohamed, and H. Harb, "Enhanced traffic congestion management with fog computing: a simulation-based investigation using iFog-Simulator," arXiv preprint, arXiv:2311.01181, 2023. https://arxiv.org/abs/2311.01181

[23] M. Fahimullah, S. Ahvar, and M. Trocan, "A review of resource management in fog computing: machine learning perspective," arXiv preprint, arXiv:2209.03066, 2022. https://arxiv.org/abs/2209.03066

[24] Z. Wang, M. Goudarzi, J. Aryal, and R. Buyya, "Container orchestration in edge and fog computing environments for real-time IoT applications," arXiv preprint, arXiv:2203.05161, 2022. https://arxiv.org/abs/2203.05161

[25] S. Iftikhar, S. S. Gill, C. Song, M. Xu, M. S. Aslanpour, and A. N. Toosi, "AI-based fog and edge computing: a systematic review, taxonomy and future directions," arXiv preprint, arXiv:2212.04645, 2022. https://arxiv.org/abs/2212.04645

[26] A. M. Alsmadi, et al., "Fog computing scheduling algorithm for smart city," International Journal of Electrical and Computer Engineering, vol. 11, no. 3, pp. 2219 – 2228, 2021. https://doi.org/10.11591/ijece.v11i3.pp2219-2228

[27] B. Tang, Z. Chen, G. Hefferman, S. Pei, W. Tao, H. He, and Q. Yang, "Incorporating intelligence in fog computing for big data analysis in smart cities," IEEE Transactions on Industrial Informatics, vol. 13, no. 5, pp. 2140 – 2150, 2017. https://doi.org/10.1109/TII.2017.2679740

[28] P. S. Rathore, R. Kharel, D. Choi, and J. H. Park, "Energy-efficient cluster head selection through relay approach for WSN," The Journal of Supercomputing, vol. 77, pp. 7649 – 7675, 2021. https://doi.org/10.1007/s11227-021-03835-0

[29] A. Hazra, P. Rana, M. Adhikari, and T. Amgoth, "Fog computing for next-generation Internet of Things: fundamentals, applications and research challenges," Journal of Cloud Computing, vol. 11, no. 1, Article 17, 2022. https://doi.org/10.1186/s13677-022-00301-7

[30] C. Guerrero, I. Lera, and C. Juiz, "A lightweight decentralized service placement policy for performance optimization in fog computing," Journal of Ambient Intelligence and Humanized Computing, vol. 10, no. 6, pp. 2435-2452, 2019. https://doi.org/10.1007/s12652-018-0914-0