Zheng *et al. J Wireless Com Network* (2024) 2024:3
https://doi.org/10.1186/s13638-023-02331-7

EURASIP Journal on Wireless Communications and Networking

RESEARCH Open Access

Tconns: a novel time-varying context-aware offloading strategy for mobile edge computing

Meiguang Zheng[1*], Jie Li[1], Yu Hu[1], Hui Xiao[1] and Zhigang Hu[1]

*Correspondence:
zhengmeiguang@csu.edu.cn

[1] School of Computer Science and Engineering, Central South University, Chang Sha, China

Abstract

Mobility is a fundamental feature of mobile edge computing. Due to the mobility of users, the contextual attributes of cloudlets such as server resources and network state will dynamically change with time during offloading, showing time-varying and fuzzy characteristics. To this end, how to make efficient offloading decision to provide low-latency, low-power and highly reliable services in MEC has become a critical issue. In this paper, we propose a time-varying context-aware cloudlet decision algorithm based on neutrosophic set, TConNS (The Code of TConNS is available at https://github.com/zhengLabs/NSO). Firstly, we establish a representation model of the multi-dimensional time-varying context of candidate cloudlets, including the mobile residence time. Secondly, we adopt the backward generator of cloud model theory to transform the contextual raw data into a single-valued neutrosophic set with the expression ability for fuzzy information. Finally, we use a series of appropriate operations under the own unique computing system of neutrosophic set to obtain the best cloudlet. Extensive experiments show that TConNS reduces the average response time by about 49% and the average energy consumption by about 46%, and also reduces the number of task failures.

Keywords: Edge offloading, Context aware, Neutrosophic set, Time-varying data, Mobility, Cloud model

1 Introduction

Mobile devices are facing the limitations in terms of battery power and storage space. It is challenging for them to meet the demand of low latency, low power consumption and high reliability when executing new applications such as VR, image processing and unmanned driving [1]. Mobile Edge Computing (MEC) is a new computing paradigm that deploys computing and storage resources at the network edge (e.g., cloudlets, micro data centers, or fog nodes) closer to mobile devices, thereby providing users with high-speed and low-latency services [2].

Computational offloading and mobility management are two key technologies in MEC. Computational offloading [3] aims to offload a part or all computing tasks of mobile devices to edge servers. Mobility management in MEC [4] refers to selecting appropriate edge nodes within the dense and complex network to provide services according to the users' mobility trace.

Many related researches have explored optimal offloading decisions [5–7] and mobile-aware offloading [8, 9], aiming to minimize energy consumption while satisfying the delay constraint. However, these studies also have some limitations as follows: (1) The variation of contexts with environmental parameters is neglected, which will have a great impact on the offloading decision adaptation; (2) All tasks will be offloaded by default. However, some tasks may be less power-wasting to process on the local device; (3) In mobile-aware scheme, offloading decisions are usually made purely based on mobility. In fact, mobility is just one of the important factors influencing the offloading decisions.

Mobility is the basic feature of MEC [10], and the core characteristic is the change of location or movement pattern. Wang et al. [10] showed that the user movement process will upload tasks and receive results through different small cell base stations (SBS). The number of users accessed by each SBSs is different, and the load involving communication and computing in the mobile edge network will change, that is, the movement of users results in a time-varying workload. Due to the mobility, The terminal devices in MEC are in different areas at different moments. The network condition and the available resources of the edge server (such as cloudlets) are dynamically updated with time [10, 11]. The values of contextual attributes that affect the offloading decision fluctuate, have time-varying characteristics [12, 13]. The fixed context offloading strategy do not consider the mobility characteristics, which may lead to the invalidation for task offloading in MEC [12].

In this paper, we make the time-varying context-aware optimal edge offloading decision based on neutrosophic set (NS). Our main contributions are summarized as follows:

(1) We establish a contextual offloading decision framework in a three-layer cloud-edge hybrid MEC environment. Furthermore, we establish a single-value NS model with the ability to express fuzzy information for four key time-varying contextual attributes.
(2) We propose a time-varying context-aware cloudlet decision-making scheme based on neutrosophic sets, and use a series of NS unique operations including neutrosophic entropy, aggregation operator and NS score function to calculate the optimal cloudlet
(3) Experiments show that compared with other methods, the proposed scheme performs better in terms of the number of task failures, response time and energy consumption.

The remainder of this paper is organized as follows. Section 2 reviews related work. Section 3 provides an overview of the framework and a description of the problem. Section 4 proposes the transformation from time-varying context sequence data to the contextual NS representation. Section 5 introduces the proposed cloudlet decision algorithm. Section 6 presents the experimental details and results. Section 7 concludes the paper and highlights future directions of study.

2 Related works

In order to make offloading decisions that maximize benefits, it is essential to consider contextual information. Studies have been conducted to make service selection or offloading decision by considering different contextual factors, such as user location [11],

wireless medium [14], application type [15], and network bandwidth [16], etc. These schemes adopted such as supervised learning [11] or a multi-criteria decision-making [14] to improve cloud computing service performance or optimize the energy consumption of mobile devices. Given a specific context, these schemes' decisions may be inaccurate and inefficient. In response to this problem, Junior et al. [17] considered six factors including network throughput, application data size, CPU utilization of mobile phone, etc., and used machine learning classification algorithms JRIP and J48 to achieve high-precision context-aware offloading.

Chen et al. [18] conducted an evaluation under the T-Mobile LTE network. It was shown that when watching streaming video in a mobile vehicle with weak wireless signal, maintaining a high bitrate is not helpful for improving the user experience. Meanwhile, reducing the resolution at this time can greatly save energy of the device. As you can see, mobility does affect the offloading context.

To this end, some studies (such as [2] and [8]) model the distribution of mobile users in the Cloudlet as a Poisson point process, which then could be solved using a Markov Decision Process (MDP). The difference is that Wang et al. [2] considered the sojourn time of the mobile device in the current service area, while Zhang et al. [8] focused on the impact of user movement on the connectivity between the user and the cloudlet.

Solutions using MDP assume that the user's movement patterns are known in advance which is difficult to obtain in fact. Zhang et al. [4] proposed a deep Q-network (DQN)-based task migration technique for MEC systems, which learns optimal task offloading and migration strategies from previous experience, without having to obtain the user's moving pattern information in advance. Mukherjee et al. [19] proposed a mobile-aware task delegation model and a code offloading model based on virtual machine migration. Yu et al. [9] proposed a dynamic mobility-aware partial offloading scheme based on short-term movement prediction, which optimize the offloading ratio and offloading path during the movement of users.

It can be seen that the existing mobile-aware edge offloading research focused on the analysis and processing of mobile patterns. In fact, the user's movement is accompanied by dynamic hidden features that change over time. These hidden features lead to uncertainty about the network enabled by MEC [20]. To ensure the effectiveness of offloading in MEC, it is necessary to consider the time-varying characteristics in addition to the mobility characteristics [12].

The current research on the time-varying characteristics in MEC mainly consider two types, namely, time-varying network and time-varying load variation. Both [12] and [21] study the time-varying characteristics of the channel in-vehicle edge computing. A time-varying channel will result in time-varying spectral efficiency (SE). If the time-varying delay is not considered, the offloading strategy cannot guarantee the task delay requirement. In [12], the time-varying SE is calculated using the average SE across the region during the task transit time. [21] also adopted the mean value scheme for time-varying features. The MobiEdge system [13] considered the time-varying characteristics of the service request workload in a frame for edge service placement and transforms the maximization problem under constraints into a linear programming problem.

It can be seen that the existing work has limited consideration of the changes context of mobile edge environment over time and does not simultaneously integrate mobility

and other contexts to make comprehensive offloading decisions. The above reasons prompt us to propose a time-varying context-aware edge offloading strategy.

3 Three-tier offloading decision framework and problems

3.1 System framework

In this paper, we consider a three-tier cloud-edge hybrid computation offloading environment, consisting of a central cloud, some cloudlets and some mobile devices. Computational tasks can be executed at these three locations. Among them, the central cloud can provide stable and powerful computing ability and massive storage, but the communication latency is high. Within the user's mobile range, there are also p cloudlets, which deploy servers with computing and network resources. The cloudlets have computing ability superior to mobile devices, and can provide low-latency services compared to the central cloud. At this time, within the delay threshold T_{timer}, the optimal cloudlet may be selected from the nearby p cloudlets to perform the task according to the context attributes.

The time-varying context-aware edge offloading framework proposed in this paper is shown in Fig. 1. The framework has three levels: (1) Obtaining historical contextual time-varying data. These data, including CPU utilization, load, and network latency are obtained through related APIs. The mobile attribute is characterized by the expected residence time within the cloudlet. (2) A novel presentation model of time-varying context. The time-varying data of contextual attributes are dynamic and imprecise. We use a neutrosophic set (NS) model to describe such uncertain fuzzy

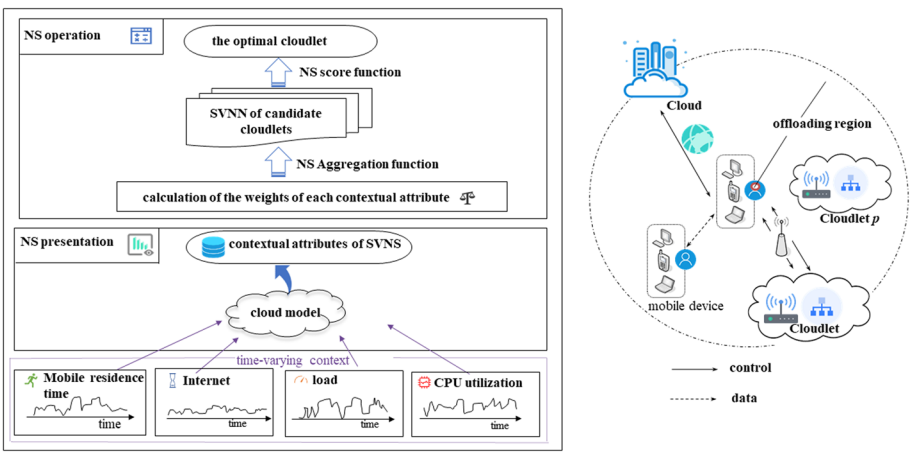

Fig. 1 The framework of time-varying context-aware offloading based on neutrosophic set. We consider a three-tier cloud-edge hybrid computation offloading environment (which is depicted in the right part), consisting of a central cloud, some cloudlets and some mobile devices. Computational tasks can be executed at these three locations. The framework, from bottom to top, has three levels: (1) Obtaining historical contextual time-varying data. These contextual data includes four types, include CPU utilization, load, network latency and the expected residence time within the cloudlet. (2) A novel presentation model of time-varying context. The time-varying data of contextual attributes are dynamic and imprecise. We use a neutrosophic set (NS) model to describe such uncertain fuzzy information and generate a single-valued NS for each contextual attribute of cloudlets. (3) The best cloudlet selection through a series operations of NS. That is, first, we calculate the weight of each context attribute. we calculate the weight of each context attribute. Then, we use the single valued NS weighted average aggregation operator to aggregate the SVNS of cloudlets into the single-valued neutrosophic number of cloudlets. Finally, we select the best cloudlet using the NS score function

information, and generate a single-valued NS for each contextual attribute of cloudlets. (3) The best cloudlet selection through a series operations of NS. That is, first, we calculate the weight of each context attribute. Then, we use the single-valued NS weighted average aggregation operator to aggregate the SVNS of cloudlets into the single-valued neutrosophic number of cloudlets. Finally, we select the best cloudlet using a combination of the NS score function and the cloudlet usage probability predicted by the mobile periodicity.

In the three-layer cloud-edge environment, there are n tasks to be executed. The execution cost consists of two parts: the task response time and the energy consumption. The execution costs of computing tasks in the local mobile device, the cloudlet and the central cloud are denoted as $C_{mob}, C_{clt}, C_{cld}$, respectively.

Accordingly, if $C_{mob} \leq C_{clt}$, it is less costly to process the task locally on the mobile device itself, and otherwise it is more appropriate to offload the task to the nearest cloudlet. When no suitable cloudlet can be obtained within T_{timer}, the task will be transferred to the central cloud and be served. The tasks in the system are offloaded to three locations, and the total cost of completing this set of tasks is formulated as follows:

$$C_{sum} = \sum_{i=0}^{x} C_{mob_i} + \sum_{j=0}^{y} C_{clt_j} + \sum_{k=0}^{z} C_{cld_k} \tag{1}$$

where x, y and z denote the number of tasks executed in local mobile devices, cloudlets and the central cloud, respectively. Here, we have $x + y + z = n$.

3.2 Three layers cloud-edge hybrid offloading cost

The execution cost consists task response time T and energy consumption E as follows:

$$C_{\text{site}} = \alpha \cdot T_{\text{site}} + \beta \cdot E_{\text{site}} \tag{2}$$

where $site \in \{mob, clt, cld\}$, α and β are the weighting factors for response time and energy consumption, determined according to user preferences. The calculations of T and E for task execution at the three locations are detailed below, respectively.

(1) Local execution

If the task is executed on a mobile device, the response time and energy consumption of the task are calculated as follows:

$$T_{mob} = I/S_{mob} \tag{3}$$

$$E_{mob} = P_{mob} \cdot (I/S_{mob}) \tag{4}$$

where I refers to the task size which is described by the number of instructions, S_{mob} is the number of instructions executed per unit time of the mobile device, and P_{mob} is the power consumption per unit time while executing the task.

(2) Task offloading to the cloudlet

If a task request is responded by the nearest cloudlet, the response time includes propagation time, communication time, queuing time and processing time. The corresponding calculation is shown as follows:

$$T_{clt} = \left(D_{clt}/S_p\right) + [(D_u/B_u) + (D_d/B_d)] + Q_{clt} + (I/S_{clt}) \tag{5}$$

where D_{clt} is the distance between the device and the cloudlet, S_p is the propagation speed. D_u, D_d, B_u, B_d are the amount of uplink data, the amount of downlink data, uplink data transmission rate, downlink data transmission rate, respectively, Q_{clt} denotes the queue waiting time, S_{clt} denotes the number of instructions executed by cloudlet per unit time.

The total energy consumption of the task execution on the cloudlet includes the energy consumption of the mobile device when sending and receiving data and waiting for the results. It is calculated as follows:

$$E_{clt} = P_{moi} \cdot \left[\left(D_{clt}/S_p\right) + (I/S_{clt}) + Q_{clt}\right] + P_s \cdot (D_u/B_u) + P_r \cdot (D_d/B_d) \tag{6}$$

where P_{moi} is the power consumed per unit time when the mobile device is waiting for a result, P_s is the power consumed by mobile device for sending data per unit time, P_r is the power consumed by mobile device for receiving data per unit time.

(3) Task offloading to the central cloud

The central cloud has powerful computing ability and storage capabilities, and thus tasks can be executed without queuing. The response time and energy consumption are calculated below.

$$T_{cld} = (D_{clt} + D_{cld})/S_p + [(D_u/B_u) + (D_d/B_d)] + [I/S_{cld}] + T_{\text{timer}} \tag{7}$$

$$E_{cld} = P_{moi} \cdot \left[(I/S_{cld}) + (D_{clt} + D_{cld})/S_p + T_{\text{timer}}\right] + P_s \cdot (D_u/B_u) + P_r \cdot (D_d/B_d) \tag{8}$$

where D_{cld} represents the distance between the nearest cloudlet and the central cloud, S_{cld} represents the number of instructions executed by the central cloud per unit time.

3.3 Time-varying context of offloading

In this paper, contexts are defined as the relevant attributes that affect the task offloading process and results. Four time-varying contextual factors are considered for offloading decision, namely, cloudlet load, CPU utilization, network conditions and mobile residence time.

The former three factors are common contextual factors which are easy to measure, collect and test [22]. (1) The load refers to the average number of tasks that are using CPU or waiting CPU over a period of time. The higher the load, the higher the queuing delay and service loss probability. (2) CPU utilization is the percentage of CPU occupied by the program in real time during its operation. The lower the current CPU utilization, the more computationally intensive tasks can be executed. (3) Network conditions determine the quality of communication between the mobile device and the server throughout the offloading process.

(4)The mobile residence time. User's movement may result in disconnection from the old cloudlet and reconnection to the new cloudlet, bringing additional task migration overhead. Different from the former three contexts, mobility is an abstract concept. In this paper, the mobile residence time of the device is used to represent the user's expected residence time within the cloudlet service range. The mobile residence time is one of the most important factors affecting edge server selection and ensuring service continuity.

The above four contexts are dynamically updated over time, all have time-varying characteristics. Thus, making the offload decision cannot only consider the state of the context at the current moment but should analyze it from a recent period of time.

3.4 Mobile residence time

The longer a user stays within the service scope of a cloudlet, the more likely the task will be completed at the current cloudlet without additional overhead resulted from task migration [23]. The variation of mobile residence time is shown in Fig. 2. Suppose the user moving along the direction α_0 at the moment t_0, changes the direction of movement to α_1 at the moment t_1, and the user moves along α_2 again by t_2.

Taking cloudlet2 as an example, we estimate the mobile residence time of users within cloudlet2's service range. The cloudlet position is (a, b), R denotes the service scope of the cloudlet. The user current position is (x, y). The user's direction of movement α and the user moving speed V can be obtained via GPS. D is the straight line distance between the

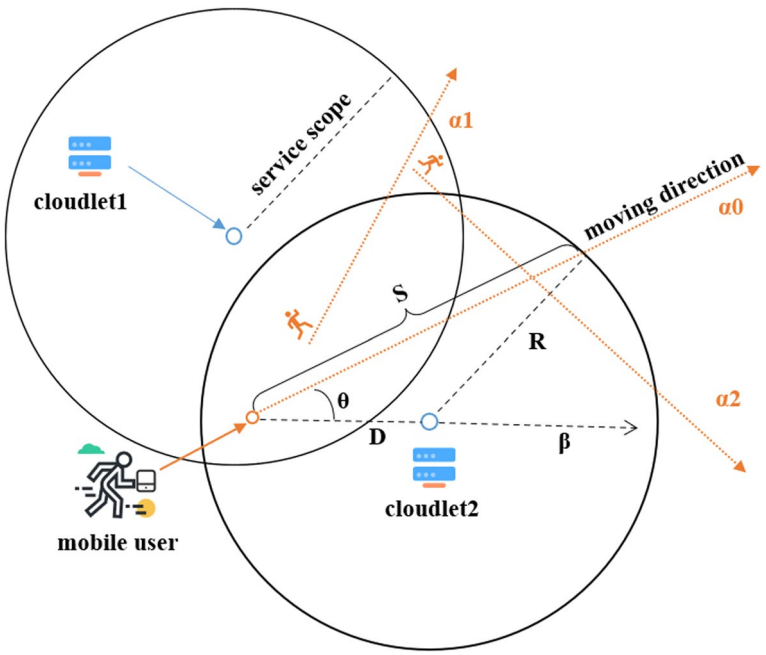

Fig. 2 Illustrate the measure of mobile residence time. Suppose the user moving along the direction α_0 at the moment t_0, changes the direction of movement to α_1 at the moment t_1, and the user moves along α_2 again by t_2. Taking cloudlet2 as an example, we estimate the mobile residence time of users within cloudlet2's service range. The cloudlet position is (a, b), R denotes the service scope of the cloudlet. The user current position is (x, y). The user's direction of movement α and the user moving speed V can be obtained via GPS. D is the straight line distance between the user's current position and the cloudlet

user's current position and the cloudlet. Then the residence time of the user in cloudlet2 at moment t_0 can be calculated by Eq. (9).

$$M = \frac{S}{v} = \frac{D\cos\theta + \sqrt{R^2 - D^2\sin^2\theta}}{v} \tag{9}$$

where S is the distance of the user needs to travel to leave the coverage of the cloudlet along the moving direction. $\theta = \arccos\frac{\alpha\cdot\beta}{|\alpha|\times|\beta|}$, β is the direction vector from the user's current position to the cloudlet. We have $\beta = (a-x, b-y)$. Instead of calculate the residence time of cloudlet through a single straight-line distance and velocity, we describe the time-varying characteristics of the estimated residence time, record the estimated residence time within q moments. There are already some software that obtains the relevant interfaces to achieve accurate step counting, speed acquisition, etc. Examples include Google's Speedometer, a GPS Speed Tracker. By calculating the expected residence time of mobile users at q moments within p cloudlets respectively, we can obtain the time-varying data sequences, denoted as $M_i^j, i \in \{1,2,...,p\}, j \in \{1,2,...,q\}$.

4 Methods

4.1 Time-varying context model based on single-valued NS

4.1.1 Time-varying context data sequence of a cloudlet

Suppose there are p candidate cloudlets. The time-varying sequences of cloudlet load L, CPU utilization C and network conditions D in q consecutive moments are denoted as L_i^j, C_i^j, D_i^j, $i \in \{1,2,...,p\}$, $j \in \{1,2,...,q\}$. The load and CPU utilization can be viewed by top and vmstat commands respectively in Linux environment. The network conditions are measured by network latency. The time-varying mobility property is characterized by the expected user residence time within the cloudlet. By calculating the expected residence time of mobile users at q moments within p cloudlets respectively, we can obtain the time-varying data sequences, denoted as M_i^j, $i \in \{1,2,...,p\}, j \in \{1,2,...,q\}$. The time-varying contextual data of $cloudlet_i$ are collectively denoted as a quaternion vector $CON(i) = (L_i, C_i, D_i, M_i)$.

After normalizing the contextual original data, the four time-varying contextual data sequences of p candidate cloudlets at q moments can be expressed as matrix $CON(cloudlet_i^j)_{p\times 4}, i \in \{1,2,...,p\}, j \in \{1,2,...,q\}$, which is formulated as follows:

$$\begin{aligned} CON\left(cloudlet_i^j\right)_{p\times 4} &= \left(L_{(p\times q)}, C_{(p\times q)}, D_{(p\times q)}, M_{(p\times q)}\right) = \\ &\begin{bmatrix} \left(L_1^1, L_1^2, \ldots, L_1^q\right) & \left(C_1^1, C_1^2, \ldots, C_1^q\right) & \left(D_1^1, D_1^2, \ldots, D_1^q\right) & \left(M_1^1, M_1^2, \ldots, M_1^q\right) \\ \left(L_2^1, L_2^2, \ldots, L_2^q\right) & \left(C_2^1, C_2^2, \ldots, C_2^q\right) & \left(D_2^1, D_2^2, \ldots, D_2^q\right) & \left(M_2^1, M_2^2, \ldots, M_2^q\right) \\ \vdots & \vdots & \vdots & \vdots \\ \left(L_p^1, L_p^2, \ldots, L_p^q\right) & \left(C_p^1, C_p^2, \ldots, C_p^q\right) & \left(D_p^1, D_p^2, \ldots, D_p^q\right) & \left(M_p^1, M_p^2, \ldots, M_p^q\right) \end{bmatrix} \end{aligned} \tag{10}$$

The i-th row of the matrix $CON(cloudlet_i^j)$, $CON(i)$ corresponds to the context data sequence of the $cloudlet_i$, expressed as follows:

$$\begin{aligned} &CON(i) = (L(i), C(i), D(i), M(i)) \\ &= \left((L_i)_{(1*q)}, (C_i)_{(1*q)}, (D_i)_{(1*q)}, (M_i)_{(1*q)}\right) \end{aligned} \tag{11}$$

4.1.2 The SVNS representation of the time-varying context

Most of the existing work on analyzing the time-varying features of mobile edges adopt the average value based scheme, which is difficult to accurately characterize the inconsistent and uncertain fuzzy information. In this paper, we consider the characterization tool of fuzzy information, **single-valued neutrosophic set (SVNS)**. SVNS [24] is a branch of neutrosophic set (NS) theory [25], which expands the traditional fuzzy set, adopts three measures of truth-membership, indeterminacy-membership and falsity-membership to characterize fuzzy decision-making information. It has the ability to delicately and accurately describe the fuzzy nature of objective things. At the same time, the neutrosophic set has independent unique operations, which have been widely used in multi-attribute decision making [26], artificial intelligence [27] and service evaluation [28].

The value of the membership function of the SVNS is a real number. Given a domain X, a SVNS A on X includes truth-membership function $T_A(x)$, indeterminacy-membership function $I_A(x)$ and falsity-membership function $F_A(x)$, denoted as follow:

$$A = \{\langle X, T_A(x), I_A(x), F_A(x)\rangle \mid x \in X\}$$

where $T_A(x), I_A(x), F_A(x) \in [0,1]$, and $0 \leq T_A(x) + I_A(x) + F_A(x) \leq 3$.

In order to use SVNS to characterize the time-varying context, it is necessary to establish three membership functions for each time-varying context factor of $cloudlet_i$, that is, to obtain the three membership values for each context factor of $cloudlet_i$. Converting the contextual data sequence expressed in Eq. (11) to a SVNS, it can be expressed as follows:

$$\forall con \in \{L, C, D, M\}, CON(i) \Rightarrow \{T_i^{con}(x),\ I_i^{con}(x),\ F_i^{con}(x)\} \tag{12}$$

4.1.3 The context SVNS generation using backward cloud model

In order to establish the three membership functions of the time-varying context, **cloud model** [29](CM) is used to complete the transformation expressed in Eq. (5). The cloud model is a cognitive model that implements the duplex transformation of qualitative concepts and quantitative data. The contextual data sequence have been acquired in section 4.1, so the current problem is one that goes from quantitative (data sequence) to qualitative (degree of membership).

The cloud model has three numerical features: expectation $\hat{E}x$, entropy $\hat{E}n$, and super-entropy $\hat{H}e$. $\hat{E}x$ is the expectation of the spatial distribution of cloud drops in the theoretical domain, which represents the basic certainty of the qualitative concept; $\hat{E}n$ represents the measure of uncertainty of the qualitative concept, which reflects the range of values that can be accepted by this concept in the theoretical domain. $\hat{H}e$ is the entropy of the entropy $\hat{E}n$, which is the uncertainty of the entropy and expresses the deviation of the cloud model. Therefore, $\hat{E}x$ can be taken as the truth-membership T_A of SVNS, $\hat{E}n$ as the indeterminacy-membership I_A, and $\hat{H}e$ as the falsity-membership F_A, that is,

$$T_A(x) = \hat{E}x(x) \tag{13}$$

$$I_A(x) = \hat{E}n(x) \tag{14}$$

$$F_A(x) = \hat{He}(x) \tag{15}$$

In this paper, hybrid cloud and cloud models both refer to the word 'cloud'. To distinguish them, hybrid cloud is used to refer to nodes or clusters in distributed computing. It is a computational paradigm. However, cloud, cloud model, cloud droplet and other related words are all terms in cloud theory, which involves uncertain concepts in the artificial intelligence field.

In cloud model, a cloud consists of cloud drops. A cloud drop is a realization of a qualitative concept, and a certain number of cloud drops can express a cloud. Here, the value of one-dimensional time-varying context of $cloudlet_i$ at moment j can be regarded as a cloud drop. Its sequence of values describing the dimensional context at q moments can characterize the cloud model which is denoted as $CM^{con}(i)$. That is, $CM^{con}(i) = (con_i^1, con_i^2, ..., con_i^q)$. For $cloudlet_i$, use the membership calculation scheme shown in Eq. (13) - Eq. (15). That is, the backward cloud generator is selected to establish a context cloud model for each dimension as follows:

$$T_i^{con}(x) = \hat{Ex}_i^{con} = \bar{X} = \frac{1}{q}\sum_{j=1}^{q} con_i^j \tag{13*}$$

$$I_i^{con}(x) = \hat{En}_i^{con} = \sqrt{\frac{\pi}{2}} \times \frac{1}{q}\sum_{j=1}^{q}\left|con_i^j - \hat{Ex}\right| \tag{14*}$$

$$F_i^{con}(x) = \widehat{He}_i^{con} = \sqrt{S^2 - \hat{En}^2}, S^2 = \frac{1}{q-1}\sum_{k=1}^{q}\left(con_i^j - \bar{X}\right)^2 \tag{15*}$$

where $con \in \{L, C, D, M\}$, con_i^j denotes the time-varying context sequence of a certain context of $cloudlet_i$, $i \in \{1, 2, ..., p\}$, $j \in \{1, 2, ..., q\}$.

A element in the SVNS A is called **single-valued neutrosophic number (SVNN)**, which is expressed as $\{T_A, I_A, F_A\}$. For the $cloudlet_i$, the original time-varying context data sequences of each dimension generates three numeric features through the backward cloud generator, that is, three SVNNs are generated.

So far, the transformation from the time-varying context data sequence of the candidate cloudlets to the context SVNN has been completed, which can be expressed as follow:

$$\begin{aligned} &CON(i) = (con_i^1,\ con_i^2,\ ...,\ con_i^q) = CM_i^{con} \Rightarrow \\ &\{T_i^{con}(x), I_i^{con}(x), F_i^{con}(x)\} = \{\hat{Ex}_i^{con},\ \hat{En}_i^{con},\ \hat{He}_i^{con}\} \end{aligned} \tag{12*}$$

Therefore, the time-varying contextual numerical sequence matrix $CON(cloudlet_i^j)_{p\times 4}$ of the p candidate cloudlets is transformed into the following SVNS context matrix as follows:

$$SVNS\left(cloudlet_i^{con}\right)_{p\times 4} = \left\{T_i^{con}, I_i^{con}, F_i^{con}\right\}_{p\times 4} =$$
$$\begin{bmatrix} \left\{T_1^L, I_1^L, F_1^L\right\} & \left\{T_1^C, I_1^C, F_1^C\right\} & \left\{T_1^D, I_1^D, F_1^D\right\} & \left\{T_1^M, I_1^M, F_1^M\right\} \\ \left\{T_2^L, I_2^L, F_2^L\right\} & \left\{T_2^C, I_2^C, F_2^C\right\} & \left\{T_2^D, I_2^D, F_2^D\right\} & \left\{T_2^M, I_2^M, F_2^M\right\} \\ \vdots & \vdots & \vdots & \vdots \\ \left\{T_p^L, I_p^L, F_p^L\right\} & \left\{T_p^C, I_p^C, F_p^C\right\} & \left\{T_p^D, I_p^D, F_p^D\right\} & \left\{T_p^M, I_p^M, F_p^M\right\} \end{bmatrix}$$

where $i \in \{1, 2, ..., p\}, con \in \{L, C, D, M\}$. Each row vector of the matrix $SVNS(i)$ represents the four-dimensional context SVNNs of $cloudlet_i$, and the four column vectors respectively correspond to the SVNS of all candidate cloudlets for the four-dimensional context.

The method of transformation from the original context time-varying sequence to the context SVNS proposed in this paper is extensible. Under specific scenario, other customization decision-making factors can also be extended to take into account.

4.2 Time-varying context-aware cloudlet decision

After obtaining the SVNN of each time-varying contextual attribute of the candidate cloudlet, the optimal cloudlet offloading decision can be made. In this paper, the offloading problem is transformed into a multi-criteria decision-making (MCDM) problem. Here, the MCDM based on NS are represented by SVNN, which can characterize the ambiguity of decision information. For the SVNS context matrix $SVNS(cloudlet_i^{con})_{p\times 4}$, some unique appropriate operators under the NS computing system are selected to determine the optimal cloudlet.

4.2.1 Candidate cloudlet assessment by neutrosophic operation

(1) the weight of contextual attribute by SVNS entropy

In MCDM problems, different contextual attributes contribute to the decision at different degrees, and thus should be assigned different weight values. In this paper, the weights of each contextual attribute in the SVNS environment are derived using the NS entropy [30]. The entropy of SVNS is used to describe the degree of uncertainly of SVNS, which is defined as follow:

$$E(A) = 1 - \frac{1}{n}\sum_{i=1}^{n}\{(T_A(x_i) + F_A(x_i)) \cdot |I_A(x_i) - I_{A^c}(x_i)|\}$$

where A^c is the complementary set [30] of A and it satisfies that $T_{A^c} = F_A, I_{A^c} = 1 - I_A, F_{A^c} = T_A$.

In this paper, the weight corresponding to each context attribute is completely unknown, which is consistent with the characteristics of entropy. Therefore, the corresponding weights for each attribute are calculated as follows:

$$\omega_{con} = \frac{1 - E(A_{con})}{\sum_{j=1}^{4}(1 - E(A_{con}))} \tag{16}$$

where $con \in \{L, C, D, M\}$ denotes four contextual attributes, $E(A_{con})$ denotes the entropy of the SVNS of the four context properties, that is, the entropy of SVNS represented by the four column vectors of $SVNS(cloudlet_i^{con})_{p\times 4}$.

Next, the contextual SVNS aggregated into the SVNN of the candidate cloudlet based on the SVNS weighted average aggregation operator $SVNSWA_{\omega}$ [26]. The operation will aggregate each row in the matrix $SVNS(cloudlet_i^{con})_{p\times 4}$ to derive the SVNN of candidate $cloudlet_i$ as follows:

$$\begin{aligned} &SVNSWA_{\omega}\left(SVNN_i^L, SVNN_i^C, SVNN_i^D, SVNN_i^M\right) \\ &= \left\langle 1-\prod_{con}\left(1-T_i^{con}\right)^{\omega_{con}}, \prod_{con}\left(I_i^{con}\right)^{\omega_{con}}, \prod_{con}\left(F_i^{con}\right)^{\omega_{con}} \right\rangle \end{aligned} \tag{17}$$

This operation aggregates each row in the matrix $SVNS(cloudlet_i^{con})_{p\times 4}$, $SVNS(i) = \left\{\left\{T_i^L, I_i^L, F_i^L\right\}\left\{T_i^C, I_i^C, F_i^C\right\}\left\{T_i^D, I_i^D, F_i^D\right\}\left\{T_i^M, I_i^M, F_i^M\right\}\right\}$, to obtain the SVNN of the candidate $cloudlet_i$, that is, $SVNN_i = \{T_i, I_i, F_i\}$.

(2) cloudlet score calculation by NS score function

After obtaining the SVNN of $cloudlet_i$, the score of each candidate cloudlet is calculated using the NS score function [25] which is an important metric in SVNN ranking. The score of $cloudlet_i$ is calculated as follow:

$$score(cloudlet_i) = (T_i + 1 - I_i + 1 - F_i)/3 \tag{18}$$

where T_i, I_i and F_i are the truth-membership value, indeterminacy-membership value and falsity-membership value of $cloudlet_i$, respectively. The cloudlet with the highest score is the optimal cloudlet. Now, we have the preliminaries score of $cloudlet_i$ as the target cloudlet with the time-varying context feature.

4.2.2 Composite score by movement periodicity

Studies on human mobility have shown that users often move back and forth to several places, showing periodicity and predictability in their movements [23]. Therefore, we propose a prediction algorithm for cloudlet usage based on tail sequence matching and calculate the predicted cloudlet usage probability. Finally, make the optimal cloudlet decision with the help of this probability and mentioned NS score function.

Definition 1 Tail matching subsequence (TMS). A mobile user's cloudlet history usage sequence are denoted as $S = (S_1, S_2, S_3, ..., S_n)$. Its tail subsequences with length l are denoted as $SE(l) = (S_{n-l+1}, ..., S_{n-1}, S_n)$. If there is a subsequence of S with length of l, $S' = (S'_m, S'_{m+1}, S'_{m+2}, ..., S'_{m+l})$, satisfying $S'_{m+i} = S_{n-l+i}, i \in (0, l)$, then S' is a TMS of the historical usage sequence S.

In the historical use sequence S, there may be more than one TMS corresponding to a tail subsequence SE of a particular length l. If these *TMSs* exist in S, the next element of TMS will most likely be the next element of SE as the prediction target cloudlet.

Definition 2 Tail-matched prediction cloudlet (TMC). A mobile user's cloudlet history usage sequence is $S = (S_1, S_2, ..., S_n)$, $S' = (S'_m, S'_{m+1}, ..., S'_{m+l})$ is a TMS of S, and then the cloudlet indicated by the next access location of S' in S, i.e., S'_{m+l+1} is a TMC for that user.

In the example shown in Fig. 3, a historical usage sequence of cloudlets by mobile users is cloudlet A, D, B, F, C, and so on. We can see $ADBF$ is a TMS of length 4, and BF is a TMS of length 2. The corresponding TMC C and E can also be obtained. According to TMS and TMC, the prediction algorithm of cloudlet usage is detailed in algorithm 1.

Input: cloudlet history usage sequence S;
Onput: TMS set and TMC set
initialize scan pointer $p = S.end$;
initialize the TMS head pointer $C.start = S.start$ and tail pointer $C.end = S.start$;
initialize TMS sets $TMSs = \phi$ and TMC sets $TMCs = \phi$;
while $C.end \; != \; S.length - 1$ **do**
  if $S[p] \; != \; S[C.end]$ **then**
    $C.end = C.end + 1$;
  else
    $C.start = C.end - 1$;
    while $S[p-1] == S[C.start]$ **do**
      $C.start = C.start - 1$;
      $p = p - 1$;
    end while
    $TMSs.append(C(S[C.start + 1], S[C.end]))$;
    $TMCs.append(S[C.end + 1])$;
    $C.end = C.end + 1$;
    $p = S.length - 1$;
  end if
end while
return $TMSs$ and $TMCs$;

Algorithm 1 Prediction algorithm for cloudlet usage based on tail sequence matching (PCTSM)

Obviously, the length of SE is not fixed, and there may be more than one TMS. Therefore, for multiple TMSs, they are given corresponding weights. There will be duplicate elements in $TMCs$. We remove these duplicate elements, and then record the duplicate times of elements in $TMCs$ as $num_1, num_2, ..., num_n$ and the corresponding TMS length as $len_1, len_2, ..., len_n$. Obviously, the larger num and len are, the higher the probability of using the cloudlet next. The using probability is calculated as:

$$prob_i = \omega_1 \frac{num_i}{\sum_{j=1}^{n} num_j} + \omega_2 \frac{len_i}{\sum_{j=1}^{n} len_j} \tag{19}$$

where ω_1 and ω_2 denote the weight of duplicate times of TMC and the length of TMS, respectively, $j \in \{1, 2, ..., n\}$, and n is the number of non-repeating cloudlet in $TMCs$.

We filter out the cloudlets that are not in the prediction range to reduce the computational complexity. For the cloudlets in the prediction range, the score of $cloudlet_i$ will be updated using the probability $prob_i$, and the composite score of $cloudlet_i$ will be calculated as:

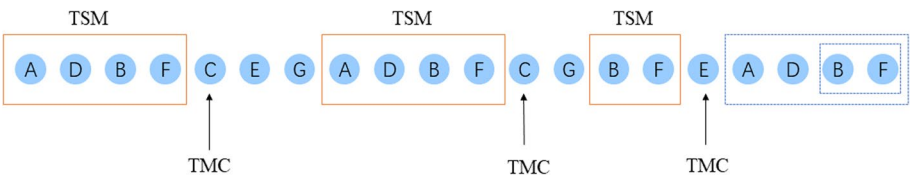

Fig. 3 Illustrate a mobile user's historical usage sequence of cloudlets. *ADBF* is a tail matching subsequence (TMS) of length 4, and *BF* is a TMS of length 2. The tail-matched prediction cloudlet is *C* and *E* correspondingly

$$Cscore_i = score_i * prob_i \tag{20}$$

where $Cscore_i$ is the composite score of $cloudlet_i$ after considering movement periodicity. The best cloudlet is the one with the highest composite score.

4.2.3 Time-varying context-aware cloudlet decision algorithm

The time-varying context-aware cloudlet decision algorithm based on neutrosophic set(TConNS) is shown in Algorithm 2.

Input: the delay threshold T_{timer}, the sequence of four contextual data of p candidate cloudlets in the last q moments $cloudlet_i^j, con \in \{L, C, D, M\}, i \in \{1, 2, ..., p\}, j \in \{1, 2, ..., q\}$;
Onput: the task offloading position
1: **if** there are cloudlets respond to task requests in T_{timer} **then**
2: Normalize the $cloudlet_i^j$ and construct the time-varying context matrix $CON(cloudlet_i^j)_{p\times 4}$
3: **for** $row\ in\ CON(cloudlet_i^j)_{p\times 4}$ **do**
4: **for** $column\ in\ CON(i)$ **do**
5: $\forall con \in \{L, C, D, M\}$, Using Eq.(6*) - Eq.(8*) to perform NS transformation of the
6: time-varying context, $CON(i) \rightarrow \{\hat{E}x_i^{con}, \hat{E}n_i^{con}, \hat{H}e_i^{con}\}$
7: **end for**
8: **end for**
9: construct the SVNS context matrix $SVNS(cloudlet_i^j)_{p\times 4}$
10: **for** $column\ in\ SVNS(cloudlet_i^j)_{p\times 4}$ **do**
11: calculate the weight for each context attribute by Eq.(16)
12: **end for**
13: **for** $row\ in\ SVNS(cloudlet_i^j)_{p\times 4}$ **do**
14: aggregate SVNS context into $SVNN_i$ of candidate $cloudlet_i$ by Eq.(17)
15: calculate the score and the Csore of candidate cloudlets by Eq.(18) and Eq.(20)
16: **end for**
17: the optimal cloudlet is the one with highest composite score
18: **if** $C_{mob} > C_{clt}^{optimal}$ **then**
19: **return** the task offloading position is the optimal cloudlet
20: **else**
21: **return** the task offloading position is the mobile device itself
22: **end if**
23: **else**
24: **return** the task offloading position is the central cloud
25: **end if**

Algorithm 2 Time-varying context-aware cloudlet decision algorithm based on neutrosophic set (TConNS)

In TConNS, step 3 to step 8 convert the context raw data sequence into a context SVNS matrix with a time complexity of $4{\cdot}O(p)$. Step 9 to step 16 use the SVNS matrix to make decisions, and the time complexity is $4{\cdot}O(p)$. The time complexity of the entire algorithm is $C \cdot O(p)$, where C represents the $|con|$, that is the number of context dimensions considered.

5 The experiment, results and discussion

In this section, we conduct several experiments to evaluate the TConNS with better performance. Section 5.1 describes the experimental environment and performance metrics. Section 5.2 compares the TConNS with other three related offloading schemes and analyzed.

5.1 Experimental settings

(1) experimental environment

In this paper, we use Advantech EIS-D210 [31] (3846MIPS, 1.5GHz, 4GB RAM) as the parameter of cloudlet and use Dell PowerEdge (31790MIPS, 3.0GHz, 768GB RAM) as

the parameter of cloud server. The service scope of the cloudlet is set as 50 m, the bandwidth between the user and the cloudlet is set to 100Mbps, and the bandwidth between the user and the cloud server is set as 1Gbps [32]. The task size is allocated according to uniform distribution with an average value of 4600 M instructions and the data transfer size of each task is also allocated in the same way with an average value of 750 kilobytes [33].

The experiment was run on Windows 1020H2, using the python language, version 3.9. Coordinate distances are calculated using python's geodesic package, and *numpy* and *pandas* are used for matrix manipulation and data analysis.

(2) datasets

The experiments in this paper use the EUA dataset [34] and the Alibaba cluster dataset [35]. The EUA dataset collects geographic locations of edge servers and mobile users in the Melbourne, Australia area, and can be used to study mobility in edge environments. The Alibaba cluster dataset records data related to the actual production of 4000 servers over an 8-day period. We extract two kinds of data on CPU utilization and load of the servers from the dataset. Network conditions are measured by network latency.

(3) Comparing algorithms and evaluation metrics

To verify the effectiveness of TConNS, we compare TConNS with three other context-aware offloading schemes.

(a) appAware [15]: This scheme offloads tasks to different cloudlets depending on the requested application type, and tasks will be offloaded to the cloud computing center when no suitable cloudlets is available;

(b) mCloud [14]: In this scheme, different offloading locations are selected by the currently available wireless media of the mobile device. TOPSIS multi-attribute decision making is performed when multiple available wireless media exist;

(c) VMM [19]: A mobility-aware task offloading model is proposed. As the user moves, the task is switched between different cloudlets until the task execution is completed.

(4) Baselines

Four baseline methods were used to further verify the effects of mobility, network conditions, load and CPU utilization on cloudlet selection. These four baselines are simplified versions of TConNS. They are: TCon-wL (without load) that does not consider the load of cloudlet, TCon-wC (without cpuUtil) that does not consider the CPU utilization of cloudlet, TCon-wD (without delay) that does not consider network conditions and TCon-wM (without mobility) that does not consider users' mobility.

We consider three metrics: (a) the average number of task failures; (b)the response time of task completion and (c) energy consumption. A Task whose response time exceeds the delay threshold is defined as a failed task.

5.2 Experimental results and analysis

5.2.1 Case study

Assumed that there are 10 candidate cloudlets near the user. Considering the periodicity of the user's movement, the TMC predicted by algorithm 1 are cloudlets B, D, G and I. The matching subsequence length and the matching times are returned. Assuming that the weights of matching sequence length and times of matches are equal, the predicted

Table 1 TMC related information

TMC	Matching length	Matching times	Usage probability
cloudlet B	4	1	0.20
cloudlet D	4	3	0.32
cloudlet G	2	3	0.25
cloudlet I	5	1	0.23

Table 2 SVNS of cloudlets on time-varying contexts

Candidate cloudlet	*L*	*C*	*D*	*M*
Cloudlet B	0.47, 0.25, 0.02	0.48, 0.30, 0.10	0.42, 0.37, 0.17	0.44, 0.36, 0.14
Cloudlet D	0.49, 0.28, 0.02	0.51, 0.20, 0.12	0.52, 0.32, 0.08	0.60, 0.24, 0.06
Cloudlet G	0.42, 0.22, 0.06	0.46, 0.30, 0.04	0.50, 0.31, 0.08	0.58, 0.28, 0.05
Cloudlet I	0.51, 0.28, 0.02	0.50, 0.20, 0.08	0.54, 0.27, 0.05	0.62, 0.22, 0.10

cloudlet usage probability is calculated using Eq. (19). All the above results are shown in Table 1.

For cloudlet B, D, G and I, within the prediction range, the original data of the four context attributes within 20 moments are taken to characterize the time-varying properties. After normalizing the data, the time-varying context data are transformed into SVNS of the cloudlets' contexts using Eq. (13*) - Eq. (15*), i.e., the matrix $SVNS(cloudlet_i^{con})_{p\times 4}$, as shown in Table 2.

Next, the weights of each context attribute are calculated using Eq. (16) to obtain $\omega_{con} = (0.24, 0.27, 0.22, 0.27)$, where $con \in \{L, C, D, M\}$. We use Eq. (17) to aggregate the SVNS of cloudlets on the time-varying contexts into the SVNN of each candidate cloudlet, and finally, use Eq. (18) and Eq. (20) to calculate the score and the composite score of each candidate cloudlets, as shown in Table 3.

The cloudlet I, which has the highest score, exhibits high truth-membership, low indeterminacy-membership and falsity-membership for each context parameter. Its score is the highest. However, it is not the final choice because the user has the highest probability to access cloudlet D considering the periodicity of user movement, and its composite score is the highest. Therefore, here the cloudlet D is the optimal cloudlet.

5.2.2 Comparative analysis

(1) Proportion of tasks offloaded to different locations

Figure 4 shows the percentage of 100 tasks offloaded to different locations by using three schemes. When the cloudlet does not meet the requirements of the task to be processed, appAware and mCloud choose to process the task on the local device or the center cloud instead of looking for another cloudlet. In appAware, the proportion of offloading to cloud is larger due to only the availability of the application to process a certain type of task on the cloudlet. In the experiment of mCloud, only the wireless medium is considered as a contextual condition, and only the availability of the wireless medium

Table 3 Candidate cloudlets' SVNN and its score

Candidate cloudlet	SVNN	Score	*Cscore*
Cloudlet B	0.46, 0.32, 0.08	2.06	0.41
Cloudlet D	0.53, 0.25, 0.06	2.22	0.71
Cloudlet G	0.50, 0.28, 0.05	2.17	0.54
Cloudlet I	0.55, 0.24, 0.06	2.25	0.52

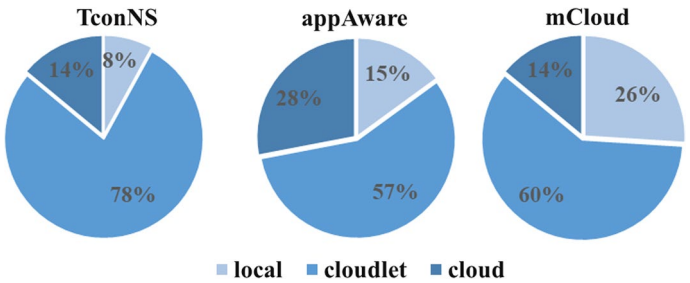

Fig. 4 Proportion comparison of offloading to different locations. The figure shows the percentage of 100 tasks offloaded to three different locations (mobile device itself, cloudlets and the central cloud) by using three schemes

is used to decide where to process the task. When there is no available WIFI media, there are a large number of tasks processed locally on the device.

(2) Number of failed tasks

The number of task failures is measured when the number of tasks is 20, 40, 60, 80, and 100, respectively. The number of task failures of TConNS and the comparison experiments are shown in Fig. 5.

TConNS captures the user's movement tendency by predicting the user's residence time in the cloudlet scope. If the expected residence time in a cloudlet is small, this cloudlet is assigned a small truth-membership and a small comprehensive score. Thus, this cloudlet cannot become the best cloudlet, avoiding task migration and reducing the number of task failures. In VMM, as users move, tasks switch between different cloudlets leads to a considerable increase in the number of failed tasks. In addition, TConNS considers multiple contextual attributes and selects the best cloudlet to provide high-quality service for tasks and reduce the number of task failures, which is more comprehensive than appAware and mCloud that only consider a single context. Furthermore, the data sequence of the most recent period is chosen by TConNS to characterize the time-varying properties of the attributes. The appropriate context in the most recent period is also appropriate in the most recent future, which has a significant impact on best cloudlet decision. Selecting the best cloudlet can reduce the number of failed tasks.

(3) Response time and energy consumption

The comparisons of the response time and energy consumption between TConNS and the other three algorithms when processing different number of tasks are shown in Fig. 6 and Fig. 7. The average response time of TConNS is about 49% lower and the average energy consumption is about 46% lower compared to the other schemes.

The above comparison results show the comprehensive effect of TConNS is better. The large proportion of tasks offloaded to the cloudlet (see Fig. 4) and the small number of

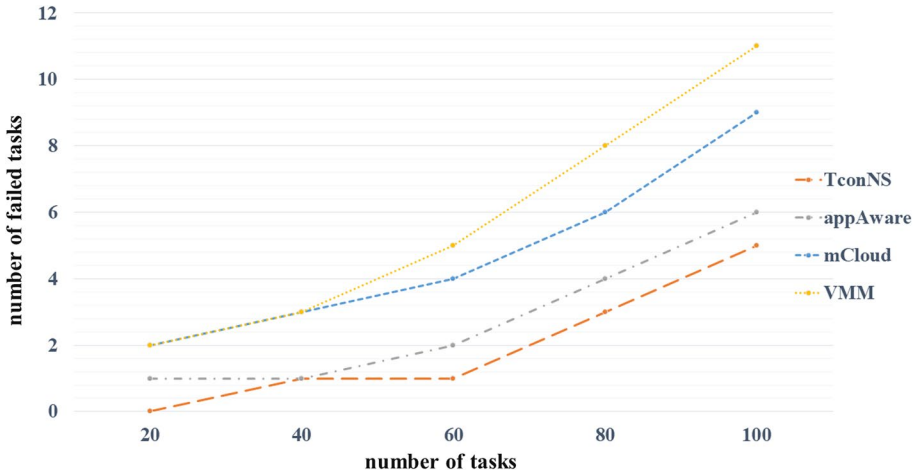

Fig. 5 Comparison of failed tasks. The figure shows the number of failure tasks of TConNS and the comparison schemes. The number of task failures is measured when the number of tasks is 20, 40, 60, 80, and 100, respectively. It validates that TconNS can reduce the number of failed tasks

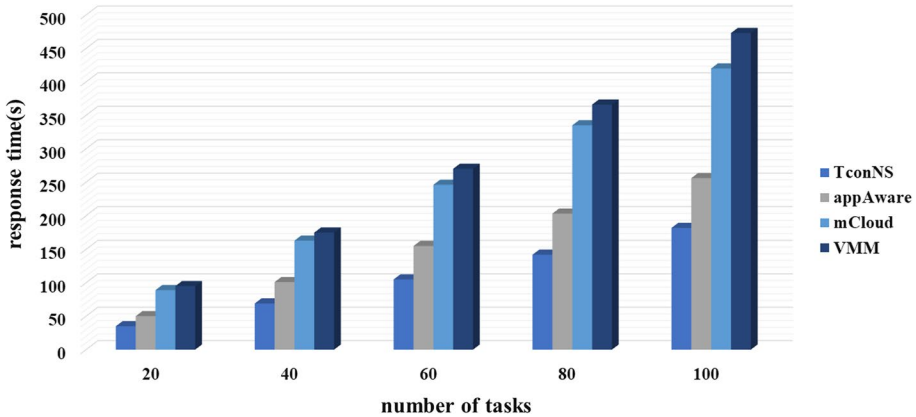

Fig. 6 Comparison of response time. The figure shows the comparison of the response time between TConNS and the other three algorithms when processing different number of tasks (n=20, 40, 60, 80, 100). The average response time of TConNS is about 49% lower compared to the other schemes

failed tasks (see Fig. 5) analyzed above both contribute to the lower response time and energy consumption of TConNS. In addition, TConNS integrates the four time-varying contexts of cloudlet load, CPU utilization, network conditions, and user mobility for optimal cloudlet selection. The response time and energy consumption of the optimal cloudlet the task are lower, while the other comparison methods do not consider the selection of the optimal cloudlet.

(4) Baselines Analysis

Figure 8 depicts comparison results of number of failed tasks among TConNS and baselines. Figure 9 and 10 depicts comparison results of response time and energy consumption among TConNS and baselines with different number tasks, respectively. It can be seen that the fusion of these four context properties is crucial to the choice of cloudlet. In our TConNS, the cloudlet is chosen with the impact of multiple contextual attribute information taken into account, rather than modeling from a single

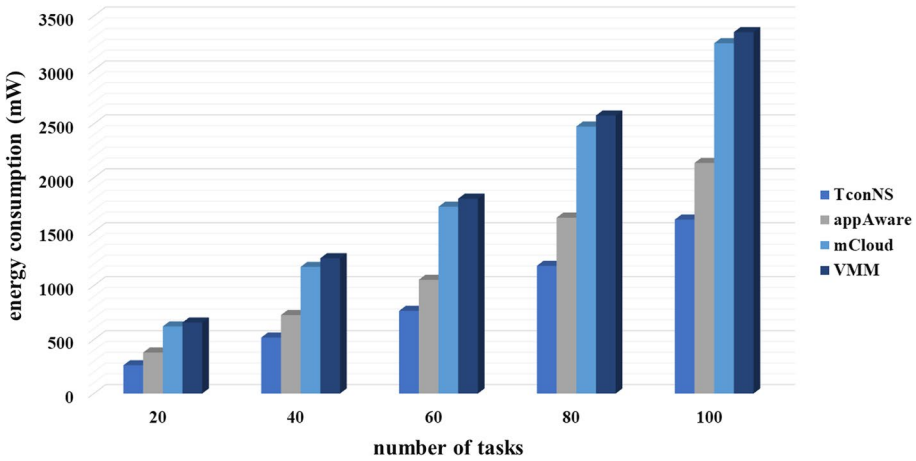

Fig. 7 Comparison of energy consumption. The figure shows the comparison of energy consumption between TConNS and the other three algorithms when processing different number of tasks (n=20, 40, 60, 80, 100). The average energy consumption of TConNS is about 46% lower compared to the other schemes

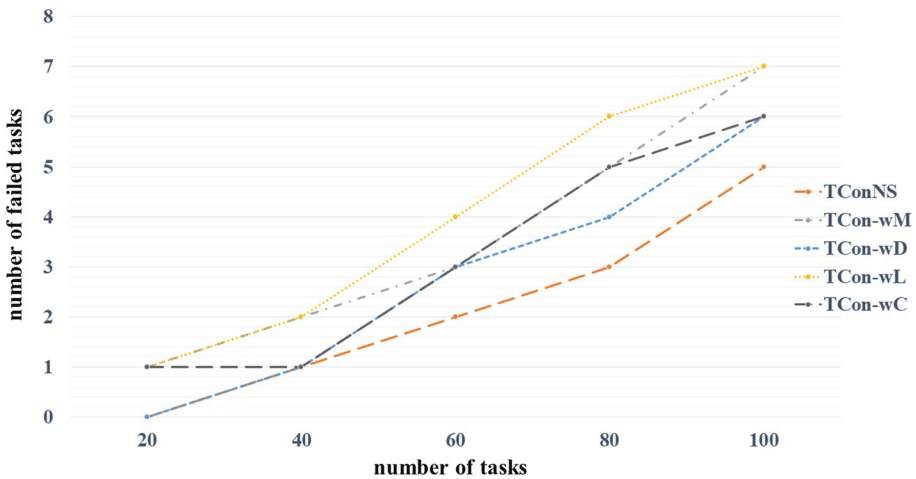

Fig. 8 Comparison of the number of failed tasks with baselines. The figure shows the comparison of failed tasks between TConNS and the other four baselines when processing different number of tasks (n=20, 40, 60, 80, 100)

dimension of information. The load condition corresponds to queuing time, CPU utilization corresponds to processing time, network conditions correspond to communication delay, and mobility corresponds to propagation time. And, all of these time are related to energy consumption. TConNS covers most possible influencing factors, and the best cloudlet can be decided to achieve the goal of time saving and energy saving.

And, it can be found that each contextual factor has a different degree of influence on the selection of cloudlets and the offloading of tasks. Among them, the load is the most 'prominent' and has the greatest impact on the results. This is consistent with the weights of attributes calculated using the entropy weight method as shown in Eq. (16), and also indicates the effectiveness of the entropy weight method.

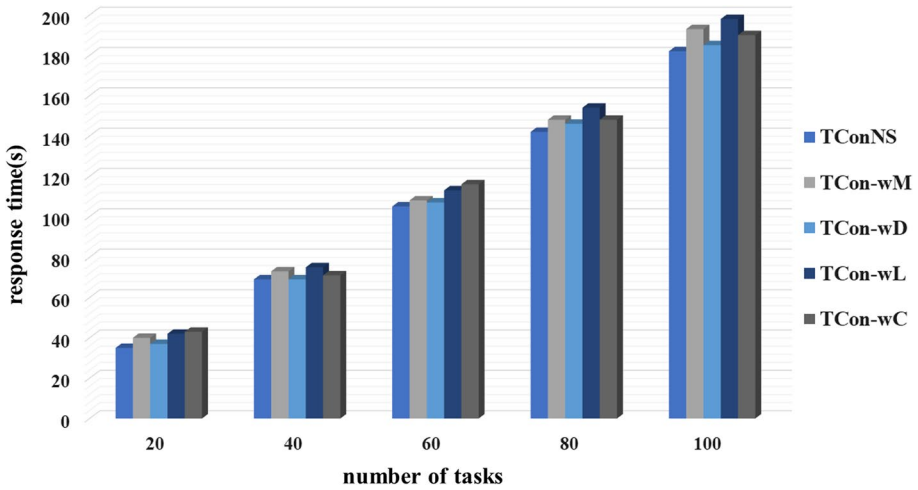

Fig. 9 Comparison of the response time with baselines. The figure shows the comparison of response time between TConNS and the other four baselines when processing different number of tasks (n=20, 40, 60, 80, 100)

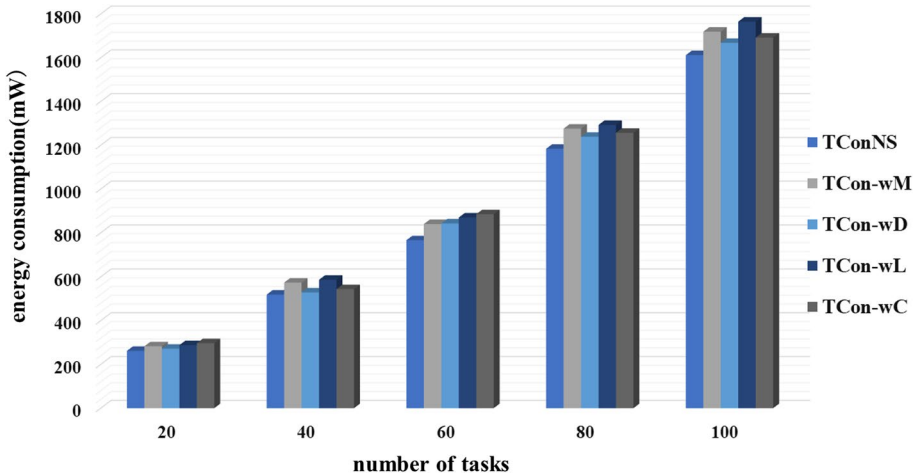

Fig. 10 Comparison of the energy consumption with baselines. The figure shows the comparison of energy consumption between TConNS and the other four baselines when processing different number of tasks (n=20, 40, 60, 80, 100)

In summary, TConNS fully considers the four context attributes of cloudlet load, CPU utilization, network conditions and user mobility to select the best cloudlet from the global perspective, which is better than considering only the single-dimensional context method.

6 Conclusion

In this paper, we investigated task offloading and edge server selection under mobile edge computing. A time-varying context-aware cloudlet decision framework based on the NS was proposed. Four context attributes with time-varying characteristics, including load, CPU utilization, network conditions and mobile residence time, were considered for multi-attribute cloudlet decision making. The neutrosophic set is used to

characterize the time-varying context, the SVNS model of context attributes is established, and the optimal cloudlet is selected by the neutrosophic aggregation operation and neutrosophic score function. Simulation experimental results show that the proposed strategy performs better in terms of the number of task failures, response time and energy consumption compared to other offloading scheme.

In future work, we will consider incorporating more scenario-tailored contextual attributes, as well as using deep learning methods to predict user trajectories to measure user mobility more accurately.

Abbreviations

MEC	Mobile edge computing
TConNS	Time-varying context-aware cloudlet decision algorithm based on neutrosophic set
SVNS	Single-valued neutrosophic set
SBS	Small cell base station
DQN	Deep Q-network
NS	Neutrosophic set
MDP	Markov decision process
SE	Spectral efficiency
GPS	Global positioning system
SVNN	Single-valued neutrosophic number
CM	Cloud model
MCDM	Multi-criteria decision-making
TMS	Tail matching subsequence
TMC	Tail-matched prediction cloudlet

Acknowledgements
The authors gratefully acknowledge the valuable comments and suggestions of the reviewers, which have improved the presentation of this article.

Author's contributions
MZ and JL conceived the main idea and contributed to the writing and the revisions. JL and YH conducted the experiments. All authors read and approved the final manuscript.

Funding
This work is supported by the National Natural Science Foundation of China(No.62172442) and (No.62172451), and Natural Science Foundation of Hunan Province (No. 2020JJ5775).

Availability of data and materials
The datasets used or analyzed during the current study are available from the corresponding author on reasonable request.

Declarations

Competing interests
The authors declare that they have no competing interests.

Received: 5 February 2023 Accepted: 21 December 2023
Published online: 04 January 2024

References

1. Y. Mao, C. You, J. Zhang, K. Huang, K.B. Letaief, A survey on mobile edge computing: The communication perspective. IEEE Commun. Surv. Tutor. **19**(4), 2322–2358 (2017)
2. D. Wang, X. Tian, H. Cui, Z. Liu, Reinforcement learning-based joint task offloading and migration schemes optimization in mobility-aware MEC network. Chin. Commun. **17**(8), 31–44 (2020)
3. X. Xu, Q. Liu, Y. Luo, K. Peng, X. Zhang, S. Meng, L. Qi, A computation offloading method over big data for iot-enabled cloud-edge computing. Futur. Gener. Comput. Syst. **95**, 522–533 (2019)
4. C. Zhang, Z. Zheng, Task migration for mobile edge computing using deep reinforcement learning. Futur. Gener. Comput. Syst. **96**, 111–118 (2019)
5. W. Zhang, Y. Wen, Energy-efficient task execution for application as a general topology in mobile cloud computing. IEEE Trans. Cloud Comput. **6**(3), 708–719 (2018)
6. N. Fernando, S.W. Loke, W. Rahayu, Computing with nearby mobile devices: a work sharing algorithm for mobile edge-clouds. IEEE Trans. Cloud Comput. **7**(2), 329–343 (2019)

7. G. Mohammad, Z. Mehran, H. Abolfazl Toroghi, A fast hybrid multi-site computation offloading for mobile cloud computing. J. Netw Comput. Appl. **80**, 219–231 (2017)
8. Y. Zhang, D. Niyato, P. Wang, Offloading in mobile cloudlet systems with intermittent connectivity. Futur. Gener. Comput. Syst. **14**(12), 2516–2529 (2015)
9. F. Yu, H. Chen, J. Xu, Dmpo: Dynamic mobility-aware partial offloading in mobile edge computing. IEEE Trans. Mob. Comput. **89**, 722–735 (2018)
10. Z. Wang, Z. Zhao, G. Min, X. Huang, Q. Ni, R. Wang, User mobility aware task assignment for mobile edge computing. Futur. Gener. Comput. Syst. **85**, 1–8 (2018)
11. P. Nawrocki, B. Sniezynski, J. Kolodziej, P. Szynkiewicz, Adaptive context-aware energy optimization for services on mobile devices with use of machine learning considering security aspects. In: 20th IEEE/ACM international symposium on cluster, cloud and internet computing(CCGRID),Melbourne, Australia, May 11-14, pp. 708–717 (2020)
12. S. Li, S. Lin, L. Cai, W. Li, G. Zhu, Joint resource allocation and computation offloading with time-varying fading channel in vehicular edge computing. IEEE Trans. Veh. Technol. **69**(3), 3384–3398 (2020)
13. T. Wu, X. Fan, Y. Qu, Mobiedge: Mobile service provisioning for edge clouds with time-varying service demands. In: 27th IEEE international conference on parallel and distributed systems(ICPADS),Beijing, China, Dec 14-16, pp. 490–497 (2021)
14. B. Zhou, A.V. Dastjerdi, R.N. Calheiros, S.N. Srirama, R. Buyya, mcloud: A context-aware offloading framework for heterogeneous mobile cloud. IEEE Trans. Serv. Comput. **10**(5), 797–810 (2017)
15. D.G. Roy, D. De, A. Mukherjee, R. Buyya, Application-aware cloudlet selection for computation offloading in multi-cloudlet environment. J. Supercomput. **73**(4), 1672–1690 (2017)
16. A. Salehan, H. Deldari, S. Abrishami, Application-aware cloudlet selection for computation offloading in multi-cloudlet environment. J. Supercomput. **75**(7), 3769–3809 (2019)
17. W. Junior, E. Oliveira, A. Santos, K.L. Dias, A context-sensitive offloading system using machine-learning classification algorithms for mobile cloud environment. Futur. Gener. Comput. Syst. **90**, 503–520 (2019)
18. X. Chen, T. Tan, G. Cao, Energy-aware and context-aware video streaming on smartphones. In: 39th IEEE international conference on distributed computing systems(ICDCS),Dallas, TX, USA, Jul 7-10, pp. 861–870 (2019)
19. A. Mukherjee, D.G. Roy, D. De, Mobility-aware task delegation model in mobile cloud computing. J. Supercomput. **75**(1), 314–339 (2019)
20. Z. Xu, S. Wang, S. Liu, H. Dai, Q. Xia, W. Liang, G. Wu, Learning for exception dynamic service caching in 5g-enabled mecs with bursty user demands. In: 40th IEEE international conference on distributed computing systems(ICDCS),Singapore, Nov 29–Dec 1, pp. 1079–1089 (2020)
21. Y. Liu, W. Wang, H. Chen, F. Lyu, L. Wang, W. Meng, X. Shen, Physical layer security assisted computation offloading in intelligently connected vehicle networks. IEEE Trans. Wireless Commun. **20**(6), 3555–3570 (2021)
22. L. Qi, W. Dou et al., A context-aware service evaluation approach over big data for cloud applications. IEEE Trans. Cloud Comput. **8**(2), 338–348 (2020)
23. P. Pirozmand, G. Wu, B. Jedari, F. Xia, Human mobility in opportunistic networks: Characteristics, models and prediction methods. J. Netw. Comput. Appl. **42**, 45–58 (2014)
24. H. Wang , F. Smarandache, Y. Zhang, R. Sunderraman, Single valued nertrosohpic sets. Rev. Air Force Acad. **10** (2012). https://www.researchgate.net/publication/262034557
25. F. Smarandache, A unifying field in logics: Neutrosophic logic. Multiple-Valued Logic **8**(3) (1999). https://www.researchgate.net/publication/266416576
26. J. Peng, J. Wang, J. Wang, H. Zhang, X. Chen, Simplified neutrosophic sets and their applications in multi-criteria group decision-making problems. Int. J. Syst. Sci. **47**(10), 2342–2358 (2016)
27. A.Q. Ansari, R. Biswas, S. Aggarwal, Proposal for applicability of neutrosophic set theory in medical ai. Int. J. Comput. Appl. **27**(5), 5–11 (2011)
28. H. Ma, Z. Hu, K. Li, H. Zhang, Toward trustworthy cloud service selection: A time-aware approach using interval neutrosophic set. J. Parall. Distribut. Comput. **96**, 75–94 (2016)
29. D. Li, H. Meng, X. Shi, Membership clouds and membership cloud generators. J. Comput. Res. Develop. **32**(6), 15–20 (1995)
30. P. Majumdar, S.K. Samanta, On similarity and entropy of neutrosophic sets. J. Intell. Fuzzy Syst. **26**(3), 1245–1252 (2014)
31. Advntech: the edge-side inference server based on Nvidia Jetson Nano platform. [EB/OL]. https://www.advantech.com.cn/products/9140b94e-bcfa-4aa4-8df2-1145026ad613/mic-710ai/mod_784f6f0f-8402-450c-981f-3122b85cd489
32. A.D.S. Veith, M.D. de Assunção, L. Lefèvre, Latency-aware placement of data stream analytics on edge computing. In: 16th international conference on service-oriented computing(ICSOC),Hangzhou, China, November 12-15. Lecture notes in computer science, vol. 11236, pp. 215–229 (2018)
33. L. Yang, B. Liu, J. Cao, Y. Sahni, Z. Wang, Joint computation partitioning and resource allocation for latency sensitive applications in mobile edge clouds. IEEE Trans. Serv. Comput. **14**(5), 1439–1452 (2021)
34. P. Lai, Q. He, M. Abdelrazek, F. Chen, J.C. Hosking, J.C. Grundy, Y. Yang, Optimal edge user allocation in edge computing with variable sized vector bin packing. In: 16th international conference on service-oriented computing(ICSOC),Hangzhou, China, Nov 12-15. Lecture notes in computer science, vol. 11236, pp. 230–245 (2018)
35. A. I. Alibaba production cluster data V2018. [EB/OL]. https://github.com/alibaba/clusterdata/tree/v2018

Publisher's Note