Future Generation Computer Systems 26 (2010) 207–216

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Context reasoning using extended evidence theory in pervasive computing environments

Daqiang Zhang [a,*], Minyi Guo [a], Jingyu Zhou [a], Dazhou Kang [b], Jiannong Cao [c]

[a] Department of Computer Science, Shanghai Jiao Tong University, Shanghai, 200240, PR China
[b] Department of Computer Science, Nanjing University, Nanjing, 210089, PR China
[c] Department of Computing, The Hong Kong Polytechnic University, Hong Kong

ARTICLE INFO

Article history:
Received 15 February 2009
Received in revised form
27 July 2009
Accepted 3 August 2009
Available online 7 August 2009

Keywords:
Evidence theory
Context reasoning
Pervasive computing

ABSTRACT

Most existing context reasoning approaches implicitly assume that contexts are precise and complete. This assumption cannot be held in pervasive computing environments, where contexts are often imprecise and incomplete due to unreliable connectivity, user mobility and resource constraints. To this end, we propose an approach called CRET: Context Reasoning using extended Evidence Theory. CRET applies the evidence theory to context reasoning in pervasive computing environments. Because evidence theory is limited by two fundamental problems – computation-intensiveness and Zadeh paradox, CRET presents evidence selection and conflict resolution strategies. Empirical study shows that CRET is desirable for pervasive applications.

1. Introduction

Pervasive computing aims to create smart environments that embed computation and communication in a manner that contextually interacts with users to ease their daily life [1,2]. Context reasoning enables pervasive applications to automatically adapt to the changeable contexts and to unobtrusively integrate users into their environments. The proliferation of pervasive applications has fostered an increasing attention to context reasoning, where contexts often refer to pieces of information that captures the characteristics of pervasive computing.

Contexts collected in pervasive computing environments are often imprecise and incomplete [3–5]. A number of sensing and computational devices, such as hand-held devices, sensor networks and Radio Frequency IDentification (RFID), can detect contexts, but they cannot provide precise contexts. For instance, in a smart meeting room, a sensor detects a context — *the location of the user is meeting room*. Usually, the sensor does not know the exact location of the user. Furthermore, sensors often miss pieces of contextual information. When the user talks in the meeting room, he or she uses many mimics to express his or her idea appropriate to the topic. In this case, contexts collected by sensors are highly imprecise and incomplete. This is because the sensor technology, collected data and their interpretation as contexts are prone to error. The sources of errors consist of, but are not limited to, inaccurate measurement and noise from external environments [6,7]. This case is just a common example in pervasive computing environments, and it is an essential requirement for context reasoning to handle such contexts.

A variety of context reasoning approaches have been proposed to assist pervasive applications in adaption, but they mainly focus on inferring contexts from precise and complete contexts. Bayesian network [8,9], case-based [10], logic-based [11], ontology-based [12–14] and rule-based [15] have been exploited in pervasive computing environments. The typical scenario is given as below. Sensors are embedded into physical spaces to continuously collect and report their readings. Once the contexts change, context reasoning will infer hidden contexts according to sensor readings, and then notifies pervasive applications to adapt. However, most existing approaches implicitly assume that the contexts being checked are complete and precise. This assumption cannot hold in pervasive computing environments, where contexts are often imprecise and incomplete due to unreliable connectivity, user mobility, and resource constraints.

Evidence theory is a technique for reasoning the truth from pieces of information [16]. It has achieved widespread success in auditing, decision support, financial asset evaluation, process engineering and quality control. However, evidence theory is severely limited by two fundamental problems — computation-intensiveness and Zadeh paradox. In evidence theory, Dempster's

* Corresponding author.
E-mail addresses: zhangdq@sjtu.edu.cn (D. Zhang), csjcao@comp.polyu.edu.hk (J. Cao).

0167-739X/$ – see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.future.2009.08.005

rule is used to combine evidence, but it is extremely computation-intensive when combining a large amount of evidence [17]. It has been proven that Dempster's rule of combination is NP-complete [18]. This problem can be partially compensated by Bayes approximation, but it only works for singleton evidence [19]. On the other hand, evidence theory does not consider conflicting evidences, which may lead to a conclusion in favor of a less probable event, a phenomenon known as the Zadeh paradox [20]. Contexts which are highly imprecise, incomplete and vary with individuals, significantly restrict evidence theory in pervasive computing environments.

In this paper, we propose an approach – CRET: Context Reasoning using extended Evidence Theory – to deal with inconsistent contexts in pervasive computing environments. CRET is applied to context collection, representation, storage and reasoning in our context-aware architecture. In order to reduce computation overhead, CRET presents a $k - l$ strategy for evidence selection to filter the less important evidence. This strategy sorts the evidence according to the weighted sum of their masses and then selects evidence with the highest beliefs. With respect to the Zadeh paradox, CRET introduces conflicting factors. To summarize, the main contributions of this paper are two-fold. (1) It introduces an approach using evidence theory for context reasoning in pervasive computing environments. It systematically shows how to apply evidence theory to context reasoning. (2) It solves the two fundamental problems of evidence theory by evidence selection and conflict resolution strategies.

The remainder of this paper is organized as follows. Section 2 introduces the related work. Section 3 briefly gives an overview of the evidence theory. Section 4 describes the CRET approach in detail and discusses evidence selection and conflict resolution strategies. Section 5 reports our empirical study and Section 6 concludes our work with future directions.

2. Related work

Context reasoning is an important technique for context-awareness and has generated a great many approaches. In general, they can be classified into case-based, logic-based, ontology-based, probabilistic and rule-based approaches. In this section, we briefly give an overview of them, as well as the advances in evidence theory.

Case-based context reasoning approaches infer contexts based on the past cases [21,10,22]. Case-based approaches, however, suffer from two problems. One is how to automatically abstract facts and generate cases, and the other is how to measure the similarity among cases. Rule-based approaches share the similar idea with case-based approaches, together with the similar limitations [15,23].

Logic-based context reasoning, involving first-order and temporal logic, is another popular reasoning scheme [8]. Gaia project used it to represent context in the form of predicate, and reasoned about high-level context based on the pre-defined rules [11]. Logic-based approaches are designed for exact reasoning so that they are not suitable for inference from imprecise and incomplete contexts. Moreover, they do not take semantics into consideration. Note that approaches based on the fuzzy logic (e.g., Type I and Type II fuzzy logic) are dedicated to fuzzy reasoning, but their systems are not easy to implement and their computation overheads are quite heavy.

Ontology-based reasoning approaches incorporate the semantics into context representation and reasoning [24]. With the support of powerful ontology software such as Stanford Protégé, IBM IODT and Maryland Swo, ontology is widely used for context representation in pervasive applications. Yet, it has limited capability in dynamically inferring contexts. It requires defining all the rules or training a model beforehand, and a premise that all ontologies related to the specific domain must be defined already. Normal users in pervasive applications cannot satisfy these requirements due to lack of comprehensive knowledge about their domains, and thus they cannot but resort to domain experts. This incurs higher human cost and restricts the ontology application.

Probabilistic reasoning approaches make use of probability theory to predict uncertain contexts. Bayesian network can be regarded as a standard, even canonical, probabilistic reasoning technique [8,25–28]. It represents contexts by graph and probability. It requires that all hypotheses should be exclusive and exhaustive, which cannot be held in pervasive applications. Meanwhile, the Bayesian network is also criticized for its exponential computation overhead [29,30].

In contrast, evidence theory seems to be a better choice for inference from imprecise and incomplete contexts for its capability of constructing the ground truth from pieces of information. It relaxes the requirement held by the Bayesian network, and allows for probability assignment to sets or intervals. It has attracted a great deal of attention and generated many applications in various domains [31–34]. However, evidence theory suffers from two rudimentary problems — computation-intensive and Zadeh paradox. These two problems become conspicuous in pervasive applications, because contexts are not only continuously varied and generated, but also highly conflicting. In [19], Bayesian approximation was proposed to reduce the computation overhead of evidence theory, but it works for those applications that only care about singleton evidence. In [35], constant approximation cuts down the computation overhead at the cost of accuracy. Regarding the Zadeh paradox, several methods have been proposed in [36,37], but they cannot solve the Zadeh paradox completely.

3. Background

Evidence theory is a mathematical theory of evidence that constructs a coherent picture of reality through computing the probability of an event given evidence [16,38]. It is founded on the grounds of the following concepts and principles.

3.0.1. Frame of discernment

Let Θ be the Frame Of Discernment (FOD), denoting a set of mutually exclusive and exhaustive hypotheses about problem domains. Correspondingly, 2^{Θ} is the power set of Θ.

3.0.2. Mass

Mass stands for a belief mapping from 2^{Θ} to the interval between 0 and 1, represented as m. The masses of the empty set and the sum of all the subsets in power set are 0 and 1, respectively. Mass can be assigned to sets or intervals. Let $\emptyset$ and C be the empty set and a subset of Θ. Mass m is defined as Eq. (1).

$$\begin{aligned} m : 2^{\Theta} &\rightarrow [0, 1] \\ m(\emptyset) &= 0 \\ \sum_{C \subseteq \Theta} m(C) &= 1. \end{aligned} \tag{1}$$

3.0.3. Belief and plausibility

The belief of a hypothesis is the sum of the beliefs for those hypotheses that are its subsets. Conversely, the plausibility of a hypothesis is the sum of all the beliefs of sets that intersect with it. Eq. (2) gives their definitions:

$$Bel(C) = \sum_{B|B\subseteq C} m(B)$$
$$Pls(C) = \sum_{B|B\cap C\neq\emptyset} m(B), \quad (2)$$

where B is a subset of Θ. Bel is the degree of belief to which the evidence supports C, constituted by the sum of the masses of all sets enclosed by it. Pls denotes the degree of belief to which the evidence fails to refute C, that is, the degree of belief to which it remains plausible, i.e., the possibility that the hypothesis could possibly happen.

3.0.4. Dempster's rule

In order to aggregate the evidence from multiple sources, Dempster's rule plays a significantly meaningful and interesting part. Let C_i be a subset of Θ and $m_j(C_i)$ be a mass assignment for hypothesis C_i collected from the j-th source. For subsets $\{C_1, C_2, \ldots, C_n\}$ and mass assignments $\{m_1, m_2, \ldots, m_n\}$, Dempster's rule is given as:

$$\underline{m}\{C\} = (m_1 \oplus m_2 \oplus \ldots \oplus m_n)(C)$$
$$= \frac{1}{K}\sum_{\tau=C} m_1(C_1)\cdot m_2(C_2)\ldots m_n(C_n), \quad (3)$$

where τ equals to $C_1 \cap C_2 \cap \ldots \cap C_n$, and K is the normalizing constant that is defined as Eq. (4).

$$K = 1 - \sum_{\tau=\emptyset} m_1(C_1)\cdot m_2(C_2)\ldots m_n(C_n). \quad (4)$$

Eqs. (3) and (4) show that Dempster's rule combines evidence over the set of all evidence. Its complexity grows exponentially with the increase of the amount of evidence. In [18], the complexity of Dempster's rule is proved as NP-complete. Meanwhile, once some probabilities of evidence are zero, Dempster's rule will get a wrong inference that a less probable event is regarded as the most probable event that could happen. This phenomenon is named for Zadeh paradox.

4. Context reasoning using extended evidence theory in pervasive computing environments

We present a context-aware architecture for pervasive applications, where CRET: Context Reasoning using extended Evidence Theory, is applied. We first introduce our context-aware architecture and then describe CRET in detail, followed by discussions.

4.1. System model

Fig. 1 illustrates our context-aware architecture, which is a hierarchical module consisting of context providers, a manager and consumers. The context providers gather context data on the environments from sensors. Note that contexts may come from applications, and our context collection is capable of collecting contexts at intervals or on demand. The context manager preprocesses contexts and responds to requests from adapters and services. After the context information is generated, the context manager will format it and remove inconsistencies in the preprocessing step, and then move into the reasoning step where hidden

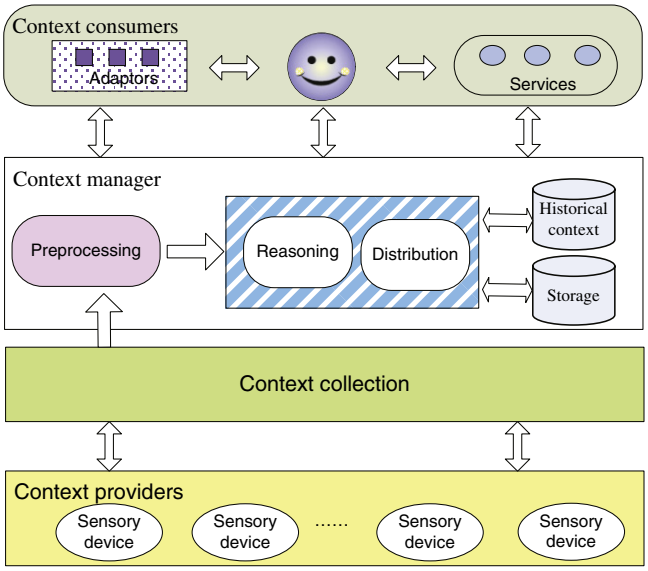

Fig. 1. Context-aware architecture. Context-awareness is provided by context collection, preprocessing, reasoning and distribution.

contexts are derived from present and historical contexts. A historical context database is introduced to store the past contexts, because contexts that are updated frequently increase in size considerably. Finally, the context manager notifies adapters and services, and distributes contexts to them according to their specific privacy and security requirements and their resource constraints. We are implementing a prototype to demonstrate our context-aware architecture. We employ TinyOS 2.0, Micaz and Microsoft SQL server to collect, preprocess and store contexts. Our prototype currently supports devices with Bluetooth, Wi-Fi and WLAN function.

It is noteworthy to point out that CRET is applied to context collection, reasoning and storage in the context manager. This enables the context manager to handle imprecise and incomplete contexts. The goal of this study is to provide a context-aware architecture with context reasoning functionality. The current version of CRET is a centralized approach, where a static node is selected to centrally control the system. From here on, we describe the process of applying evidence theory to pervasive applications and then extend evidence theory to overcome its two significant problems.

4.2. CRET model

We model the sensor-driven pervasive applications as a hierarchical structure, consisting of sensors, objects, and activity layers [39]. Sensors are deployed to monitor objects in the sensor layer. All objects are in the object layer, where some objects are monitored by sensors. Activities are in the activity layer, which can be inferred according to the contexts of objects. In order to formally describe our model, we define $S = \{s_1, s_2, \ldots, s_n\}$ as the set of sensors, $O = \{o_1, o_2, \ldots, o_n\}$ as the set of objects, $A = \{a_1, a_2, \ldots, a_n\}$ as the set of activities, *SO* as simple objects, monitored by sensors directly, *DO* as deduced objects, obtained from *SO*, *CO* as composite objects, comprised of *SO* or *DO*, *E* as the set of evidence, and *D* as discount rate, reflecting the reliability of sensor readings. The system infers the activity candidate set and then makes a decision using Dempster's rule.

Fig. 2 illustrates our system model, where o_5 and o_6 are deduced object and composite object, and lines with various arrows denote relationships among sensors, objects and activities. Objects o_1, o_2, o_3 and o_4 are monitored by sensors s_1, s_2, s_3 and s_4, respectively. Object o_5 is deduced from o_1, and o_6 is made up of objects o_2 and

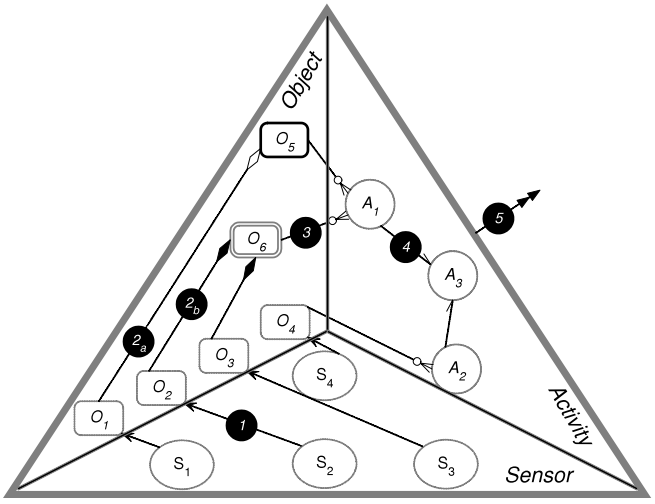

Fig. 2. System model for CRET.

o_3. Suppose a scenario that *a sensor is deployed to monitor the door of a refrigerator where coffee, milk and tea are stored. When the door is open, users make a drink by selecting coffee, milk or tea, or their combinations as simple* a_1 *or* a_2*, or complex activity* a_3*. In this case, the refrigerator is type of object* o_4*, coffee, milk and tea are type of object* o_5 *and their combinations are type of object* o_6.

In Sections 4.3–4.5, we describe how to apply CRET to this scenario. The process of applying CRET consists of (1) propagating evidence in the sensors layer, (2) propagating evidence in the objects layer, and (3) recognizing activities. In Section 4.6, we solve the two problems of CRET - intensive computation and Zadeh paradox.

4.3. Propagating evidence in the sensors layer

Sensors are vulnerable and vary drastically with environments, which leads to unreliable sensor readings. Therefore, it is important to take into account the reliability of sensors in the evidence aggregation process. As proposed by Shafer [16], this reliability can be controlled by the discount rate, which is derived from the sensor reliability model [31]. In CRET, the sensor discount rate is incorporated, given as Eq. (5):

$$m^r(C) = \begin{cases} (1-r)m(C) & C \neq \Theta \\ r + (1-r)m(\Theta) & otherwise, \end{cases} \tag{5}$$

where r is the sensor discount rate with its value between 0 and 1. When r is 0, the sensor is completely reliable; when r is 1, the sensor is absolutely unreliable.

In the beginning, sensors s_1, s_2, s_3 and s_4 are installed and working with discount rates as 0.02, 0.02, 0.1 and 0.2. Evidence on sensor nodes is represented by masses as: $m_{s_1}(s_1) = 1$, $m_{s_2}(s_2) = 1$, $m_{s_3}(\rightharpoonup s_3) = 1$ and $m_{s_4}(s_4) = 1$. Then, the discounted masses are calculated as:

$$\begin{array}{llllll} m^r(s_1) & = & 0.98, & m^r(s_1, \rightharpoonup s_1) & = & 0.02 \\ m^r(s_2) & = & 0.98, & m^r(s_2, \rightharpoonup s_2) & = & 0.02 \\ m^r(\rightharpoonup s_3) & = & 0.9, & m^r(s_3, \rightharpoonup s_3) & = & 0.1 \\ m^r(s_4) & = & 0.8, & m^r(s_4, \rightharpoonup s_4) & = & 0.2. \end{array}$$

There is a relationship between sensors and their associated objects. For example, given the frames Θ_a and Θ_b of sensor s_a and its associated object o_b, the relationship between sensor s_a and object o_b denotes the evidence propagation from sensors to objects layers. CRET associates objects with sensors by maintaining a compatible relationship between them, which is defined as evidential mapping by Eq. (6):

$$\Gamma : m(o_b) = f(s_a - > o_b), \tag{6}$$

Table 1
Evidential mapping.

Relationship		Evidence value
$\{o_1\}$	$\longrightarrow \{o_5\}$	0.9
$\{\rightharpoonup o_1\}$	$\longrightarrow \{\rightharpoonup o_5\}$	1.0
$\{o_1\}$	$\longrightarrow \{o_5, \rightharpoonup o_5\}$	0.1
$\{o_1, \rightharpoonup o_1\}$	$\longrightarrow \{o_5, \rightharpoonup o_5\}$	1.0
$\{o_4\}$	$\longrightarrow \{a_2\}$	0.3
$\{\rightharpoonup o_4\}$	$\longrightarrow \{\rightharpoonup a_2\}$	1.0
$\{o_4\}$	$\longrightarrow \{a_2, \rightharpoonup a_2\}$	0.7
$\{o_4, \rightharpoonup o_4\}$	$\longrightarrow \{a_2, \rightharpoonup a_2\}$	1.0

where f is a mapping function, propagating the evidences from sensor s_a to object o_b. For objects o_1, o_2, o_3 and o_4 in Fig. 2, their masses are computed as:

$$\begin{array}{lllllll} \Gamma : m(o_1) & = & m^r(s_1) & = & 0.98 \\ \Gamma : m(o_1, \rightharpoonup o_1) & = & m^r(s_1, \rightharpoonup s_1) & = & 0.02 \\ \Gamma : m(o_2) & = & m^r(s_2) & = & 0.98 \\ \Gamma : m(o_2, \rightharpoonup o_2) & = & m^r(s_2, \rightharpoonup s_2) & = & 0.02 \\ \Gamma : m(\rightharpoonup o_3) & = & m^r(s_3) & = & 0.9 \\ \Gamma : m(o_3, \rightharpoonup o_3) & = & m^r(s_3, \rightharpoonup s_3) & = & 0.1 \\ \Gamma : m(o_4) & = & m^r(s_4) & = & 0.8 \\ \Gamma : m(o_4, \rightharpoonup o_4) & = & m^r(s_4, \rightharpoonup s_4) & = & 0.2. \end{array}$$

In this step, the evidence is propagated to all the sensors and objects. With the help of the mapping function, the masses of sensors are transferred to the objects layer.

4.4. Propagating evidence in the objects layer

This step intends to propagate evidence in the objects layer. We collect evidence from the observations illustrated in Table 1. Note that evidence $\{o_1\} \longrightarrow \{o_5\}$ refers to that a 0.9 confidence about object o_5 when we observe object o_1. We take simple object o_4, deduced object o_5 and composite object o_6 to explain this step.

4.4.1. Calculating masses for the deduced objects: DO ← E

Although the deduced objects are not monitored directly by sensors, evidential mapping is capable of calculating their masses from their evidence. For instance, evidential mapping propagates masses from objects o_1 to o_5 as:

$$\begin{array}{lll} m(\{o_5\}) & = & m(\{o_1\}) * m(\{o_1\} \rightarrow \{o_5\}) \\ & = & 0.882 \\ m(\{o_5, \rightharpoonup o_5\}) & = & m(\{o_1\}) * m(\{o_1\} \rightarrow \{o_5, \rightharpoonup o_5\}) \\ & & + m(\{o_1, \rightharpoonup o_1\}) * m(\{o_1, \rightharpoonup o_1\} \\ & & \rightarrow \{o_5, \rightharpoonup o_5\}) \\ & = & 0.118. \end{array}$$

4.4.2. Calculating masses for the composite objects: CO ← E

This step aims to propagate evidence to the composite objects. This is also achieved by evidential mapping and is much more complex than those of the deduced objects. It includes two substeps: a) calculating masses to the sets of objects that comprise the composite objects, and b) calculating masses of the composite objects. In the first step, CRET computes the masses of these sets using Eq. (6).

$$\begin{array}{lll} \Gamma : m_1(\{o_2, o_3\}) & = m(\{o_2\}) & = 0.98 \\ \Gamma : m_2(\{o_2, o_3\}) & = m(\{o_3\}) & = 0.9 \\ \Gamma : m_1(\{(o_2, o_3), \rightharpoonup (o_2, o_3)\}) & = m(\{o_2, \rightharpoonup o_2\}) & = 0.02 \\ \Gamma : m_2(\{(o_2, o_3), \rightharpoonup (o_2, o_3)\}) & = m(\{o_3, \rightharpoonup o_3\}) & = 0.1. \end{array}$$

Thus, we get different beliefs for observation combinations from two sources, e.g., beliefs of $m_1(\{o_2, o_3\})$ and $m_2(\{o_2, o_3\})$ from readings of sensors s_2 and s_3. How to generate the masses of the

composite objects is an interesting job. We present an aggregation mechanism by using the weighted sum, defined as Eq. (7):

$$m(C) = m_1 \hat{\oplus} m_2 \hat{\oplus} \ldots \hat{\oplus} m_n(C) = \frac{\sum_{i=1}^{n} w_i \cdot m_i}{\sum_{i=1}^{n} m_i}, \quad (7)$$

where C is a subset of Θ, w_i is the weight for $m_i(A)$ and $\sum_{i=1}^{n} w_i$ equals to 1. Let r_i be the discount rate for mass m_i. The weight of w_i is calculated by Eq. (8).

$$w_i = \frac{r_i}{\sum_{i=1}^{n} r_i}. \quad (8)$$

Theorem 1. *The mass m(C) that is calculated by Eq. (7) is a proper mass.*

Proof. A proper mass means that a mass must satisfy all the requirements given by Eq. (1). Because m_i is less than 1, the value of $\frac{\sum_{i=1}^{n} w_i \cdot m_i}{\sum_{i=1}^{n} w_i}$ is between 0 and 1. When C is an empty set, the value of $m(C)$ is equal to 0. Suppose we have l subsets of Θ. The sum of $m(C_j)$ is calculated as $\frac{\sum_{j=1}^{l}\sum_{i=1}^{n} m_{ji}}{\sum_{i=1}^{n} m_i}$ that is equal to $\frac{\sum_{i=1}^{n}\sum_{j=1}^{l} m_{ij}}{\sum_{i=1}^{n} m_i}$. The value of $\sum_{j=1}^{l} m_{ij}$ is m_i so that the sum of all the subsets of Θ equals to 1. Therefore, $m(C)$ is a proper mass. □

In our case, parameters w_2 and w_3 are computed as 0.6 and 0.4, and ϖ equals to the value of $w_2 * m_1(\{o_2, o_3\}) + w_3 * m_2(\{o_2, o_3\})$. Eq. (9) shows the evidential mapping for $m\{(o_6)\}$. In the same way, we get the mass value of $m(\{o_6, \rightharpoonup o_6\})$.

$$m\{(o_6)\} = m(\{o_2, o_3\}) = \frac{\varpi}{w_2 + w_3} = 0.948. \quad (9)$$

4.4.3. Generating an activity candidate set from objects: A ← O

So far, we have generated an activity candidate set in which users may do one or more activities at a time. *Although knowing precisely what the users want is impossible, observations show that most routine tasks are predictable*. In fact, an object-activity mapping table is collected in CRET that conforms closely to the reality of how users work. This mapping table varies with each scenario and can be collected from the observations. For Fig. 2, activity a_1 obtains the masses propagated by objects o_5 and o_6 according to the object-activity mapping table. Then its mass is calculated as:

$$\begin{array}{lcll} m_1(\{a_1\}) & = & m(\{o_5\}) & = 0.882 \\ m_2(\{a_1\}) & = & m(\{o_6\}) & = 0.948 \\ m_1(\{a_1, \rightharpoonup a_1\}) & = & m(\{o_5, \rightharpoonup o_5\}) & = 0.118 \\ m_2(\{a_1, \rightharpoonup a_1\}) & = & m(\{o_6, \rightharpoonup o_6\}) & = 0.052. \end{array}$$

For the same reason, the mass of activity a_2 is calculated as Eq. (10). At this point, CRET has generated an activity candidate set and is ready for user activity recognition in the next step.

$$\begin{array}{lcl} m_{a_2}(\{a_2\}) & = & m_{o_4}(\{o_4\}) * m(\{o_4\} \rightarrow \{a_2\}) \\ & = & 0.24 \\ m_{a_2}(\{a_2, \rightharpoonup a_2\}) & = & m_{o_4}(\{o_4\}) * \\ & & * m(\{o_4\} \rightarrow \{a_2, \rightharpoonup a_2\}) + \\ & & + m_{o_4}(\{o_4, \rightharpoonup o_4\}) * \\ & & * m(\{o_4, \rightharpoonup o_4\} \rightarrow \{a_2, \rightharpoonup a_2\}) \\ & = & 0.76. \end{array} \quad (10)$$

4.5. Recognizing user activities: $a_i \leftarrow A$

This step focuses on recognizing user activities from the activity candidate set. For example, activity a_1 is chosen because of its bigger mass than that of other activities. Observations in previous steps reveal that the original accuracy of sensors has a significant impact on the final results. Two methods are available to avoid the influence of sensor reliability. One method is to use powerful sensors to improve accuracy, but this increases cost and prevents pervasive computing. The other method is to use multiple sensors to monitor the same objects or sample the same context several times. In most cases, the latter method is preferred for pervasive applications because it inflicts less cost.

Table 2
An example of activity candidate set.

Activity	m_1	m_2	$\underline{m}$- Final result
$\{a_1\}$	0.40	0.20	?
$\{a_2\}$	0.30	0.20	?
$\{a_3\}$	0.20	0.30	?
$\{a_1, a_2\}$	0.05	0.15	?
$\{a_2, a_3\}$	0.04	0.10	?
$\Theta = \{a_1, a_2, a_3\}$	0.01	0.05	?

To describe clearly the mechanism of activity recognition, we extend the example used in Section 4.4.3. Suppose that we collect the evidence for the same scenario and process it by the steps mentioned above (see Table 2). According to the Eq. (4), the normalizing constant K is calculated as

$$\begin{aligned} K &= 1 - \sum_{B \cap C = \emptyset} m_1\{B\} \cdot m_2\{C\} \\ &= 1 - \{m_1\{a_1\} \cdot m_2\{a_2\} + \cdots \\ &\quad + m_1\{a_2, a_3\} \cdot m_2\{a_1\}\} \\ &= 1 - \{0.4 * 0.2 + \cdots + 0.04 * 0.2\} \\ &= 0.477. \end{aligned}$$

Note that the calculation of the normalizing constant is computed over the entire set of evidence. When the amount of evidence and sources increase, Dempster's rule requires exponential time to calculate the normalizing constant. Existing work [18] has proved that the complexity of Dempster's rule is NP-complete. Moreover, when some evidence is zero, Dempster's rule will get wrong results, which is well-known as the Zadeh paradox. According to the Eq. (3), the combination mass of activity a_1 is computed as

$$\underline{m}\{a_1\} = m_1 \oplus m_2(\{a_1\}) = \frac{\sum_{B \cap C = a_1} m_1(B) \cdot m_2(C)}{K} = 0.3606.$$

In the same way, we calculate the combined masses $\underline{m}\{a_2\}$, $\underline{m}\{a_3\}$, $\underline{m}\{a_1, a_2\}$, $\underline{m}\{a_2, a_3\}$ and $\underline{m}\{\Theta\}$ as 0.3795, 0.2201, 0.0241, 0.0147 and 0.0010, respectively. After all the combination masses have been obtained, applications will compute the beliefs using Eq. (2) and then recognize the activities. Correspondingly, activity a_2 in Table 2 is selected as the prediction. Up to now, all the steps of applying CRET to pervasive computing environments have been described. However, two problems are still not solved — the intensive computation that arises from calculating the normalizing constant and combined masses [16], and the Zadeh paradox that emerges from conflicting evidences [20].

4.6. Evidence selection and conflict resolution strategies

In order to solve the problems of evidence theory, we present two strategies (i.e., evidence selection and conflict resolution to the Zadeh paradox).

4.6.1. Evidence selection strategy

The calculation for the normalizing constant K in Dempster's rule is time-consuming because it requires traversing all the

evidence and masses. We find that an observation illustrated as Lemma 1 can reduce the operations for the normalizing constant to $|\Theta|$. Note that Dempster's rule, which uses the Eq. (4) to calculate the normalizing constant, is appropriate for calculating the combined mass for any one hypothesis without calculating all the combined masses.

Lemma 1. *The normalizing constant K can be computed from all the masses that are to be normalized.*

Proof. Let $m_1, m_2, \ldots, m_n$ be n masses that are to be normalized. Given that the normalizing process is to normalize all the masses, the normalizing constant K must equal to $\sum_{i=1}^{n} m_i$. □

The other computation overhead of CRET emerges from the calculation of the combined masses. Suppose we calculate the combined mass for the evidence C by calculating its unnormalized mass and then normalizing it. The total number of operations required is proportional to $log(|C|) + log(|\Theta|)$ for ordered lists, but is exponential in $|\Theta|$ for n-dimensional array representation. Because the decisions made by pervasive applications, in most cases, rely on the most related contexts, CRET employs a process of selecting evidence. This process consists of two substeps.

First, CRET defines the importance of evidences by t_i that is given as:

$$t_i = \sum_{j=1}^{n} w_j \cdot m_j, \tag{11}$$

where j is the number of masses and w_j is the weight of mass m_j for evidence i. Consider that the masses from multiple sources have different impact on the decision-making for pervasive applications. CRET is capable of tuning w_j to emphasize the most important masses.

Second, CRET selects the most related evidence by $k-l$ strategy. Let n be the amount of evidence, k be the minimum amount of evidence to be kept, l be the maximum amount of evidence to be kept, η be the threshold on the sum of masses of selected evidence, Σ be the sum of masses of selected evidence, ϑ be the number of selected evidence and *LSP* be the least small probability event. Lines from 5 to 17 in Algorithm 1 are $k - l$ selection strategy for evidence selection. When l is equal to k, the selection strategy turns into the top K selection strategy. Note that the step of randomized quicksort in the $k - l$ strategy is just marking the order of evidence according to the t_i and does not swap evidence orders during the sorting process.

Fig. 3. A well-known example of the Zadeh paradox.

Fig. 4. Introducing conflicting factors for the Zadeh paradox.

4.6.2. Conflict resolution to Zadeh paradox

The Zadeh paradox refers to the situation that Dempster's rule becomes problematic when combining conflicting evidence whose sum masses are zero [20]. Fig. 3 illustrates a well-known example of the Zadeh paradox, where activity a_2 is a low probability event supported by little evidence, while activities a_1 and a_3 are strongly supported by some evidence. Dempster's rule, however, implies that activity a_2 with belief as 1 should be absolutely believed, and activities a_1 and a_3 get the beliefs as 0. This result is counter-intuitive. In pervasive computing environments, the evidence is often highly imprecise and incomplete and thus the Zadeh paradox is more likely to happen. In our experiments, we find that the Zadeh paradox has a great impact on the performance of CRET.

According to [17,37], all the masses are conditional beliefs (i.e., $m(x|e) = m(x)$) and thus evidence are partially believed. In essence, it is the masses of activities a_1 and a_3 that cause the unsatisfactory results. In order to remove the Zadeh paradox, we introduce conflicting factors. Fig. 4 illustrates the assignment of conflicting factors where they are added to the FOD, which affects the degree of beliefs for all hypotheses.

Definition 1. *Let ε be the measurement for the Zadeh paradox, and set its value equal to the minimum combined mass of the evidence.*

Definition 1 shows the measurement for the Zadeh paradox. The less the ε, the better the performance of solving the Zadeh paradox. Theorem 2 proves that conflicting factors can reduce the influence of the Zadeh paradox.

Theorem 2. *Conflicting factors can decrease the value of ε for that well-known example of the Zadeh paradox.*

Proof. In Fig. 3, the combined mass $\underline{m}\{a_1\}$ is $\frac{0.99-\delta_2}{1-\delta_2+(\delta_2+0.0001)/\delta_1}$ according to the Dempster's rule. Given a specific δ_2, we infer that mass $\underline{m}\{a_1\}$ is monotonic increasing. For the same reason, we infer that the combined mass $\underline{m}\{a_3\}$ is monotonic increasing. According to the principle that the sum of all combined masses is a sum up to 1, the combined mass for activity a_2 decreases. That means measurement ε becomes smaller. Thus, conflicting factors reduce the influence of the Zadeh paradox. □