2022 IEEE/ACM Seventh International Conference on Internet-of-Things Design and Implementation (IoTDI) | 978-1-6654-9641-4/22/$31.00 ©2022 IEEE | DOI: 10.1109/IOTDI54339.2022.00012

CoPI: Enabling Probabilistic Conflict Prediction in Smart Space Through Context-awareness

Jie Hua[†], Haoxiang Yu[†], Sangsu Lee[†], Hamim Md Adal[‡],Colin Milhaupt[‡], Gruia-Catalin Roman[‡], Christine Julien[†]
[†]Department of Electrical and Computer Engineering, University of Texas at Austin,
{mich94hj, hxyu, sethlee, c.julien}@utexas.edu
[‡]Department of Computer Science, University of New Mexico,
{hmdadal, cmilhaupt, gcroman}@unm.edu

***Abstract*—In a smart space influenced by multiple parties, conflicts can arise when competing users try to control the same devices in different ways. Such conflicts usually require user negotiation to resolve and thus lower people's satisfaction and trust in the smart system. Finding a conflict is the first step to resolving it, and the timing when a conflict is identified impacts the options for resolution. Most existing approaches identify conflicts only at the time they occur, which offers little help to the users in resolving the conflicts, especially without them having to compromise. A better solution is to predict potential conflicts in advance so that the users can coordinate themselves to avoid conflict situations beforehand. In this paper, we propose a novel context-aware conflict prediction framework that addresses the research gaps identified in existing literature. We mine habit patterns from the user's previous interactions with smart devices in the various environments they occupy. These habits serve as inputs to our conflict prediction algorithm which takes the habits of pairs of users and outputs context situations in which those users have the potential to conflict. To support eventual flexible resolution, we use explicit models of the uncertainties of users' behaviors to associate each potential conflict scenario with a probability of that conflict occurring for these particular users. We evaluate our framework on real-world datasets to demonstrate the effectiveness of the proposed approach.**

***Index Terms*—Internet of Things(IoT), conflict prediction, smart space**

I. Introduction

In an ambient intelligence (AmI) world, devices are interconnected and distributed in the environment to proactively provide services that are aware of users' presence and preferences. Most AmI environments are under the influence of multiple people simultaneously, whether because they support multiple cohabiting occupants or because both an occupant and a building manager have designs on how a space should be used. In supporting the potentially competing interests of multiple parties in a particular AmI system, one of the most important challenges is managing *conflicts* [1], [2].

We define a conflict as *a situation when two parties' preferences over the state of an IoT device differ*[1]. For example, an AmI system may receive a command "turn on the light" from one user as they enter the space. Meanwhile, the devices may have already been maintaining the policy "keep the illumination level low" in support of a user who was already present. As another example, a user may instruct an AmI space to heat the environment to $21°C$, but the building manager may restrict the use of radiant heating until the first of October. Such conflicts, if not handled, cause users to lose confidence in smart spaces and stop using them. On one hand, these conflicts are difficult for a system to handle automatically because they usually require negotiations between users. On the other hand, negotiation between users, automated or otherwise, requires support for pre-emptively identifying potentially conflicting behaviors so as to minimize the negative impacts.

The need to identify and resolve conflicts in AmI has, of course, not gone unnoticed. While resolving conflicts is an ultimate goal, finding conflicts is a first step, and the timing of identifying a conflict impacts the options for its resolution. If a conflict is only detected after it occurs [3], [4], resolving it can only "repair" the situation—the users have already been interrupted. Some efforts seek to detect conflicts that are about to happen [5]. Before executing a command or changing a device's state, these approaches verify the effect of the action against some specific policies or rules. Although they can resolve a conflict by avoiding the situation (i.e., rejecting a command that raises a conflict), they achieve a win-lose resolution as one user must yield. To achieve a win-win, a smart system must allow sufficient time to negotiate an acceptable resolution either through automation or explicit user intervention. Consider a case when two users need a room for a conference call—detecting the situation after the first user's call has started requires the second user to compromise. Instead, if a smart system can identify such a situation ahead of time, it could automatically adjust the users' calendars. We use "a priori" conflict prediction to refer to the act of identifying a conflict before either party has engaged in an activity that could cause the conflict. While existing frameworks [6], [7] achieve conflict prediction, their approaches lack flexibility and can therefore lower users' satisfaction. In particular, we identify three research gaps that this work fills:

- **Lack of Context-awareness**: Context is any ambient information that captures the shared state of users in a smart environment (e.g., time, location, or weather). Capturing conflicts with associated context enables flexible resolu-

[1]We assume a conflict involving more than two parties can be reduced to multiple pairwise conflicts.

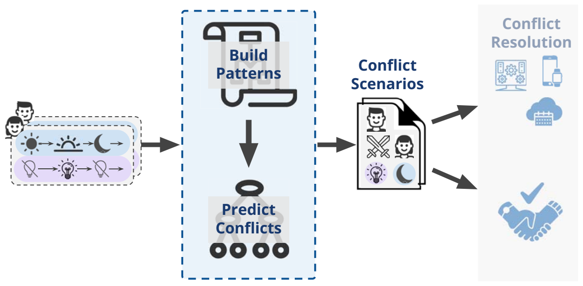


Fig. 1: We learn from users' previous interactions with the devices to predict potential conflict scenarios. Though the resolution of conflict is out of our scope, we keep it mind when designing our framework.

tion. For example in smart home control, maintaining the ambient temperature may or may not conflict with a desire for fresh outdoor air, depending on the weather. The key observation is that *the same conflict might occur differently in different contexts*, thus conflicts should be reported alongside a contextual description. Most existing frameworks either do not support context-awareness [8] or consider only time as the context of a conflict [6].

- **Determinism**: Existing frameworks treat conflicts deterministically [2], [7], that is a conflict either exists, with 100% certainty, or it does not occur at all. However, due to the dynamics of user's behaviors, predicted conflicts are not guaranteed to happen in all circumstances. Thus predicting conflicts with associated likelihood (probability) of occurring not only provides more information for flexible resolution but also faithfully reflects the severity of the identified conflicts. While this benefit of probabilistic conflict prediction is noted in other domains [9], no existing work in AmI clearly outputs conflict situations with associated probabilities.
- **User Overhead**: High user cognitive load is another shortcoming of existing frameworks. Rule- or policy-based solutions [1], [10] predict conflicts by parsing and reasoning about a given set of policies to find conflicts. Such an approach cannot generalize across different AmI spaces as it requires the user to express their intentions.

To the best of our knowledge, no existing framework explicitly accounts for all three of these aspects in predicting conflicts. We propose CoPI: a novel data-driven framework for a priori **Co**nflict **P**rediction for **I**oT devices. We target spaces equipped with smart devices (e.g., thermostats, televisions and lighting) and controlled by multiple — potentially conflicting — parties. We focus on repetitive conflicts rooted in differences in daily habits which are commonly present in our daily lives but neglected by many [11]. Fig. 1 shows our approach. The input are traces of user-device interaction histories, tagged with contextual data, which can be collected in modern sensor networks with minimal user involvement [12], [13]. We first build habit patterns for each user from these interaction histories. Although pattern mining is well-studied, existing work is undertaken in different problem settings and cannot be applied to probabilistic conflict prediction directly [14]–[17]. Thus we propose a novel pattern mining approach specifically designed to account for context and non-determinism. The second part of our framework is predicting conflicts among mined habit patterns. We find overlapping scenarios across users and compute the probability of a conflict happening under the identified scenarios. We generate conflict scenarios with associated context descriptions that can not only be consumed by automated conflict resolution systems but also can be understood by users with no particular technical expertise.

II. Related Work

We start this review by introducing existing definitions and categorizations for conflicts in AmI and placing CoPI within these existing taxonomies. We then examine existing conflict management frameworks to identify the research gaps. We conclude this section by looking at related efforts that address a key aspect of our approach: routine habit mining.

A. Conflict Categories

Conflicts in smart environments are usually defined based on smart devices. Since actions of devices can influence the characteristics of the space (e.g., temperature, illumination), some efforts [4], [18] define conflicts as situations when one service opposes another one in changing the value of a spatial characteristics. Conflicts can also occur between the user and the system when the user misuses or misunderstands the smart control system [19], [20]. These definitions represent efforts that are parallel to ours, as we focus on conflicts resulting from contradicting requests from two users for the same device.

Resendes *et al.* [21] conducted an extensive review of conflicts in smart environments, which resulted in a taxonomy of conflicts with four dimensions:

- Source: what is the entity the conflict arises over, e.g., the use of device, the ownership of a resource;
- Intervenients: who are the parties having the conflict, e.g., two occupants, an occupant and a building manager;
- Time of detection: when does detection happen, e.g., before or after the occurrence of the conflict;
- Solvability: can the detected conflict be solved by the system or does it require user intervention.

Following this taxonomy, the sources of conflicts for our problem are the states of devices; the intervenients are two users; and the time of detection is *a priori*, i.e., sufficiently in advance of the conflict that preventive action can be taken. We do not consider solvability in this work, and defer system for conflict resolution to future works.

B. Conflict Management

Conflict management is an important topic in many areas. Aircraft systems are usually equipped with robust probabilistic conflict prediction models based on the trajectories and geometry [9], [22]. Despite that their algorithms cannot be applied to our problem, they articulate the need for early resolution and probabilistic prediction to reduce the cost of conflict. Similarly, merge conflict prediction [23] in source control also benefits from associating conflicts with meta-information. In terms of managing conflicts in smart spaces, many efforts support detection only at the time of occurrence [24]–[26]. Some frameworks [4], [5] require a model of the space's ambient characteristics and knowledge about how devices influence those characteristics, and this model must be provided *a priori* using a standard language, which limits adoption. Context-awareness is also proposed in other works [27], [28] to facilitate the detection and resolution of conflicts. However, instead of computing patterns over contextual information, they require the users to manually input reasons or conditions of their behaviors to conduct context reasoning. We argue that such frameworks incur significant user overhead and are not generalizable across applications. IoTC^2 [29], and similar IoT middlewares [30] evaluates any commands from the controllers before sending them to the device to execute by checking if they conflict with other issued commands at run-time. This causes delay in detecting conflicts and limits the options for resolution.

Install-time conflict prediction is also broached in other efforts. Most of these efforts focus on solving the problem in a rule-based environment where actions and behaviors of devices are programmed to follow a set of rules [7], [31]. For instance, Shah *et al.* [10] adopts IF-THEN rules that detect conflicts with disjunctive normal form. Similarly, Sun *et al.* [1] proposed a framework to detect conflicts based on rules in XML format. These works detect conflicts among a static set of rules and assume the behavior of a AmI system is fully controlled by those rules. While their contributions focus on providing formal reasoning tools to detect conflicts deterministically, we adopt a more dynamic view and acknowledge uncertainties in users' preferences. We argue that the behavior of the devices, controlled by a user instead of static rules, cannot be predicted with 100% certainty and so are the conflicts. A more recent work [6] shares this idea by modeling uncertainties in the device usage habit for each user. However, their framework supports time as the only context and does not provide probability for predicted conflicts.

C. Routine Habit Mining

The essential input to conflict prediction is some representation of humans' interactions in the space. To reduce human burdens and achieve better feasibility, these human habits should be learned from data instead of required from the users.

Learning human habits and preferences is an important aspect of AmI [32] and is well studied by many researchers [33]–[35]. Although existing approaches have compelling results, they cannot be applied directly for conflict prediction. Many of them focus on activity recognition schemes [14], [36], [37] that employ various classification models. They predict activities based on various contextual input from sensor readings but they do not output frequent habit patterns suitable for further analysis. Some efforts focus on discovering transitions between activities and thus predict the next action from previous activities [38]–[40] which are parallel to our target. Association rule mining (ARM) is another widely adopted approach to find relations between context, activities and smart devices [15], [16], [41]. Common algorithms for ARM like the Apriori algorithm [42] and FP-growth [43] mine frequent patterns and associate each with a confidence level. However, the general versions of them target discrete items. In our cases, the "item" to predict is the state of the device which can be continuous and thus makes them unsuitable for our application.

III. Problem Model

In this section, we provide more details about the scope of the problem we seek to solve and define several terms to support our problem definition. We then introduce CoPI's view of the problem and how we approach it at a high level.

In CoPI, we predict conflicts at the device level. If one user wants the bedroom light on to read, but another wants the bedroom light off to sleep, this is a conflict. If one user wants the television on but another user wants the television off so they can read, this is a conflict. However, if one user wants the television on and another user wants the radio on in the same room, we do not explicitly capture this as a conflict. However, such a conflict can be discovered because a user who has the habit of turning the television on, given some context, also has the habit of requesting the radio to be off in that same context. This example not only showcases the potential of CoPI but also showcases the complexity of relying on humans to explicitly enumerate rules or policies that capture their habits or behaviors like in existing solutions [10]. We focus on conflicts between two users and assume a user never conflicts with themselves. We next provide some key concepts which lead to our problem definition.

A. Key concepts

IoT device. An IoT device, d, provides a function to the users of the space in which it is situated. Each device can have a *state*, which is characterized by one or more attributes. Each attribute is a key-value pair that includes the name of the attribute and its value, which can be either categorical or numerical. Every device has a distinguished categorical attribute named "status" that takes one of two values: "on" or "off". Formally, the state of an IoT device d at time t is:

$$ST_{d,t} = \{\langle a_i, v_i \rangle\} \tag{1}$$

where a_i is the name of the i^{th} attribute, and v_i is the attribute's value at time t. For a categorical attribute, v_i is an element in F_{d,a_i} which is the set of all possible values for this attribute. For a numerical attribute, we assume v_i is a real number. We also define a set A_d, which stores the names of all attributes that the IoT device d can exhibit.

We identify two types of IoT devices: i) *shareable devices* (e.g., thermostat, light, TV, radio), over which two users may simultaneously set the device to the same state, which does not result in a conflict; and ii) *non-shareable devices* (e.g., kitchen burner, computer, microwave), for which any simultaneous use is considered a conflict. For simplicity, we assume that devices labeled *non-shareable* have states that are dedicated to a specific user (e.g., rather than setting the burner state to "cooking", a user u sets the state to "cooking for u"). In this way, we can proceed with just a single definition of conflict that relies on the state of a given device.

User requests. In our problem model, users specify desired states for the IoT devices in their spaces. For any IoT device a user desires to influence, the user can specify the desired attributes by providing a device state $ST'_{d,t}$ which represents the user's preferences for the state of device d at time t. Here the "off" value for attribute "status" is a special case. Sometimes a person turns off a device because they no longer need the device and other people can use it freely. In other cases, however, one may turn off a device because they really need the device to be off (e.g., turning off the television in order to study). Thus, within a device state $ST'_{d,t}$ that captures a user's preferences for the state of device d, we include the possibility of a "don't care" value for the "status" attribute to capture the former situation and reserve the "off" value to indicate the latter. Practically, the "don't care" value can be provided by the user explicitly or it can be captured implicitly (e.g., a user's previous explicit request expires after a timeout or a significant change in the context).

Conflict. When two users share a space but desire the space's devices to be in different states, this results in a conflict. Conflicts are identified at the level of attributes (i.e., when the users specify incompatible values of a device's attributes). More formally, at a given time t, two users request the state of a device d to be $ST'_{d,t} = \{\langle a_i, v_i \rangle\}$ and $ST''_{d,t} = \{\langle a_j, v_j \rangle\}$ respectively. A conflict arises if $\exists a_i = a_j$, such that $v_i \neq v_j$ when a_i is a categorical attribute or $|v_i - v_j| > \mathcal{T}$ when a_i is a numerical attribute. We rely on a user-defined threshold $\mathcal{T}$ for numerical attribute because when users specify values that are close to each other, this may not be a perceived conflict (e.g., one user wants the thermostat set point to be $22°C$ while another user wants it to be $22.5°C$).

Context. Users' interactions in IoT enabled spaces are heavily influenced by the context. Context is any information that can be used to characterize the situation of an entity [44]. In smart spaces, users' requests for changes in the states of IoT devices can be directly tied back to the ambient context. For instance, changes to the state of smart lights can be influenced by the time of day or the weather; changes to the state of the thermostat can be influenced by the temperature, etc. We define a *context snapshot* as a set of key-value pairs $C_t = \{\langle c_i, x_i \rangle\}$, where c_i is the name of a context type, and x_i is its value at time t, which can be either a categorical value or a numerical one, depending on the particular context. Different spaces may be differently capable of capturing various measures of context, so there is no guarantee that all aspects of the context may be captured in all spaces.

Interaction histories. Users interact with IoT devices in a variety of everyday spaces, and these interactions give clues about how the user prefers their spaces to be configured. Conceptually, an *interaction history* captures the users' continuous interactions with their surroundings and the context in which those interactions occurred. Formally, for every device d, a function $h_d(t) = (C_t, ST'_{d,t})$ maps an instant in time, t to the context snapshot C_t and the user's desired device state $ST'_{d,t}$ that the user requests at time t. Practically, an interaction history can be collected by recording the changes in the device states and the context using modern sensor networks [12], [13]. For practicality, we represent a user u's interaction history as a sequence of tuples $H_u = (\ldots, \langle C_t, ST'_{d,t} \rangle, \ldots)$ where each tuple indicates either a change in context, captured in a delta of C_t relative to C_{t-1} or a change in the desired state of a single device d, captured in a difference between $ST'_{d,t}$ compared to the previous device state $ST'_{d,t-1}$. We assume that changes in devices' states are atomic and serializable.

Problem definition. From these definitions, we can now explicitly state the problem we seek to solve. In particular, we seek to identify the conflicts that a user might experience when they enter a new IoT space with other users. We term this space the *target space*. Our problem definition is:

> *Given context-tagged interaction histories for two users, predict the contexts in which these two users may conflict over an IoT device in the* target space *and the probability with which that conflict arises.*

B. The CoPI Perspective

CoPI's task is to learn from users' interaction histories to predict the potential for conflicts over the use of IoT devices in a target space. However, since the target space is a new space with different devices, we cannot learn directly from interactions with the target devices. Thus to use an interaction history, a user (or a user delegate) selects a single device from the histories to serve as a reference for a device in the target space. For each device d in the target space, user u can choose to specify a source device d^u from the interaction history to represent how the user would interact with device d in the target space; we write such a mapping as $d^u \rightarrow d$. If a user chooses not to specify a source device for a device in the target space, CoPI simply will not predict conflicts on that device for this user. For simplicity, we will use notation d to refer to a device in the target space and we use d^u to refer to the corresponding device from the interaction histories of user u. We create the set D_u to contain the devices in the target space for which user u provides a mapping (i.e., $D_u = \{d : d^u \rightarrow d \text{ exists}\}$); CoPI will only predict conflicts for user u for devices in D_u. Since d and d^u may support different attributes, our mapping considers only attributes that are present in both d and d^u.

When predicting conflicts, CoPI creates a description of a *conflict scenario*, that associates a characterization of the conflict with a description of the context in which that conflict may occur. The conflict itself is captured by the tuple $\langle d, a \rangle$, where d is the device that the two users conflict over and a is the specific attribute for which the two users have conflicting values. A conflict that involves more than one attribute for a given device is captured as multiple separate conflict tuples, one for each conflicting attribute. Characterizing the context in which a conflict occurs is somewhat more difficulty. For starters, some context values are continuous, and therefore attempting to capture discrete characterizations of the contexts in which a conflict might occur would result in infinite

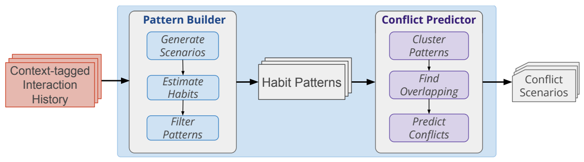


Fig. 2: Overview of CoPI

numbers of conflict scenarios. To handle this complexity, in CoPI, we define a *context range* to contain a potentially infinite set of contextual situations for a single context type.

Definition 1. (context range $\underline{R}$) A context range R_c represents a set of values for the context type with the name c. If c is a numerical context, $R_c = [x^L, x^U)$ meaning that any value between x^L and x^U is included (i.e., $\forall x^L \leq x < x^U, x \in R_c$). Note that we use an open range because the upper bound x^U is not included. For a categorical context, $R_c = Q$ where Q is a finite set of categorical values (i.e., $\forall x \in Q, x \in R_c$). We also define the domain of a context c as $\hat{R_c}$; and we assume a known maximum upper bound and minimum lower bound for numerical context and a finite domain for categorical values.

The contexts of a user's interactions are captured by multiple context types, which we term a *context scenario*:

Definition 2. (context scenario $\underline{CO}$) A context scenario is captured by a set of context ranges, one for each type of context i.e., $CO = \{\langle c_i, R_{c_i}\rangle\}$. Each context type c_i appears at most once in CO.

As an example, the context scenario described by the text "Monday 2 pm to 3 pm when raining" is written as:

$$\begin{aligned}\{ \quad & \langle dayOfWeek, \{\text{"Monday"}\}\rangle, \\ & \langle hourOfDay, [14, 15)\rangle, \\ & \langle weather, \{\text{"rain"}\}\rangle \quad \}\end{aligned}$$

We now define a *conflict scenario* as:

Definition 3. (conflict scenario $\underline{CS}$) A conflict scenario describes a situation where an IoT device conflict for the two users may arise. It is represented as:

$$CS = \langle\langle d, a\rangle, CO, p\rangle \tag{2}$$

where $\langle d, a\rangle$ is the conflict tuple, CO is a context scenario in which this conflict might happen, and p is the probability of the conflict happening; when $p = 0$ no conflict exists.

IV. Conflict Prediction Framework

In this section, we present our framework for probabilistic conflict prediction. We propose a data-driven approach as shown in Fig. 2. The input to CoPI is the context-tagged interaction history for each user (H_u). CoPI comprises two major components. The *pattern builder* processes the records in the interaction histories and mines each user's habit patterns. In particular, for each user-device pair, the pattern builder identifies a set of patterns representing the user's preferences of the state of that device under different contexts. The second component, the *conflict predictor*, takes all of the mined patterns of two users and finds conflict scenarios among them. Ultimately, CoPI outputs the predicted conflict scenarios, which can subsequently inform the users to adjust their interactions or to enable automatic conflict resolution.

A. Building Habit Patterns

To predict conflicts, we need a systematic way to represent and compare users' preferences over devices in the target spaces. When modeling human habit, most existing work [15] employs classification approaches and predicts a subsequent action the user will take. Such solutions do not apply to our problem of probabilistic conflict prediction as they do not provide a probability distribution for all possible actions. A key innovation of CoPI is that we explicitly consider the uncertainty in human interactions with IoT-enabled environments by including fuzziness when building habit patterns. Specifically, we define a notion of a *fuzzy device state* to capture the fact that users do not always interact with the same devices in exactly the same ways, even in the same contexts.

Definition 4. (fuzzy device state $\underline{FST}$) A fuzzy device state FST_d describes an uncertain state for device d. Compared to a normal device state $ST_{d,t} = \{\langle a_i, v_i\rangle\}$, a FST_d captures a *probability* associated with the user's preferences for a device d. Specifically, $FST_d = \{\langle a_i, \widetilde{v_i}\rangle\}$ where $\widetilde{v_i}$ represents the probability distribution of v_i. For a categorical attribute, $\widetilde{v_i} = \{\langle f_{i,j}, p_{i,j}\rangle | f_{i,j} \in F_{d,a_i}\}$ represents the likelihood of $f_{i,j}$ being the categorical value for attribute a_i is $p_{i,j}$. For a numerical attribute, $\widetilde{v_i} = \langle \mu_i, \sigma_i^2\rangle$ represents the mean and variance of the numerical value. We model the numerical attribute as a normal distribution: $v_i \sim \mathcal{N}(\mu_i, \sigma_i^2)$.

Based on this, we define the *habit pattern* for a user, which associates these fuzzy device states with the context scenarios in which they occur. In particular:

Definition 5. (Habit Pattern $\underline{HP}$) A habit pattern $HP_d = \langle CO, FST_d\rangle$ captures the distribution of the user's habits relative to a device d in the target space, given a concrete context scenario (CO). HP_d can be interpreted as conveying that the user's preference over the state of d follows the distribution described by FST_d.

A habit pattern connects a single context scenario to a single fuzzy device state. For instance, a habit pattern may capture the distribution of the likelihood that the user turns on a specific light in the target space when the context scenario is "Monday between 2pm and 3pm when raining." A *habit* relative to a specific device d in the target space, therefore, consists of a set of such patterns. In CoPI, we construct a habit for each device in the target space. To build habit patterns, CoPI must estimate the fuzzy device states that correspond to various encountered context scenarios using the samples in a user's interaction history. Thus, as shown in Fig. 2, the first step in the pattern builder is to generate potential context scenarios for candidates of habit patterns. Then we group the device states in the interaction histories based on the potential scenarios to estimate the fuzzy device states in the candidate habit patterns. At last we remove candidates that appear infrequently in the interaction histories based on a system-defined threshold.

Generating context scenarios. To build habit patterns, we first need to generate a discrete set of context scenarios. For each user u, we consider all context types that are available for u in the interaction histories and their corresponding domain $\hat{R}_c$. As described previously, CoPI operates over a multi-dimensional definition of context, where each type of context defines one dimension. To make the problem of working with continuous context domains tractable, we divide each context dimension into m_c equal-sized slots. For a categorical context type (e.g., *dayOfWeek*), each slot contains a possible value for the context (e.g., { "Monday", "Tuesday", ...}) and thus m_c equals to the size of domain of c. For a numerical context type (e.g., *hourOfDay*), m_c is determined based on the range of the context and the desired granularity (e.g., if $m_c = 12$, then a slot is created for every two hours). In general, a smaller m_c captures the user's preferences more accurately but requires more detailed interaction histories to succeed. In Section V, we evaluate how the choice of m_c impacts CoPI's performance.

Formally, the domain $\hat{R}_c$ for a context c is divided by m_c into a set of ranges $\{R_{c,1}, \ldots, R_{c,i}, \ldots, R_{c,m_c}\}$. For a categorical context type:

$$R_{c,i} = \{q_i : \forall q_i \in \hat{R}_c\} \tag{3}$$

In contrast, for a numerical context type:

$$R_{c,i} = \left[x^L + (i-1) \cdot \frac{x^U - x^L}{m_c},\ x^L + i \cdot \frac{x^U - x^L}{m_c}\right) \tag{4}$$

where x^L and x^U are the min and max values for context c. After the split, we can generate the potential context scenarios. As defined in Definition 3, a context scenario is represented by a set of context ranges. Thus each potential scenario is a unique combination of context ranges by selecting one from the set $\{R_{c,i}\}$ for every context type c. For instance, if a user has two categorical context (*weather*, *dayOfWeek*) and one numerical context (*hourOfDay*) with a m_c of 12, then each potential scenario represents a 2 hour block for every possible combination of *weather* and *dayOfWeek*. To give a sense of this set, the total number of context scenarios is $\prod m_c$. In our example, assuming we have 10 possible values for *weather*, then we have $12 \times 10 \times 7$ potential context scenarios in total. We use $\mathcal{CO}_u$ to represent the set of all scenarios for user u. Since the context types available for different users can be different, each $\mathcal{CO}_u$ can be different as well.

Building habit patterns. Algorithm 1 shows the process of building habit patterns. For each user u, we run BUILDPATTERNS for every device $d \in D_u$, where D_u is the set of devices in the target space for which user u has a device d^u mapped from u's interaction histories. Λ captures, for a given device d in the target space, a set of the user's preferred device states for the mapped device d^u in the interaction history. We associate Λ with a specific context scenario, CO; as such, it captures the user's preferences for d^u in the past, any time the context scenario CO appeared in the interaction history.

Algorithm 1: Building habit patterns

```
1  $u, d$: the user and the device being processed
2  $H_u$: the interaction histories for user $u$
3  $\mathcal{CO}_u$: set of candidate context scenarios for user $u$
4  Function BUILDPATTERNS:
5      initialize $cand_d \leftarrow \{\langle CO_i, \Lambda_i = \emptyset\rangle | CO_i \in \mathcal{CO}_u\}$
6      for $\langle C, ST'_{d^u,t}\rangle \in H_u$ do
7          for $CO_i \in \mathcal{CO}_u$ s.t., $contain(CO_i, C)$ do
8              insert $ST'_{d^u,t}$ to $\Lambda_i$
9          end
10     end
11     for $\langle CO_i, \Lambda_i\rangle \in cand_d$ s.t., $|\Lambda_i| > threshold$ do
12         $FST_i \leftarrow \{a_i, \text{GETV}(a_i, \Lambda_i) | a_i \in A_d \cap A_{d^u}\}$
13         $HP_{d,i} \leftarrow \langle CO_i, FST_i\rangle$
14     end
15     $\mathcal{HP}_{u,d} \leftarrow \bigcup_i HP_{d,i}$
16     return $\mathcal{HP}_{u,d}$
17 end
18 Function GETV($a_i, \Lambda_i$):
19     if $a_i$ is categorical then
20         for $f_{i,j} \in F_{d,a_i}$ do
21             $p_{i,j} \leftarrow \frac{|\{ST : ST \in \Lambda_i \wedge \langle a_i, f_{i,j}\rangle \in ST\}|}{|\Lambda_i|}$
22         end
23         $\widetilde{v}_i \leftarrow \{\langle f_{i,j}, p_{i,j}\rangle | f_{i,j} \in F_{d,a_i}\}$
24     else
25         $\mu_i, \sigma_i^2$ are the mean and variance of
             $\{v_i : ST \in \Lambda_i \wedge \langle a_i, v_i\rangle \in ST\}$
26         $\widetilde{v}_i \leftarrow \langle \mu_i, \sigma_i^2\rangle$
27     end
28     return $\widetilde{v}_i$
29 end
```

In Algorithm 1, we initialize a set of candidates for exploration, one for each potential context scenario. Initially, Λ for each candidate is empty; the goal is to fill in the fuzzy device states associated with each potential context scenarios. At the outset (lines 6-10), Algorithm 1 examines each entry in the user's interaction history; because we are building habit patterns for a specific device d in the target space, line 6 selects only interaction history entries that reference d^u, the device from the user's interaction history mapped to the target device d. For every potential context scenario (CO_i) that *contains* the context snapshot C from the interaction history entry, we add the user's preferred state of d^u in that context ($ST'_{d^u,t}$) to Λ_i in the set of candidates. In Definition 1, we defined what it means for a context range R_c to contain a context value x. Here we extend that definition to a context scenario $CO = \{\langle c_i, R_{c_i}\rangle\}$ and a context snapshot $C = \{\langle c_i, x_i\rangle\}$ as:

$$contain(CO, C) = \begin{cases} \text{FALSE}, \text{if } \exists \text{i, s.t. } x_i \notin R_{c_i} \\ \text{TRUE}, \text{otherwise} \end{cases} \tag{5}$$

After we group the relevant samples from the interaction history into the candidate habit patterns (i.e., the tuples

$\langle CO_i, \Lambda_i \rangle$), we then estimate the fuzzy device state (FST) for each context scenario and build the final set, $\mathcal{HP}_{u,d}$ containing all generated habit patterns for user u and device d. Lines 11-14 in Algorithm 1 construct this set of habit patterns. In particular, the algorithm examines each of the candidate patterns $\langle CO_i, \Lambda_i \rangle$ in $cand_d$. The first step is to filter out any candidate patterns that have very few samples in the user's interaction history; for this we use a system-provided *threshold*; if there are fewer samples for a given context scenario than *threshold*, we do not construct a habit pattern for that scenario for device d. This threshold marks the confidence the system has that the samples in the interaction history are representative of the user's actual preferences, given a particular context scenario. A higher value for the threshold results in a higher quality estimation of the user's preferences, but requires a larger interaction history to succeed.

On lines 11-14 of Algorithm 1, we estimate the fuzzy device state for this device and context scenario, given the interaction history samples. Note that not all of the attributes in A_d can be estimated from the samples because some attributes may not referenced in A_{d^u}. Thus, we only consider the attributes that both d and d^u have (line 12). The above process relies on the GETV subroutine, whose behavior in turn depends on whether the attribute for which we are computing the fuzzy device state is a categorical one or a numerical one. For each categorical attribute, we compute the probability distribution $\widetilde{v}_i = \{\langle f_{i,j}, p_{i,j} \rangle | f_{i,j} \in F_{d,a_i}\}$ from the device states in Λ_i that contain attribute a_i. Each probability $p_{i,j}$ is estimated as the number of occurrences of value $f_{i,j}$ in Λ_i divided by the size of Λ_i. For each numerical attribute, we compute the mean and variance $\widetilde{v}_i = \langle \mu_i, \sigma_i^2 \rangle$ for the values in Λ_i of attribute a_i. Once we compute the habit pattern for the context scenario CO_i, we add it to the complete set of habit patterns, which we return at the end of the BUILDPATTERNS function.

B. Predicting conflicts

Conceptually, potential conflicts arise when two users have different habits under the same context. Practically, we find them by checking all pairs of overlapping habit patterns from the two users to determine whether the associate device states are incompatible. As shown in Fig. 2, the CoPI *conflict predictor* has three stages. First, because many of a user's habit patterns represent similar behaviors, CoPI processes these to find clusters of similar patterns and combine them, thus reducing the complexity of the problem and the potential for reporting a large number of redundant potential conflicts to the users. CoPI then compares the clusters of patterns for pairs of users to find instances with the associated context scenarios overlap. Finally, for identified overlapping context scenarios, CoPI predicts the probability of the users desiring devices in the target space to be in different, conflicting states.

Clustering habit patterns. We merge users' habit patterns based on similarities in the user preferences captured in the fuzzy device states for a given device in the target space. Consider two fuzzy device states for a device d for a particular user, $FST_{d,1} = \{\langle a_i, \widetilde{v}_i \rangle\}$, $FST_{d,2} = \{\langle a_j, \widetilde{v}_j \rangle\}$. We define the *distance* (a measure of dissimilarity) of these two states as:

$$dist(FST_{d,1}, FST_{d,1}) = \sum_{a_i = a_j} |\widetilde{v}_i - \widetilde{v}_j|^2$$

$$|\widetilde{v}_i - \widetilde{v}_j| = \begin{cases} \sum_{k=1}^{n_{d,a_i}} |p_{i,k} - p_{j,k}|, \text{ if } a_i \text{ is categorical} \\ \|(\mu_i - \mu_j, \sigma_i^2 - \sigma_j^2)\|, \text{ if } a_i \text{ is numerical} \end{cases}$$

where $\|\cdot\|$ is the L2 norm of the vector. Conceptually, we compute distance by comparing the fuzzy values $\widetilde{v}_i$, $\widetilde{v}_j$ for each attribute that is in both $FST_{d,1}$ and $FST_{d,2}$. As mentioned previously, every FST for user u and device d contains the same attributes from $A_d \cap A_{d^u}$; therefore we only compute the distance for two FST that have the same set of attributes. One problem with the above definition is that for a numerical attribute, μ and σ^2 are not between 0 and 1 like p, which makes the distance function biased. To combat this problem, we normalize the values for each numerical attribute to 0 and 1 when pre-processing the interaction histories.

The problem of clustering habit patterns can thus be formulated as grouping all patterns in $\mathcal{HP}_{u,d}$ into non-overlapping clusters. "Non-overlapping" has two meanings: 1) no habit pattern belongs to two clusters and 2) no context snapshot is contained by the context scenarios of two clusters. The latter requirement ensures that any context snapshot is contained by at most one habit pattern to avoid ambiguity. We define clusters so as to minimize the following loss function, which, intuitively, tries to minimize the distance between patterns in each cluster while having as few clusters as possible:

$$Loss = \sum_{CL} (\alpha + \frac{1}{|CL|} \sum_{HP \in CL} dist(HP, \overline{HP}_{CL})) \quad (6)$$

where each cluster $CL \subseteq \mathcal{HP}_{u,d}$ is a subset of the habit patterns for the same user and device; α is a tunable penalty for each cluster (i.e., a higher α favors fewer clusters); and $\overline{HP}_{CL} = \langle CO_{CL}, FST_{CL} \rangle$ is an average of the patterns in the cluster.

CoPI computes the fuzzy device state FST_{CL} by averaging the values across the cluster. The mean μ_i and variance σ_i^2 for numerical attributes and the probability $p_{i,j}$ for categorical attributes are computed as the average of all corresponding values in the fuzzy device states of the cluster's habit patterns for the same attribute.

The context scenario $CO_{CL} = \{\langle c_i, R_{c_i} \rangle\}$ for a cluster is defined as the bounding box of all the context scenarios in the cluster. Because we assume that the set of context types associated with a given user's habit patterns is static, each context scenario for a given user contains the same set of context types. Intuitively, we define the context scenario for the cluster by creating the union of the ranges for each context type. For categorical context types, we simply union the ranges of categories used (e.g., if one context scenario happens only on Mondays, while another happens only on Tuesdays, the union is { "Monday", "Tuesday" }); for numerical context types, we take as the lower bound the minimum of all of the lower bounds used in the composite context scenarios and the

Algorithm 2: Recursively generating clusters

```
1 Function SPLIT: $CL$
2     $\Psi \leftarrow \{(CL_1, CL_2)$ : all pairs of $CL_1$ and $CL_2$ split $CL$ along every context value$\}$
3     $maxGain \leftarrow \max_{(CL_1,CL_2)\in\Psi}(\mathcal{L}_{CL} - (\mathcal{L}_{CL_1} + \mathcal{L}_{CL_2}))$
4     if $maxGain > 0$ then
5         return SPLIT$(CL_1) \bigcup$ SPLIT$(CL_2)$
6     else
7         return $\{CL\}$
8     end
9 end
```

upper bound is the maximum of all of the composite upper bounds (e.g., for the context type time, if two ranges are 1 pm to 3 pm and 2 pm to 4 pm, the union is 1 pm to 4 pm).

The "non-overlapping" requirement states that no context snapshot is contained in two clusters. To verify this property for categorical context c, given two clusters' context scenarios CO_{CL_1} and CO_{CL_2}, we simply check that the intersection of R_c from CO_{CL_1} with R_c from CO_{CL_2} is empty. To verify this property for a numerical context type c, we check whether x_c^L in CO_{CL_1} is greater than x_c^U in CO_{CL_2} and vice versa (i.e., that the two open ranges are not overlapping).

Given these definitions, Algorithm 2, shows how we compute a set of clusters from a set of habit patterns. We invoke the SPLIT function in Algorithm 2 using an initial input of a single cluster that is equivalent to $\mathcal{HP}_{u,d}$. We then recursively split this single cluster into sets of smaller clusters.

To find the split, we mimic the CART algorithm [45], using context type as the feature for splitting. Here we choose one context type for each split and divide the bounding box for the cluster's context scenario based on the selected context type. For a categorical context type, c, we test all pairs of two subsets that split the range R_c of the bounding box without intersection. For a numerical context type, we split the range based on possible pivot values (i.e., a pivot value x^P splits a range $[x^L, x^U)$ into two ranges $[x^L, x^P)$, $[x^P, x^U)$).

To support the process, we represent the loss function in Equation 6 as $Loss = \sum_{CL} \mathcal{L}_{CL}$; this allows us to compute the *maxGain* for each candidate split. After we generate all possible splits, we compute the loss of the two new clusters in each split $\mathcal{L}_{CL_1}$, $\mathcal{L}_{CL_2}$ and compare them to the original loss $\mathcal{L}_{CL}$ as shown on line 3. If the *maxGain* of the split that maximizes the decrease in loss is greater than 0, we perform the split and recursively find sub-clusters for the new sets (line 5). If the gain is not greater than 0, we return the current cluster as the optimal clustering because no further split can decrease the loss (line 7). For simplicity of writing, we still use $\mathcal{HP}_{u,d} = \{HP_{CL}\}$ to refer to this new set of patterns.

Finding overlapping context scenarios. After clustering, we have the final set of habit patterns for each user-device pair $\mathcal{HP}_{u,d}$. A conflict may arise when two habit patterns for the same device from two users overlap, in particular when two users express preferences for the same device d in the same context. As mentioned before, "overlapping" is defined as the situation when the intersection of the context scenarios of the two habit patterns is not empty. To determine conflict situations, therefore, we first find all pairs of habit patterns HP_1, HP_2 such that $HP_1 \in \mathcal{HP}_{u_1,s}$ and $HP_2 \in \mathcal{HP}_{u_2,s}$, whose context scenarios overlap: $\bigcap\{CO_1, CO_2\} \neq \emptyset$. To compute the intersection, we construct a context range $R_{c,\cap}$ for each context type c that appears in CO_1 or CO_1. If c only appears in one of the two context scenarios, assuming in CO_1, $R_{c,\cap}$ simply equals to the corresponding context range R_c in CO_1. If c appears in both context scenarios, we compute the intersection of the two context ranges. For categorical context types, we intersect the ranges of the categories used (e.g., if one context scenario happens on Mondays and Tuesdays, while another happens on Tuesdays and Wednesdays, the intersection is {"Tuesday"}). For numerical context types, we take as the lower bound the maximum of the two lower bounds in CO_1 and CO_2 and the upper bound is the minimum of the two upper bounds (e.g., for the context type time, if two ranges are 1 pm to 3 pm and 2 pm to 4 pm, the intersection is 2 pm to 3 pm). We use $CO_\cap$ to denote $\bigcap\{CO_1, CO_2\}$.

If device states associated with overlapping context scenarios have different attribute values, this results in a potential conflict that is reported to the users. A novel aspect of CoPI, however, is the ability to report a *probability* of that conflict occurring; we next describe how this probability is computed.

Computing the probability of a conflict. In CoPI, the inconsistency and uncertainty in users' preferences are represented using fuzzy device states. We use these fuzzy device states to compute the probability of conflicts when users' context scenarios overlap. Given two fuzzy device states, $FST_{s,1} = \{\langle a_i, \widetilde{v}_i\rangle\}$ and $FST_{s,2} = \{\langle a_j, \widetilde{v}_j\rangle\}$, for every attribute that is in both states, there is a chance that a conflict may arise due to difference in values for that attribute.

If the attribute in question is a categorical one, the fuzzy values for this attribute are $\widetilde{v}_i = \{\langle f_{i,k}, p_{i,k}\rangle | f_{i,k} \in F_{s,a_i}\}$ and $\widetilde{v}_j = \{\langle f_{j,k}, p_{j,k}\rangle | f_{j,k} \in F_{s,a_j}\}$. Given that $a_i = a_j$, the probability of a conflict between the two users for this attribute (termed a, below) on device d is:

$$P_a = \sum_{1\le k\le n_{s,a}} p_{i,k} - p_{i,k} \times p_{j,k} \tag{7}$$

If the attribute is a numerical one, the fuzzy values for each user are $\widetilde{v}_i = \langle \mu_i, \sigma_i^2\rangle$ and $\widetilde{v}_j = \langle \mu_j, \sigma_j^2\rangle$. Unlike a categorical attribute, this attribute's value is a real number, and thus two values conflict when the difference between them exceeds a user-defined threshold $\mathcal{T}$. Given that $a_i = a_j$ (which we again term a), the probability of the two users conflicting for this device and attribute is computed as:

$$P_a = P(\mathcal{N}_1 > \mathcal{T}) + P(\mathcal{N}_2 > \mathcal{T}) \tag{8}$$

where $\mathcal{N}_1 = \mathcal{N}(\mu_i - \mu_j, \sigma_i^2 + \sigma_j^2)$, $\mathcal{N}_2 = \mathcal{N}(\mu_j - \mu_i, \sigma_i^2 + \sigma_j^2)$.

Recall that every device has a designated "status" attribute that indicates whether it is on or off; in addition, a user can specify a "don't care" state for the "status" attribute within the user preferences. We treat the "status" attribute as a special case, and position the "don't care" value as the first possible in the list of the attribute's potential values. Thus, $p_{i,1}$ and $p_{j,1}$ are, respectively, the likelihood that the user u_1 and u_2 "don't care" about the state of device d. For "status" attribute a, the probability of having conflict is:

$$P_{\text{status}} = \sum_{2 \leq k \leq n_{s,a}} p_{i,k} - p_{i,k} \times p_{j,k} - p_{i,k} \times p_{j,1} \qquad (9)$$

The difference for "status" is that when a user specifies a "don't care" state, it never conflicts with the state specified by the other user even if they are different values. Thus we do not account for differences caused by "don't care".

Overall, for each $p_a > 0$, we generate a conflict scenario $CS = \langle\langle d, a\rangle, CO_\cap, p_a\rangle$.

V. Evaluation

In this section, we evaluate CoPI on two real-world datasets. We seek to answer the following research questions:

- RQ1: Is CoPI capable of producing informative and human-readable conflict scenarios?
- RQ2: Is probabilistic prediction preferred when predicting conflicts in (noisy) real-world situations?
- RQ3: How important is context in predicting conflicts?
- RQ4: How does the granularity of contexts and the value α used for conflict prediction impact the performance of CoPI?

To answer these questions, we implement the aforementioned algorithms in Python[2]. We use *scikit-learn* [46] to realize the clustering algorithm in Algorithm 2[3]. The *threshold* in Algorithm 1 is set to 10 considering the size of the data and the desired number of samples. For the first three experiments, α is determined using a grid search and the optimal discovered value is selected. We evaluate how CoPI is sensitive to the choice of α in the fourth experiment.

In the experiments, we use datasets containing IoT device interactions for multiple users during the same period of time. We generate ground-truth conflicts by extracting overlapping traces of different users and checking if the state of device attributes set by the users would cause conflicts. We first describe the two datasets we used, and then we describe in depth how we generate such ground-truth conflicts.

A. Datasets and Evaluation Setup

The first dataset, REFIT [47] contains electrical data of 20 households in Loughborough, UK collected from 2013 to 2015. The records consist of timestamped power consumption measures for each appliance in Watts, sampled at 8 second intervals. Since it is not possible to differentiate multiple residents in a single household in the data, we treat each household as one user. To minimize the impact of multiple residents, we choose 5 households with fewer than 3 residents and a time range longer than 1.5 years. From the appliances, we choose TV and washing machine as the devices because they have the highest numbers of user interactions in the traces.

The raw power consumption cannot be used as the device state in CoPI, so we pre-process the data to create on and off states for each device. We manually determine a threshold for each device and assume that the device is on when the power consumption is above the threshold and off when it is below. Due to noise in the measurements, we ignore states that last for less than 5 minutes. We then create the event sequence to mimic the user's interactions with these devices to generate these states. For this dataset, the state of the device is described with only this single attribute, "status". We use three context types for REFIT dataset: *minuteOfDay*, *dayOfWeek* and *weather*. The first two are derived from the timestamps in the dataset. The *weather* context is a string describing the current weather condition (e.g., rain, sunny, cloudy), which we retrieve from an online database [4] for the same period of time in Loughborough. In the experiments, we treat *minuteOfDay* as numerical context and other two types as categorical context.

Our second dataset records HVAC settings of an office building in Ottawa, Canada at the room level from 2018 to 2019 [48]. It contains the thermostat set temperature in Celsius, the outdoor temperature, and the occupancy information of 14 rooms sampled in 15 minute intervals. For this dataset, we use the thermostat as the device, and we assume each room is used by a single user. Thus the temperature set in this room is assumed to reflect the preferences of that user. However, because the data is recorded in an office building, the default set temperature is determined by the building manager, and it will reset everyday to a default value if not overridden. To work around this, we use the occupancy information to only process records when the room is occupied. Because of this, the number of valid records is reduced. We choose 3 rooms that have sufficient data (over 1500 hours) for both training and testing. For this dataset, the device "thermostat" has one numerical attribute "setpoint", which is a temperature in Celsius. Note that we ignore the attribute "status" because the thermostat is always on in the building and thus no conflict would result from a difference in the value of "status". We use three context types for the thermostat dataset: *minuteOfDay*, *mode* and *outdoorTemp*. The first context is the same as the previous dataset. The second one, *mode*, is a binary value describing the mode of the thermostat (heating or cooling) set by the building manager[5]. The *outdoorTemp* is a numerical context type representing the outdoor temperature.

In summary, we test 3 types of devices in the experiments: television (**TV**), washing machine (**WM**) and thermostat (**AC**). A conflict arises for the first two devices when one user wants the device to be "on" while another wants the device to be "off'. For the AC, since the attribute "setpoint" is numerical, we assume that a conflict arises when the difference in the set points of two users exceeds 1 degree Celsius. For context types, we use *minuteOfDay*, *outdoorTemp*, *dayOfWeek*,

[2]https://github.com/UT-MPC/ConflictDetection/tree/IoTDI

[3]*scikit-learn* currently does not support categorical features for CART algorithm. Thus we adopt integer encoding to convert categorical contexts which would have slightly worse results but is computationally more efficient.

[4]https://www.worldweatheronline.com/

[5]Note that *mode* is a context type rather than a device attribute because the user has no control over the value of the attribute.

weather and *mode*. The first two are numerical context types and the rest are categorical. When discretizing numerical context types as described in Section IV-A, we use $m_c = 24$ for *minuteOfDay* and $m_c = 12$ for *outdoorTemp*.

B. Ground-truth and metrics

In both datasets, the interactions from different users overlap in time, and users are either in the same building (HVAC) or the same community (REFIT). Thus we assume that context snapshots for different users are the same, and we find ground-truth conflicts by pretending the users live in a shared space. A conflict observation[6] is generated for any pair of users who set the state of a device in conflicting ways at the same time. To estimate the ground-truth probabilities of conflict, we generate samples similar to the process of generating context scenarios in Section IV-A. We discretize numerical contexts to create potential context scenarios, then we count the number of occurrences of each scenario in the data; this value captures how many times a user has experienced a certain context scenario. At each occurrence, if there is a conflict observation, we increase the count of conflicts. This count is recorded separately for each pair of users and device. Thus the ground-truth probability for two users having conflicts for a device under a potential scenario is computed as the count of conflicts for that pair of users for that device divided by the number of occurrences of that context scenario. To ensure the quality of our ground-truth estimation, we only consider scenarios that occurred more than 20 times. Each sample is associated with a pair of users and a device specifying the ground-truth probability of the two users having conflicts over that device.

Throughout the experiments, we use the mean absolute error (MAE) to represent the performance. After the samples are generated, we can compare the ground-truth probability with the probability predicted by the method under evaluation. For instance, if we are evaluating CoPI, we compute the predicted probability for each sample by finding the probability in the predicted conflict scenario that contains this sample. If no conflict scenario can be found, it means we predict the context scenario as non-conflict and thus the predicted probability is 0. For a set of samples, the $\textbf{MAE} = \frac{1}{N}\sum |p - \hat{p}|$ where N is the number of samples, p is the ground-truth probability and $\hat{p}$ is the probability predicted by the method under evaluation.

C. Predicted conflict scenarios

To answer RQ1, we show the predicted conflict scenarios produced by CoPI for real users. We process the TV data for users **H3** and **H9** and the AC data for users **R1** and **R9**. We generate 6 conflict scenarios for AC and 9 for TV in total. Among them, we choose one with the highest probability and another one that has similar context but different probability to showcase the potential of CoPI.

As defined in Definition 3, the conflict scenario we produce is in the form: $\langle d, a, CO, p\rangle$. We present the four conflict scenarios by connecting the components with some words:

[6]We use the phrase *conflict observation* in the ground truth to contrast with the conflict scenario that CoPI predicts.

- **H3** and **H9** have a likelihood of **71.3%** to conflict over the **status** of **TV** when time is (8 pm to 10 pm), day is {"Monday"}, and weather is {"Clear", "Rain", "Fog", "Cloudy"}.
- **H3** and **H9** have a likelihood of **25.4%** to conflict over the **status** of **TV** when time is (6 pm to 7 pm), day is {"Monday"} and weather is {"Clear", "Rain", "Fog", "Cloudy"}.
- **R1** and **R9** have a likelihood of **99.5%** to conflict over the **setpoint** of **AC** when time is (12 pm to 5 pm), outdoor temperature is ($30°C$ to $35°C$) and mode is {"cooling"}.
- **R1** and **R9** have a likelihood of **50.1%** to conflict over the **setpoint** of **AC** when time is (2 pm to 8 pm), outdoor temperature is ($15°C$ to $20°C$) and mode is {"cooling"}.

The predicted conflict scenarios can be understood by a human without much explanation. Our result is informative, as we output the corresponding probability and context with it. This experiment also qualitatively answers RQ2 to support our statement that context-awareness is needed for conflict prediction. In TV conflicts, the probability changes from 71.3% to 25.4% with a difference in time of one hour even though the other contexts remain identical. While interactions with TV may be related to the time of day, the setpoint of the AC is related to outdoor temperature, as a 10 degree difference in temperature incurs a 49.4% difference in probability. This comparison demonstrates that 1) context impacts the likelihood of conflict for real users, and 2) a conflict prediction framework should support various context types because different devices and users may be sensitive to different contexts types.

D. Conflict prediction accuracy

In the second experiment, we compare CoPI with 3 variants to answer RQ2 and RQ3. The variants are:

- No conflict prediction (**NC**): This variant serves as a baseline for not predicting conflicts. It always predicts the probability as 0 (i.e. $\hat{p} = 0$) and thus the MAE equals to the average of the ground truth probability.
- Deterministic prediction (**Det.**): This variant represents some existing solutions that predict conflicts deterministically [2], [7]. We implement this variant by considering only the highest probability in the fuzzy device state. When computing the probability for categorical attributes in Equation 7, the highest probability is changed to 1 while the rest are changed to 0. For numerical attribute in Equation 8, we assume the variance is $\sigma_i^2 = \sigma_j^2 = 0$.
- Static prediction (**Static**): For each pair of users, this method predicts an optimal constant probability that equals the median of the ground-truth probabilities. Note that this number cannot be computed in real life and thus it only shows the best possible performance if we do not predict dynamically based on context.

To test CoPI and the variants, we split the data into training and testing based on days. 40% of the total days in the input data are randomly selected for testing purposes. The ground-truth samples and the MAE are computed for the interaction

TABLE I: Average MAE for all variants in %.

device	method	conflict	non-conflict	overall
WM	NC	5.2±0.5	0.0±0.0	0.1±0.0
	Det.	5.2±0.5	0.0±0.0	0.1±0.0
	Static	5.2±0.5	0.0±0.0	0.1±0.0
	CoPI	4.8±0.5	0.1±0.0	0.2±0.0
TV	NC	23.4±0.7	0.0±0.0	10.1±0.5
	Det.	27.9±1.8	0.2±0.1	12.5±0.8
	Static	21.9±0.8	0.9±0.2	10.0±0.4
	CoPI	9.0±0.3	1.2±0.1	4.6±0.2
AC	NC	50.2±3.0	0.0±0.0	47.0±2.1
	Det.	45.9±3.1	0.0±0.0	43.1±3.2
	Static	16.7±2.6	43.5±5.0	18.3±2.8
	CoPI	13.1±1.4	24.4±6.0	13.8±1.7
AC	Time	18.4±2.6	38.6±19.4	19.7±2.1
TV	Time	8.9±0.2	1.3±0.1	4.7±0.2

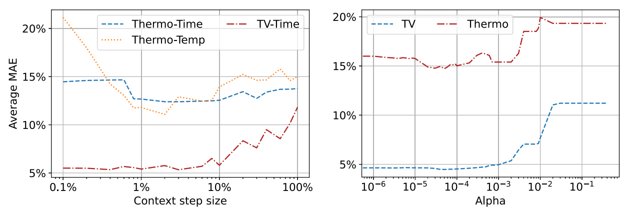
histories in the test days. We repeat the process for 10 times for all three devices. Table. I shows the mean and standard deviation of MAE in percentage for conflict ($p > 0$) and non-conflict ($p = 0$) samples. For TV and WM from the REFIT dataset, we generated over 2700 samples each. For AC from the HVAC dataset, we generated about 110 samples on average. As a result, the variance for AC is noticeably larger. The method "Time" in the last two lines means running CoPI with only one context, *minuteOfDay*. We only show it for TV and AC because WM have very few conflict observations and thus the results have no changes.

For WM, all four methods have almost identical performance. From the MAE of NC in conflict samples, we can see that the ground-truth probability of conflict is low ($< 6\%$). The reason is that users interact with WM infrequently and rarely have conflicts. In this case, simply predicting 0 probability (NC) could be sufficient, but CoPI also strikes a balance by having similar overall performance.

For TV and AC, CoPI has, overall, the best performance. In comparison, deterministic prediction (Det.) can capture the non-conflict cases better, but it has significantly worse performance in conflict cases. The reason is that it predicts either 100% or 0% and thus cannot capture conflicts with diverse probabilities. Especially for AC, since the ground-truth probability is higher, the overall performance of Det. is much worse than CoPI. *Therefore probabilistic prediction is preferred when predicting conflicts for real-world situations.*

On the other hand, static prediction (Static), even with unrealistic ground-truth knowledge, cannot capture both conflict and non-conflict cases well at the same time. For devices like TV, where the ground-truth probabilities for conflict cases are relatively small (23.4%), it captures non-conflict cases well but not the conflict cases. While for devices like AC where the probabilities are higher (50.2%), it captures the conflict cases but not the non-conflict cases. *Overall, predicting conflict dynamically based on context is important for real use cases.*

Although we demonstrate that supporting various contexts is important in the first experiment, we also want to show that supporting more contexts can quantitatively improve the performance. In the last two lines of Table. I, we show the performance of running CoPI with only the context *minuteOfDay*. For TV, since the interaction patterns are more related to time, removing other contexts has little impact on the performance, But for AC, since the outdoor temperature is a key element that influences people's behaviors, removing it makes the performance noticeably worse. Especially for non-conflict samples, the MAE and variance increase significantly. *Therefore, supporting only time as the context as in some previous works [6] is not sufficient for conflict prediction.*



(a) Different step sizes (b) Different alpha values

Fig. 3: Changes in MAE with various context step size and alpha

E. Context granularity

In the third experiment, we evaluate how the granularity in context discretization impacts performance. Context discretization is the process of generating potential context scenarios for the habit patterns as described in Section IV-A. For each numerical context, we divide the domain into m_c equal-sized slots. We evaluate how the choice of m_c impacts the performance of conflict prediction.

We run the experiment for only TV and AC because they have more conflict samples. The numerical contexts are *minuteOfDay* (time) for both TV and AC and *outdoorTemp* (temp) for AC. Fig. 3a shows average MAE. We vary the granularity for one context at a time; the X axis is the step size which is the inverse of m_c. For consistency across contexts, it is shown in percentage on a logarithmic scale.

The performance of CoPI is relatively insensitive to the step size when it is in a reasonable range (1% to 10%). From the line "TV-time" we can see that when the size of the data is sufficiently large, we should choose a small step size. On the contrary, from the two lines for AC, we can see the small step size increases the error because the quality of the estimation for each potential scenario is poor due to insufficient samples in the interaction history. In addition, by comparing the AC lines, we can see that since the behavior of a thermostat is related more to the temperature than the time, the changes in time context have less impact than changes in temperature.

F. CoPI's sensitivity to α

Similar to the previous experiment, we also evaluate CoPI's sensitivity to the value of α from Equation. 6. The purpose of α is to reduce redundancies in the generated conflicts by merging similar patterns. A higher value of α results in fewer redundancies but potentially more error. Thus in this experiment, we vary the choice of α from 0 to 0.5 and predict conflicts for TV and AC. Fig. 3b shows the average MAE against the value of α on a logarithmic scale.

From the results, we can see that the average MAE grows quickly when α is above 10^{-3}. The optimal value for α appears between 10^{-5} and 10^{-4} for both devices but the change in performance is small when α is small. Thus in general, CoPI's performance is not sensitive to the choice of α for small values.

G. Discussion

CoPI successfully captures users' intentions and predicts potential conflicts. In this work, CoPI leverages all available contexts to make predictions. As shown in the experiments, when some contexts have more impacts on users' behaviors than the others, CoPI learns to treat them with finer granularity because of the clustering algorithm in Algorithm 2. However, when processing more context types, CoPI requires more samples for the interaction history. In the future, this may be alleviated by pre-processing interaction histories to remove context types that are less relevant.

With CoPI, we target applications that require *a priori* conflict prediction (e.g., roommate matching, space design). For example, a smart home system can predict conflicts for the next day based on the results from CoPI and the weather forecast to adjust the service accordingly. If the likelihood of raining tomorrow is 80% and CoPI predicts the probability of conflict when raining is 90%, then the actual likelihood of having conflict tomorrow is 72%. This example demonstrates the flexibility of CoPI and the importance of predicting conflict with probabilities in that subsequent system can easily utilize our results to make further computation.

In this paper, we assume CoPI mostly runs offline and infrequently for *a priori* prediction. Thus, we only discuss the scalability problem of CoPI qualitatively. CoPI scales well with the number of devices because the conflicts for each device in a space can be processed in parallel. On the other hand, CoPI finds conflicts for every pair of users and therefore the execution time for CoPI increases exponentially with the number of users. In our implementation, we employ a spatial index tree [49] to store and query more efficiently. In the future, more heuristics can be applied to improve this aspect.

VI. Conclusion

In this paper, we proposed CoPI, a first demonstration of the potential of probabilistic conflict prediction. CoPI enables many new applications such as informing the users of potential conflicts with new roommates or adjusting the IoT resources when designing a new office based on the conflicts among employees. Although we do not focus on resolving conflicts, our context-enriched conflict scenarios are informative to support prioritizing, negotiating and eventually resolving the conflicts in a more efficient and fine-grained way. However, work with CoPI is not complete. Privacy is a missing aspect and future works can focus on providing conflict scenarios with more privacy concerns. In CoPI, we only consider ambient context which we assume every user shares. For personal context (e.g., activity of a user), conflicts may occur when the context is different for the users and thus additional process is required when searching for potential conflict scenarios. In conclusion, CoPI provides an important step in the direction of ambient intelligence for multiple residents by predicting IoT device conflicts with context-awareness.

Acknowledgements

The authors would like to thank the members of the Mobile and Pervasive Computing group at UT Austin for their assistance in the evaluation. This work was funded in part by the National Science Foundation under grants CNS-1909221, CNS-1907959. Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF. A preliminary version of this paper appears in the Ph.D. dissertation of the first author [50].

References

[1] Y. Sun *et al.*, "Conflict detection scheme based on formal rule model for smart building systems," *IEEE Transactions on Human-Machine Systems*, vol. 45, no. 2, pp. 215–227, 2014.

[2] D. Chaki *et al.*, "A conflict detection framework for iot services in multi-resident smart homes," in *Proc. of ICWS*, IEEE, 2020.

[3] A. Ranganathan *et al.*, "Olympus: A high-level programming model for pervasive computing environments," in *Proc. of PerCom*, 2005.

[4] I. Park *et al.*, "A dynamic context-conflict management scheme for group-aware ubiquitous computing environments," in *Proc. of COMPSAC*, IEEE, 2005.

[5] V. Tuttlies *et al.*, "Comity-conflict avoidance in pervasive computing environments," in *Proc. of OTM*, 2007.

[6] D. Chaki *et al.*, "Fine-grained conflict detection of iot services," in *Proc. of SCC*, 2020, pp. 321–328.

[7] M. Yagita *et al.*, "An application conflict detection and resolution system for smart homes," in *Proc. of SEsCPS*, 2015.

[8] S. Munir *et al.*, "Depsys: Dependency aware integration of cyber-physical systems for smart homes," in *Proc. of ICCPS*, 2014.

[9] H. Erzberger *et al.*, "Conflict detection and resolution in the presence of prediction error," in *Proc. of ATM R&D*, 1997.

[10] T. Shah *et al.*, "Conflict detection in rule based iot systems," in *Proc. of IEMCON*, 2019.

[11] K. Parrott *et al.*, "Work at home: Conflict and compromise on the use of space," *Housing and Society*, vol. 17, no. 3, pp. 17–25, 1990.

[12] C. Liu *et al.*, "Scents: Collaborative sensing in proximity iot networks," in *Proc. of PerCom Workshops*, 2019.

[13] C. S. Abella *et al.*, "Autonomous energy-efficient wireless sensor network platform for home/office automation," *IEEE Sensors Journal*, vol. 19, no. 9, pp. 3501–3512, 2019.

[14] T. Gu *et al.*, "A pattern mining approach to sensor-based human activity recognition," *IEEE Transactions on Knowledge and Data Engineering*, vol. 23, no. 9, pp. 1359–1372, 2010.

[15] I. H. Sarker *et al.*, "Abc-ruleminer: User behavioral rule-based machine learning method for context-aware intelligent services," *Journal of Network and Computer Applications*, vol. 168, p. 102 762, 2020.

[16] A. Yassine *et al.*, "Mining human activity patterns from smart home big data for health care applications," *IEEE Access*, vol. 5, 2017.

[17] P. Lago *et al.*, "Learning and managing context enriched behavior patterns in smart homes," *Future Generation Computer Systems*, vol. 91, pp. 191–205, 2019.

[18] I. Armac *et al.*, "Modeling and analysis of functionality in ehome systems: Dynamic rule-based conflict detection," in *Proc. of ECBS*, 2006.

[19] M. Zoller *et al.*, "Realization of a upnp agent for controlling adaptive multimedia environments," M.S. thesis, Technische Universität Darmstadt, 2008.

[20] F. J. Miandashti *et al.*, "An empirical approach to modeling user-system interaction conflicts in smart homes," *IEEE Transactions on Human-Machine Systems*, vol. 50, no. 6, pp. 573–583, 2020.

[21] S. Resendes *et al.*, "Conflict detection and resolution in home and building automation systems: A literature review," *J. of Ambient Intelligence and Humanized Computing*, vol. 5, pp. 699–715, 2014.

[22] J. W. Adaska *et al.*, "Robust probabilistic conflict prediction for sense and avoid," in *2014 American Control Conference*, IEEE, 2014, pp. 1198–1203.

[23] C. Brindescu *et al.*, "Planning for untangling: Predicting the difficulty of merge conflicts," in *2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE)*, IEEE, 2020, pp. 801–811.

[24] C. Shin *et al.*, "Conflict resolution method utilizing context history for context-aware applications," *Cognitive Science Research Paper*, vol. 577, p. 105, 2005.

[25] L. Capra *et al.*, "Carisma: Context-aware reflective middleware system for mobile applications," *IEEE Transactions on Software Engineering*, vol. 29, no. 10, pp. 929–945, 2003.

[26] T. R. B. Silva *et al.*, "Conflicts treatment for ubiquitous collective and context-aware applications," *Journal of Applied Computing Research*, vol. 1, no. 1, pp. 33–47, 2011.

[27] B. Ospan *et al.*, "Context aware virtual assistant with case-based conflict resolution in multi-user smart home environment," in *2018 international conference on computing and network communications (coconet)*, IEEE, 2018, pp. 36–44.

[28] A. Alkhresheh *et al.*, "Context-aware automatic access policy specification for iot environments," in *2018 14th International Wireless Communications & Mobile Computing Conference (IWCMC)*, IEEE, 2018, pp. 793–799.

[29] A. Al Farooq *et al.*, "Iotc 2: A formal method approach for detecting conflicts in large scale iot systems," in *Proc. of IM*, 2019.

[30] C. Shin *et al.*, "Mixed-initiative conflict resolution for context-aware applications," in *Proc. of UbiComp*, 2008.

[31] C. Vannucchi *et al.*, "Symbolic verification of event–condition–action rules in intelligent environments," *Journal of Reliable Intelligent Environments*, vol. 3, no. 2, pp. 117–130, 2017.

[32] M. Galushka *et al.*, "Temporal data mining for smart homes," in *Designing Smart Homes*, Springer, 2006, pp. 85–108.

[33] F. Leotta *et al.*, "Surveying human habit modeling and mining techniques in smart spaces," *Future Internet*, vol. 11, no. 1, p. 23, 2019.

[34] J. Hua *et al.*, "Riot: Enabling seamless context-aware automation in the internet of things," in *Proc. of MASS*, 2019.

[35] O. Brdiczka *et al.*, "Learning situation models in a smart home," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 39, no. 1, pp. 56–63, 2008.

[36] I. H. Sarker, "A machine learning based robust prediction model for real-life mobile phone data," *Internet of Things*, vol. 5, 2019.

[37] M. Ehatisham-ul-Haq *et al.*, "Opportunistic sensing for inferring in-the-wild human contexts based on activity pattern recognition using smart computing," *Future Generation Computer Systems*, vol. 106, pp. 374–392, 2020.

[38] G. Singla *et al.*, "Recognizing independent and joint activities among multiple residents in smart environments," *Journal of Ambient Intelligence and Humanized Computing*, vol. 1, no. 1, pp. 57–63, 2010.

[39] P. Rashidi *et al.*, "Keeping the resident in the loop: Adapting the smart home to the user," *IEEE Trans. on Systems, Man, and Cybernetics, Part A (Systems and Humans)*, vol. 39, no. 5, pp. 949–959, 2009.

[40] T. Van Kasteren *et al.*, "Accurate activity recognition in a home setting," in *Proc of UbiComp*, 2008.

[41] A. Aztiria *et al.*, "Discovering frequent user–environment interactions in intelligent environments," *Personal and Ubiquitous Computing*, vol. 16, no. 1, pp. 91–103, 2012.

[42] R. Agrawal *et al.*, "Fast algorithms for mining association rules," in *Proc. of VLDB*, 1994.

[43] J. Han *et al.*, "Mining frequent patterns without candidate generation," *ACM sigmod record*, vol. 29, no. 2, pp. 1–12, 2000.

[44] A. K. Dey, "Understanding and using context," *Personal and Ubiquitous Computing*, vol. 5, no. 1, pp. 4–7, 2001.

[45] L. Breiman *et al.*, *Classification and regression trees*. Routledge, 2017.

[46] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in python," *Journal of Machine Learning research*, vol. 12, pp. 2825–2830, 2011.

[47] D. Murray *et al.*, "A data management platform for personalised real-time energy feedback," 2015.

[48] W. Liu *et al.*, "Modeling window and thermostat use behavior to inform sequences of operation in mixed-mode ventilation buildings," *Science and Technology for the Built Environment*, vol. 27, no. 9, 2021.

[49] A. Guttman, "R-trees: A dynamic index structure for spatial searching," in *Proceedings of the 1984 ACM SIGMOD international conference on Management of data*, 1984, pp. 47–57.

[50] J. Hua *et al.*, "Facilitate interactions with devices in the internet of things (iot) with context-awareness," Ph.D. dissertation, 2022.