sustainability

MDPI

Article

Improving Temporal Event Scheduling through STEP Perpetual Learning

Jiahua Tang [1], Du Zhang [1,*], Xibin Sun [2] and Haiou Qin [3]

[1] Faculty of Innovation Engineering, Macau University of Science and Technology, Macau SAR 999078, China
[2] Computer Engineering Technical College, Guangdong Polytechnic of Science and Technology, Guangzhou 510640, China
[3] School of Information Engineering, Nanchang Institute of Technology, Nanchang 330029, China
* Correspondence: duzhang@must.edu.mo

Abstract: Currently, most machine learning applications follow a one-off learning process: given a static dataset and a learning algorithm, generate a model for a task. These applications can neither adapt to a dynamic and changing environment, nor accomplish incremental task performance improvement continuously. STEP perpetual learning, by continuous knowledge refinement through sequential learning episodes, emphasizes the accomplishment of incremental task performance improvement. In this paper, we describe how a personalized temporal event scheduling system SmartCalendar, can benefit from STEP perpetual learning. We adopt the interval temporal logic to represent events' temporal relationships and determine if events are temporally inconsistent. To provide strategies that approach user preferences for handling temporal inconsistencies, we propose SmartCalendar to recognize, resolve and learn from temporal inconsistencies based on STEP perpetual learning. SmartCalendar has several cornerstones: similarity measures for temporal inconsistency; a sparse decomposition method to utilize historical data; and a loss function based on cross-entropy to optimize performance. The experimental results on the collected dataset show that SmartCalendar incrementally improves its scheduling performance and substantially outperforms comparison methods.

Keywords: *STEP* perpetual learning; temporal inconsistency; interval temporal logic; temporal event scheduling; SmartCalendar

check for updates

Citation: Tang, J.; Zhang, D.; Sun, X.; Qin, H. Improving Temporal Event Scheduling through STEP Perpetual Learning. *Sustainability* **2022**, *14*, 16178. https://doi.org/10.3390/su142316178

Academic Editors: Haoran Wei, Zhendong Wang, Yuchao Chang and Zhenghua Huang

Received: 20 October 2022
Accepted: 29 November 2022
Published: 3 December 2022

1. Introduction

To date, most machine learning applications follow a one-off metaphor: given a static dataset and a learning algorithm, generate a function or a model for a task. Such a metaphor prevents these applications from adapting to the dynamic environment and achieving incremental task performance improvement [1–3]. Unlike the one-off metaphor, human learning is a long-term process with continuous knowledge refinement [4]. Researchers have proposed several paradigms that model the complexity, diversity, and accumulative nature of human learning [5–9]: *Lifelong Learning*, *Never-ending learning*, and *STEP perpetual learning*.

Lifelong Learning (LL) learns to obtain knowledge continuously by carrying out sequential tasks [6–8]. The knowledge of past tasks is accumulated so that the learner can make use of them to help learn a new task [6]. *Never-ending learning* is a paradigm that: learns various types of knowledge, improves subsequent learning based on learned knowledge, and with sufficient self-reflection [3,5]. NEL defines the problem as an ordered pair comprised of a set of learning tasks and coupling constraints [3]. However, these solutions do not formally define learning stimuli, nor do they distinguish learning episodes and working episodes.

Unlike previous studies, *STEP perpetual learning* (PL) is a novel paradigm that regards learning *stimuli* as important as tasks, experience, and performance measures [2,10–13]. Each time a perpetual learning agent (PeLA) encounters stimuli, learning episodes will be triggered to improve the agent's problem-solving knowledge, which leads to better performance. Therefore, *STEP* PL enables us to design a special PeLA, SmartCalendar, whose performance will be progressively better and can satisfactorily perform tasks in the end.

This work chooses the calendar application as one usage scenario to demonstrate how a PeLA incrementally improves its performance through continuous knowledge refinement. In today's fast-paced world, many people use calendar systems to improve productivity and organize life. Internet giants recognized the importance of calendar systems, and released their applications, e.g., Google calendar, Microsoft Outlook, and Apple calendar [14–16]. Despite the vast investment in these applications, we find that these applications are still flawed in the core function, which is helping users schedule events. For instance, if the user has two events to attend on 1 October 2022. One is an exam from 9 a.m. to 10 a.m. in Room A201, and the other is a meeting from 10 a.m. to 11 a.m. in Room C408. Because there is not enough time between two events for the user to move from one location to another, the user cannot attend the second event on time. We say there is a temporal conflict or *temporal inconsistency* (TI) between two events [17] (formal definitions of events and TIs are given later). In this case, the user can reschedule events by adjusting the starting point and ending point of events. Considering that it is difficult for users to detect all conflicts in time, especially those caused by the transition of events, an intelligent calendar system should be able to detect and solve temporal conflicts between events. However, none of the aforementioned calendar applications has such capabilities.

Researchers proposed some scheduler models [18–22] to resolve conflicts. TASHA [18] utilizes rules based on event types to resolve temporal conflicts. ADAPTS employs a Decision Tree (DT) to make decisions from four strategies: modify the original event, modify the new event, modify both events, and delete the original event [19,20]. As an updated version, Ref. [22] adopts a non-linear programming model which objects to minimizing the duration changes of conflicting events. However, the above scheduler models are still far from an intelligent event scheduling system. They treat all events as equally important [21], which contradicts the highly personalized nature of the calendar scene. Refs. [19,22] trained a DT model on the complete dataset, which is incompatible with the cumulative nature of the calendar system. Additionally, Refs. [19,20,22] oversimplify the user's actual strategies by categorizing solutions into four classes.

1.1. Challenges

Here are questions an intelligent event scheduling system must answer:

1. How to schedule temporally conflicting events using existing knowledge?

What the system knows is past events and conflicts with solutions. The number of events and TIs the user has per day is limited, whereas most of the attributes of a TI are discrete attributes with dozens of values. Therefore, for a long time after the use, the system's knowledge is insufficient for constructing a conventional ML model to solve TIs directly [1,23]. In this work, we propose three methods *replicate solutions of identical cases* (RSIC), *reference solutions of similar cases* (RSSC), and *generate solutions with strategy distribution* (GSSD) to deal with TIs in different scenarios: Given a TI, RSIC looks for cases where user preferences are determined, i.e., the *same TI* in history, then adopts the strategy used in the same TI. RSSC seeks *similar TIs* based on the importance of attribute values and draws on their solutions to generate a strategy. If there does not exist the same or similar TI, GSSD obtains a strategy from the distribution of historical user decisions. Therefore, at all stages of operation, the system chooses the most appropriate method to resolve the current conflict.

2. How to consistently improve the system's performance?

As a personalized system, the user's preference is the golden rule for addressing TIs, which requires the system to incrementally improve its performance to adapt to an individual's preferences over time. To this end, we design the stimuli-driven learning processes according to the *STEP* PL framework: The completion of each working episode drives the system to resolve knowledge inadequacy by recording relevant data and revising the corresponding meta-knowledge. Disagreement between the user and the system on the TI solution prompts the system to address knowledge deficiency by refining the problem-solving model. As a result, the system has more data for reference and is gradually approaching users' decision preferences. Eventually, the system can generate user-acceptable solutions to TIs.

1.2. Contributions

Our work's contributions are summarized as follows:

1. Interval temporal logic (ITL) only considers the original interval relationships within events. While in real life, realistic events could occur at various locations, and the transition of events cannot be ignored for the fulfillment of an event. Thus, we propose complete temporal classes based on ITL to identify temporal relations by considering the events' transition.
2. To the best of our knowledge, we are the first to model temporal events scheduling and management system under *STEP* PL. We introduce a theoretical model to provide guidelines to develop algorithms to recognize, resolve, and more importantly, learn from TIs. The stimuli-driven learning processes enable the system to realize incremental performance improvement.
3. There is no existing calendar dataset that matches our definitions. We collected a five-user dataset, which is available at https://github.com/hensontang9/Temporal_conflicts, (accessed on 14 September 2022). For each user, there are about 4000 events and 600 Tis. Experiments on the collected dataset verified the feasibility and self-improvement capability: Our system exhibits significantly better performance than the comparison methods. Moreover, it incrementally improves its scheduling performance during use. A prototype is implemented on the Android platform, which is available at https://github.com/hensontang9/SmartCalendar, (accessed on 14 September 2022).

2. Related Work

2.1. Temporal Logics

Temporal logics are systems of rules and symbolism for temporal representation and reasoning, which are categorized by Refs. [17,24]: "*propositional* versus *first-order*", "*linear* versus *branching*", "*point* versus *interval*", etc. ITLs are first-order, linear time, intervals, and continuous operators [17,25,26] and possess a richer representation formalism to define interval relations than the point-based scheme due to the additional expressiveness obtained by reasoning about time interval [24,27]. Hence, we chose ITL to help qualitatively delineate temporal relations within events.

2.2. Temporal Scheduling Applications

To the best of our knowledge, the existing temporal scheduling applications focus on either scheduling periodical events [28–31], tracking events that users have interest in [32–34], efficiently creating and sharing appointments according to the given availability preference [35,36], or planning events with the minimized start-to-end time duration [37,38]. They do not cover developing an event scheduling system that acquires the capability to resolve temporal conflicts and improve performance over time.

2.3. Incremental Performance Improvement Paradigms

M.I. Jordan et al. pointed out that one of the challenges in machine learning is to develop systems that never stop learning new tasks and improving their performance continuously [3]. Related explorations include lifelong learning [6–8], never-ending learning [5], perpetual learning [2,10–13], etc.

Lifelong learning agents learn one task at a time, continuously acquire new knowledge and refine existing knowledge [6,7,39]. By adding a specific KB, the capability to identify new tasks, and the ability to learn on the job, Z. Chen et al. extended the definition of LL and employed the LL method in the field of topic modeling [8,40,41]. NEL is a system that can improve itself using the knowledge learned from self-supervised experience [5]. As a case study, *Never-Ending Language Learner* has been equipped with 120 million candidate beliefs by extracting information from the Web [9,42,43].

There are several important differences between the above paradigms and *STEP* PL: First, the learning episodes in PL are discrete and triggered by stimuli, whereas learning in the aforementioned approaches is not triggered by any events and is largely continuous. Second, PL emphasizes the accomplishment of incremental task performance improvement through sequential learning episodes, whereas the above work is primarily oriented toward knowledge acquisition. In this paper, our calendar system is designed following the concept of *STEP* PL.

3. Preliminaries

3.1. Events

An event ε is defined to be:

$$\varepsilon = (\epsilon, \alpha, \tau, \vartheta, p, \mathcal{L}, \zeta, \iota) \tag{1}$$

where

- ϵ indicates the event type, $\epsilon \in \{\text{work, study, social, family, entertainment, personal}\}$;
- α represents the activity, $\alpha \in Act_\epsilon$ where Act_ϵ is the set of activities belonging to event type ϵ;
- τ is the time interval, $\tau = (St, St + Dur)$, St is the starting point, Dur is the duration;
- ϑ denotes an event's flexibility, $\vartheta \in \{\text{rigid, flexible}\}$. The time intervals of rigid events need to be strictly adhered to, while that of flexible events can be adjusted;
- p represents the host and participant, $p = \left(p^{host}, p^{part}\right)$, in which
- p^{host} denotes a host in a set $\boldsymbol{p}^{host}$ of all events' hosts, i.e., $p^{host} \in \boldsymbol{p}^{host}$;
- p^{part} is a participant in a set $\boldsymbol{p}^{part}$ of all events' participants, i.e., $p^{part} \in \boldsymbol{p}^{part}$;
- $\mathcal{L}$ indicates the location, $\mathcal{L} = \left(\mathcal{L}^{name}, \mathcal{L}^{long}, \mathcal{L}^{la}\right)$, $\mathcal{L}^{name}$, $\mathcal{L}^{long}$ and $\mathcal{L}^{la}$ are the name, longitude, and latitude of the location;
- ζ denotes the periodicity, $\zeta \in \{\text{once, every day, every week, every month, every year}\}$
- ι is a flag representing the whole event's importance, $\iota \in \{\text{important, normal}\}$.

3.2. Complete Temporal Classes

Based on the primitive relationship *Meets* [25,44,45], Figure 1 depicts the interval relationships between τ_i and τ_j (τ_i and τ_j are time intervals of events ε_i and ε_j) [17]: $\text{Before}(\tau_i, \tau_j)$, $\text{Overlaps}(\tau_i, \tau_j)$, $\text{Starts}(\tau_i, \tau_j)$, $\text{During}(\tau_i, \tau_j)$, $\text{Finishes}(\tau_i, \tau_j)$, $\text{Equals}(\tau_i, \tau_j)$.

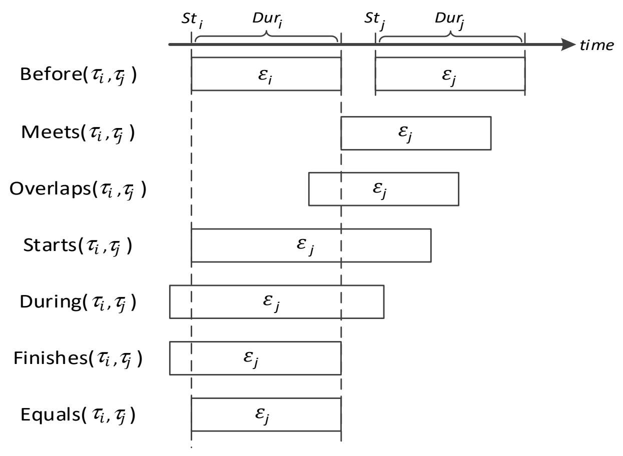

Figure 1. Interval temporal relations for events ε_i and ε_j [17].

However, in the calendar scenario, events' transitions cannot be ignored since events may occur in different locations. Let $\tau_{ij} = \left(St_{ij}, St_{ij} + Dur_{ij}\right)$ denote the time interval to travel from $\mathcal{L}_i$ to $\mathcal{L}_j$ ($\mathcal{L}_i$, $\mathcal{L}_j$ are locations where ε_i, ε_j take place. St_{ij} and Dur_{ij} are the starting point and duration of transition). If ε_i and ε_j can be realized concerning a single given circumstance, then ε_i and ε_j are consistent with each other and we use $\vDash \left(\varepsilon_i, \varepsilon_j\right)$ to denote that. Otherwise, we say that they are inconsistent and this is denoted as $\nvDash \left(\varepsilon_i, \varepsilon_j\right)$. We propose *temporal class* (TC) to represent the temporal relation between two events by considering events' transitions.

There does not exist temporal inconsistency between ε_i and ε_j and both events can be scheduled without compromise if any of the following statements is true:

- Meets$\left(\tau_i, \tau_j\right)$, when $Dur_{ij} = 0$ $(\mathrm{NTI-M_0})$
- Before$\left(\tau_i, \tau_j\right)$, when $Dur_{ij} = 0$ $(\mathrm{NTI-B_0})$
- Before$\left(\tau_i, \tau_j\right) \wedge St_j \geq St_{ij} + Dur_{ij}$, when $Dur_{ij} > 0$ $(\mathrm{NTI-B_1})$

The presence of overlapping time intervals indicates direct temporal conflicting circumstances. Direct temporal conflicting circumstances can be further divided into the following two subcases. The first subcase is that conflicting events occur at the same location and the user does not need extra time to commute $\left(Dur_{ij} = 0\right)$. We are aware that there is a *direct temporal inconsistency* between ε_i and ε_j without the travel time complication if any of the following conditions is true:

- Overlaps$\left(\tau_i, \tau_j\right)$, when $Dur_{ij} = 0$ $(\mathrm{DTI-O_0})$
- Starts$\left(\tau_i, \tau_j\right)$, when $Dur_{ij} = 0$ $(\mathrm{DTI-S_0})$
- Equals$\left(\tau_i, \tau_j\right)$, when $Dur_{ij} = 0$ $(\mathrm{DTI-E_0})$
- During$\left(\tau_i, \tau_j\right)$, when $Dur_{ij} = 0$ $(\mathrm{DTI-D_0})$
- Finishes $\left(\tau_i, \tau_j\right)$, when $Dur_{ij} = 0$ $(\mathrm{DTI-F_0})$

Let $\nvDash_{\mathrm{dti-nt}} \left(\varepsilon_i,\ \varepsilon_j\right)$ denote the following:

$$\nvDash_{\mathrm{dti-nt}} \left(\varepsilon_i, \varepsilon_j\right) \equiv_{\mathrm{def}} \left[\nvDash \left(\varepsilon_i, \varepsilon_j\right) \wedge [\mathrm{DTI-O_0} \vee \mathrm{DTI-S_0} \vee \mathrm{DTI-E_0} \vee \mathrm{DTI-D_0} \vee \mathrm{DTI-F_0}]\right]. \quad (2)$$

The second subcase is that the locations of conflicting events are different and the user needs extra time to travel from one location to another $\left(Dur_{ij} > 0\right)$. We are aware that there is a *direct temporal inconsistency* between ε_i and ε_j with the travel time complication if any of the following conditions are true:

- Overlaps$\left(\tau_i, \tau_j\right)$, when $Dur_{ij} > 0$ $(\mathrm{DTI-O_1})$
- Starts$\left(\tau_i, \tau_j\right)$, when $Dur_{ij} > 0$ $(\mathrm{DTI-S_1})$
- Equals$\left(\tau_i, \tau_j\right)$, when $Dur_{ij} > 0$ $(\mathrm{DTI-E_1})$
- During$\left(\tau_i, \tau_j\right)$, when $Dur_{ij} > 0$ $(\mathrm{DTI-D_1})$
- Finishes $\left(\tau_i, \tau_j\right)$, when $Dur_{ij} > 0$ $(\mathrm{DTI-F_1})$

Let $\nvDash_{\mathrm{dti-tc}} \left(\varepsilon_i, \varepsilon_j\right)$ denote the following:

$$\nvDash_{\mathrm{dti-tc}} \left(\varepsilon_i, \varepsilon_j\right) \equiv_{\mathrm{def}} \left[\nvDash \left(\varepsilon_i, \varepsilon_j\right) \wedge [\mathrm{DTI-O_1} \vee \mathrm{DTI-S_1} \vee \mathrm{DTI-E_1} \vee \mathrm{DTI-D_1} \vee \mathrm{DTI-F_1}]\right]. \quad (3)$$

Let $\nvDash_{\mathrm{dti}} \left(\varepsilon_i, \varepsilon_j\right)$ denote the direct temporal inconsistency between ε_i and ε_j:

$$\nvDash_{\mathrm{dti}} \left(\varepsilon_i, \varepsilon_j\right) \equiv_{\mathrm{def}} \left[\nvDash_{\mathrm{dti-nt}} \left(\varepsilon_i, \varepsilon_j\right) \vee \nvDash_{\mathrm{dti-tc}} \left(\varepsilon_i, \varepsilon_j\right)\right]. \quad (4)$$

Even if ε_i and ε_j do not overlap, the completion of events will be compromised if the duration of commuting time is greater than the gap between ε_i and ε_j. We refer to this kind of conflicts as *indirect temporal inconsistencies* (ITI) and denote it by $\nvDash_{\text{iti}} \left(\varepsilon_i, \varepsilon_j\right)$:

$$\nvDash_{\text{iti}} \left(\varepsilon_i, \varepsilon_j\right) \equiv_{\text{def}} \left[\nvDash \left(\varepsilon_i, \varepsilon_j\right) \wedge [\text{ITI} - \text{M}_1 \vee \text{ITI} - \text{B}_1]\right], \tag{5}$$

where $\text{ITI} - \text{M}_1$ and $\text{ITI} - \text{B}_1$ are cases met the following conditions:

- Meets$\left(\tau_i, \tau_j\right)$, when $Dur_{ij} > 0$ $\quad (\text{ITI} - \text{M}_1)$
- Before$\left(\tau_i, \tau_j\right) \wedge St_j \left\langle St_{ij} + Dur_{ij}, \text{ when } Dur_{ij} \right\rangle 0$ $\quad (\text{ITI} - \text{B}_1)$

Temporal inconsistency between ε_i and ε_j, denoted as $\nvDash_{\text{ti}} \left(\varepsilon_i, \varepsilon_j\right)$, is defined to be:

$$\nvDash_{\text{ti}} \left(\varepsilon_i, \varepsilon_j\right) \equiv_{\text{def}} \left[\nvDash_{\text{dti}} \left(\varepsilon_i, \varepsilon_j\right) \vee \nvDash_{\text{iti}} \left(\varepsilon_i, \varepsilon_j\right)\right]. \tag{6}$$

It is noteworthy that when a user commutes to another location it is subjective and not fixed. Here, we assume the user commutes immediately after completing one event, corresponding to the shortest conflict length of all temporally conflicting cases.

3.3. Strategies to Overcome Temporal Conflicts

As an essential basis for adjusting events, the importance of an event is collectively influenced by attributes other than time information. These attributes are called *decision attributes*, including event type, activity, host, participant, location name, and periodicity.

An *action* A defines the modification applied to an event's time point:

$$A = \left(a^{type}, a^{time}\right), \tag{7}$$

where *action type* $a^{type} \in \{hold, advance, postpone, abandon\}$. *hold* implies remaining a time point unchanged; *advance/postpone* represents adjusting a time point earlier/later; *abandon* indicates discarding the time interval. a^{time} represents the time length to adjust, which is zero when a^{type} is hold or abandon.

The strategy C describes adjustments applied to a TI's events, and is defined as:

$$C = (A_{1,start}, A_{1,end}, A_{2,start}, A_{2,end}), \tag{8}$$

where $A_{i,\ start}$ and $A_{i,end} (i = 1,2)$ are actions applied to the starting point and ending point of the i-th event in the TI.

We use the *strategy type* c^{type} to qualitatively analyze a TI, which is defined to be:

$$c^{type} = \left(a^{type}_{1,start}, a^{type}_{1,end}, a^{type}_{2,start}, a^{type}_{2,end}\right), \tag{9}$$

where $a^{type}_{i,start}$ and $a^{type}_{i,end}$ $(i = 1,2)$ are action types of $A_{i,\ start}$ and $A_{i,end}$, respectively.

For brevity, we use c^{sys}_n and c^{user}_n to denote the strategy type that the system suggested and the user adopted in TI_n, respectively.

4. STEP PL Framework

STEP PL focuses on developing a PeLA that can consistently and continuously improve its performance at tasks over time [2]. As depicted in Figure 2, a PeLA runs in an episodic manner, where an episode can be classified as a working or learning episode.

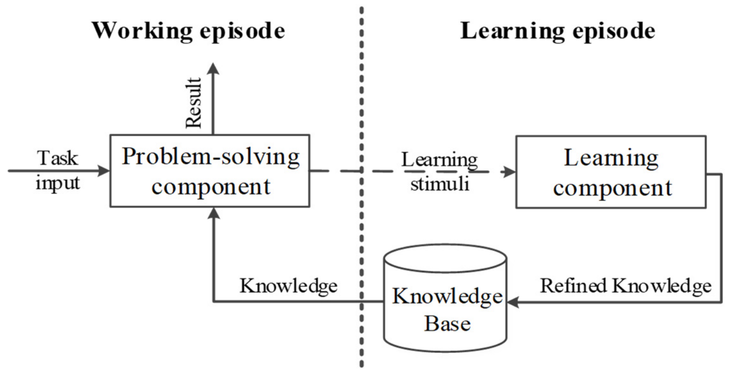

Figure 2. STEP PL framework.

Working episode. Each time the agent performs a task, it first queries corresponding knowledge in the knowledge base (KB), i.e., experience E. Based on the knowledge queried, the problem-solving component generates a result for the task. By evaluating the processing result with metrics P, we can tell how well the agent performs. However, the agent may fail to achieve the desired performance for the task if experience E is flawed or the environment changes. Hence, the agent improves its performance through the stimuli-driven learning processes, where learning stimuli S are served by knowledge deficiencies. Once the agent detects a stimulus, it knows its knowledge in E cannot properly and adequately handle the task. At this point, a subsequent learning episode is triggered by the stimulus.

Learning episode. In a learning episode, the learning component uses stimulus-specific algorithms or heuristics to augment or revise the existing knowledge, resulting in an improvement in P. In the long-running process, the agent refines E continuously, leading to incremental performance improvements of P at T over time. In the end, the agent can satisfactorily perform tasks in T.

5. Learning through Solving Temporal Conflicts

This section first briefly introduces the system from *STEP* PL perspectives, then describes approaches to resolve TIs, and finally presents the stimuli-driven learning processes.

5.1. STEP PL Perspectives of the SmartCalendar System

Figure 3 illustrates how SmartCalendar consistently and incrementally improves performance from the *STEP* PL perspectives. We define the Tasks, Learning stimuli, Performance measure, and Experience as follows.

- Tasks: We define tasks $T = \{\ T_1,\ T_2\}$. T_1 is an arrangement of an ordinary event that does not temporally conflict with others. T_2 is a rescheduling of temporally conflicting events.
- Stimuli: We define learning stimuli $S = \{S_1, S_2, S_2'\}$. S_1 and S_2 stand for successful processing tasks T_1 and T_2, respectively; S_2' refers to the user disagreeing with the system suggestion.
- Experience: we define experience E as the knowledge base that saves data regarding past events and TIs, and meta-knowledge including the *IPR profile union*, *historical TIs' dvalues*, *system-user strategy matrix*, and *model parameters*.
- Performance metric: we define performance metric P as the set containing two metrics. The first metric is *weighted cross-entropy*, which evaluates the distance between the system's prediction and the user's decision. And the second metric is *strategy type acceptance rate*, which assesses whether the system's strategy type is compatible with the user's.

In Section 5.2, we introduce methods for resolving TIs. In Section 5.3, we propose three learning components LC_1, LC_2 and LC_3 to refine knowledge regarding events, TIs and update models. After processing T_1, S_1 triggers LC_1. After processing T_2, the system generates a suggestion to reschedule conflicting events. If the user accepts the system's suggestion, S_2 triggers LC_1 and LC_2 sequentially. Otherwise, S_2', the inconsistency between the system's and the user's decision, triggers LC_1, LC_2 and LC_3 sequentially.

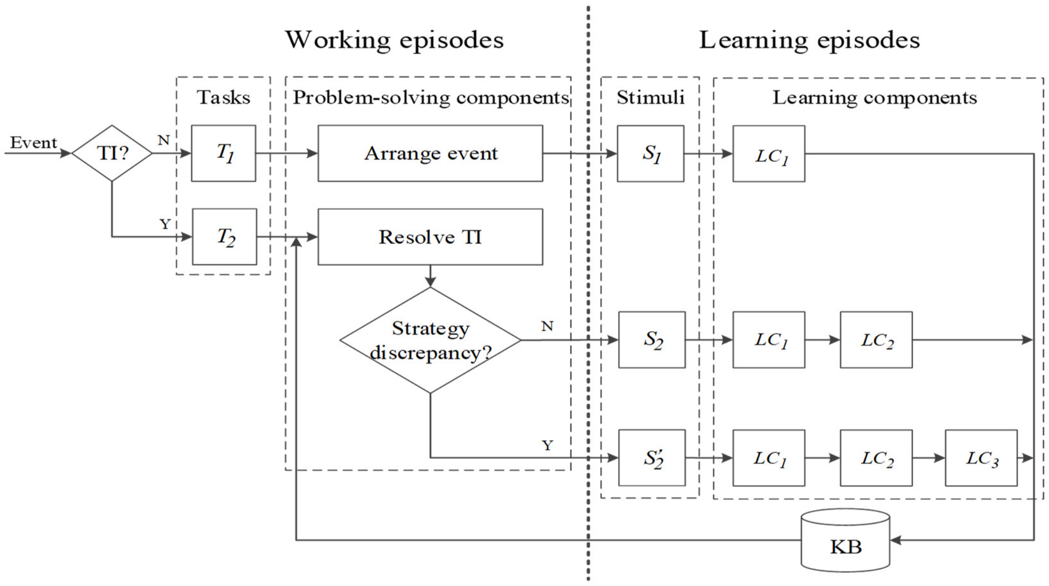

Figure 3. System framework.

5.2. Resolutions for Temporal Conflicts

5.2.1. Heuristics

The system provides recommendations based on the following principles.

Principle 1: A feasible strategy is reasonable and effective. *Reasonableness demands that a strategy satisfies events' flexibility, makes as few changes as possible, and obeys the linearity of time: the starting point should precede the ending point. Effectiveness requires a strategy conducive to the TI's settlement.*

Given a TI_n comprises ε_i and ε_j ($St_i \leq St_j$). Table 1 lists all reasonable strategy types $\left\{c_{(l)}^{type}\right\}_{l=1}^{17}$ when both events are flexible.

Table 1. All reasonable strategy types when both events are flexible.

Optional Pairs of Action Types for ε_j	**Optional Pairs of Action Types for ε_i**					
	hold—hold	***postone—hold***	***postone—advance***	***advance—advance***	***postone—postone***	***abandon—abandon***
hold— hold		$c_{(5)}^{type}$	$c_{(6)}^{type}$	$c_{(7)}^{type}$	$c_{(8)}^{type}$	$c_{(9)}^{type}$
postpone—hold	$c_{(1)}^{type}$		$c_{(10)}^{type}$	$c_{(11)}^{type}$		
hold—advance		$c_{(12)}^{type}$			$c_{(13)}^{type}$	
advance — advance	$c_{(2)}^{type}$	$c_{(14)}^{type}$			$c_{(15)}^{type}$	
postpone — postpone	$c_{(3)}^{type}$		$c_{(16)}^{type}$	$c_{(17)}^{type}$		
abandon — abandon	$c_{(4)}^{type}$					

The effectiveness of a strategy type is determined by comparing the conflict length with the adjustable length of events in the specified directions. We first describe how to calculate the adjustable length of a single event in a specific direction. Let $\varepsilon_{i,day}$ denote the set of events on the day of ε_i and that do not conflict with ε_i. Denote by $\boldsymbol{\varepsilon}_{i,pre} \subseteq \boldsymbol{\varepsilon}_{i,day}$, $\boldsymbol{\varepsilon}_{i,next} \subseteq \boldsymbol{\varepsilon}_{i,day}$ the set of events that end no later than St_i, and the set of events that start no earlier than $St_i + Dur_i$, respectively. If ε_p is the event with the largest ending point in the $\boldsymbol{\varepsilon}_{i,pre}$, then we say ε_p is the *previous event* of ε_i, denoted as $Previous(\varepsilon_p, \varepsilon_i)$. If ε_p is the event with the smallest starting point in the $''_{i,pre}$, then ε_p is the *next event* of ε_i, denoted as $Next(\varepsilon_p, \varepsilon_i)$. Denote by $Dur^{shorten}$, $Dur^{advance}$, and Dur^{delay} the maximum length that an event can be shortened, started earlier, and ended later, respectively. The value of $Dur^{shorten}$ is 0 if the event is rigid, and half the event's duration otherwise. The $Dur^{advance}$ and Dur^{delay} of ε_i, denoted as $Dur_i^{advance}$ and Dur_i^{delay}, are given by:

$$Dur_i^{advance} = \begin{cases} 0, & \varepsilon_i \text{ } is \text{ } rigid \\ St_i - St_p - Dur_p, & \exists \varepsilon_p \text{ } satisfies \text{ } Previous(\varepsilon_p, \varepsilon_i), \\ St_i - Time_{st}(\alpha_i), & \text{otherwise} \end{cases} \tag{10}$$

$$Dur_i^{delay} = \begin{cases} 0, & \varepsilon_i \text{ } is \text{ } rigid \\ St_p - St_i - Dur_i, & \exists \varepsilon_q \text{ } satisfies \text{ } Next(\varepsilon_p, \varepsilon_i), \\ Time_{end}(\alpha_i) - St_i - Dur_i, & \text{otherwise} \end{cases} \tag{11}$$

where $Time_{st}(\alpha_i)$ and $Time_{end}(\alpha_i)$ are the user-defined earliest starting point and latest ending point of the event type α_i, respectively.

For brevity, we use the q-th time point ($q \in \{1,2,3,4\}$) to refer to the first event's starting point and ending point, and the second event's starting point and ending point in a TI, respectively. For the q-th time point in TI_n, its adjustable length under includes two aspects: the length that shortening or moving an event $Dur_{n,\,q,\,l}^{single}$, and the length that additionally shortens the duration after moving an event $Dur_{n,q,\,l}^{add\_short}$, where $Dur_{n,\,q,\,l}^{single}$ and $Dur_{n,q,\,l}^{add\_short}$ are defined to be:

$$Dur_{n,q,l}^{sungle} \begin{cases} 0, & q - \text{th action type is hold or abandon} \\ Dur_i^{shorten}, & q \in \{1,2\} \wedge l \in \{5,6,10,12,14,16\} \\ Dur_i^{advance}, & q \in \{1,2\} \wedge l \in \{7,11,17\} \\ Dur_i^{delay}, & q \in \{1,2\} \wedge l \in \{8,13,15\} \\ Dur_j^{shorten}, & q \in \{3,4\} \wedge l \in \{1,10,11,12,13\} \\ Dur_j^{advance}, & q \in \{3,4\} \wedge l \in \{2,14,15\} \\ Dur_j^{delay}, & q \in \{3,4\} \wedge l \in \{3,16,17\} \end{cases} \tag{12}$$

$$Dur_{n,q,\,l}^{add\_short} = \begin{cases} Dur_i^{shorten}, & q == 1 \wedge l \in \{8,13,15\} \\ Dur_i^{shorten}, & q == 2 \wedge l \in \{7,11,17\} \\ Dur_j^{shorten}, & q == 3 \wedge l \in \{3,16,17\} \\ Dur_j^{shorten}, & q == 4 \wedge l \in \{2,14,15\} \\ 0, & otherwise \end{cases}. \tag{13}$$

$Dur_{n,l}^{single}$, the adjustable length that shortens or moves events in TI_n under $c_{(l)}^{type}$, is defined as:

$$Dur_{n,l}^{single} = \begin{cases} 0, & l \in \{4,\ 9\} \\ Dur_{n,2,\,l}^{single} + Dur_{n,3,\,l}^{single}, & l \in \{1,\ 3,\ 6,\ 7,10,\ 11,\ 16,\ 17\}. \\ Dur_{n,1,\,l}^{single} + Dur_{n,4,\,l}^{single}, & l \in \{2,\ 5,\ 8,\ 12,\ 13,\ 14,\ 15\} \end{cases} \tag{14}$$

$Dur_{n,l}^{max\_adjust}$, the maximum length that events in TI_n can be adjusted under $c_{(l)}^{type}$, is given by:

$$Dur_{n,l}^{max\_adjust} = \begin{cases} Dur_{n,2,\ l}^{single} + Dur_{n,2,\ l}^{add\_short} + Dur_{n,3,\ l}^{single} + Dur_{n,3,\ l}^{add\_short}, & l \in \{1,\ 3,\ 4,\ 6,\ 7,\ 9, 10,\ 11,\ 16,\ 17\} \\ Dur_{n,1,\ l}^{single} + Dur_{n,1,\ l}^{add\_short} + Dur_{n,4,\ l}^{single} + Dur_{n,4,\ l}^{add\_short}, & \text{otherwise} \end{cases} \quad (15)$$

The minimum length to resolve TI_n with $c_{(l)}^{type}$, denoted as $Dur_{n,l}^{min\_require}$, is defined as:

$$Dur_{n,l}^{min\_require} = \begin{cases} 0,\ l \in \{4,\ 9\} \\ St_i + Dur_i + Dur_{ij} - St_j,\ l \in \{1,\ 3,\ 6,\ 7, 10,\ 11,\ 16,\ 17\}. \\ St_j + Dur_j + Dur_{ji} - St_i,\ l \in \{2,\ 5,\ 8,\ 12,\ 13,\ 14,\ 15\} \end{cases} \quad (16)$$

If $Dur_{n,l}^{max\_adjust}$ is no smaller than $Dur_{n,l}^{min\_require}$, then we say $c_{(l)}^{type}$ is effective in resolving TI_n, the time length to be adjusted is assigned in proportion to their adjustable length. Algorithm 1 formalizes this idea.

Principle 2: Adjustments should be consistent with the user's preference *The calendar's personalized nature dictates that the user's preference is the golden rule for addressing TIs. Assuming that the user's preference for the attribute values and behavior pattern is constant, the user should handle similar situations the same way.*

Algorithm 1: Allocate Time Length to a Strategy Type

```
Input:    A conflict : $TI_n$
          A strategy type : $c_{(l)}^{type}$ ($l \in \{1,\ 2,\ \ldots,\ 17\}$)
Output:   A binary flag of effectiveness: effect_flag
          Time length for current strategy type: $\boldsymbol{Dur}^{c\_time}$
1.  Calculate $Dur_{n,l}^{max\_adjust}$, $Dur_{n,l}^{min\_require}$
2.  If $Dur_{n,l}^{max\_adjust} < Dur_{n,l}^{min\_require}$:
3.      effect_flag $\leftarrow$ False, $\boldsymbol{Dur}^{c\_time} \leftarrow [-1, -1, -1, -1]$
4.  Else:
5.      effect_flag $\leftarrow$ True, $\boldsymbol{Dur}^{c\_time} \leftarrow []$
6.      For $q = 1, 2, 3, 4$ do:
7.          $Dur^{a\_time} \leftarrow Dur_{n,q,\ l}^{single} * \min\left(\frac{Dur_{n,l}^{min\_require}}{Dur_{n,l}^{single}},\ 1\right) + \frac{Dur_{n,q,\ l}^{add\_short}}{\sum_{w=1}^{4} Dur_{n,w,\ l}^{add\_short}} * \max\left(Dur_{n,l}^{min\_require} - Dur_{n,l}^{single}, 0\right)$
8.          $\boldsymbol{Dur}^{c\_time} \leftarrow \boldsymbol{Dur}^{c\_time} + Dur^{a\_time}$
9.  Return effect_flag, $\boldsymbol{Dur}^{c\_time}$
```

In the following, we introduce three approaches to resolve TIs: RSIC, RSSC, and GSSD. After encountering a TI, the system tries the above methods in turn until it gets a feasible strategy.

5.2.2. Replicating Solutions of Identical Cases (RSIC)

Two events are the *same* if they have the same values in attributes other than the starting point and ending point. Two TIs are the *same* if they have identical temporal classes and conflict lengths, and their events are correspondingly the same. Suppose TI_m consists of two events St_k and St_l ($St_k \leq St_l, m < n$). If TI_m, TI_n are the same TIs, then we say that TI_m occurs again, and the strategy adopted in TI_m should be followed according to Principle 2.

5.2.3. Referencing Solutions of Similar Cases (RSSC)

According to Principle 2, the system can solve a TI by referring to its *similar* TI, where the similarity between TIs depends on three aspects: (1) the importance difference between conflicting events, (2) the TCs, and (3) the feasibility of the target strategy type.

The importance difference between two events is calculated from two perspectives: the importance of individual decision attribute values and the order relation between them. For brevity, the values discussed below refer to the decision attribute values. We first introduce how to derive a value's importance. Values are divided into two categories: *common* values and *rare* values. A value is a common value if it occurs no less than a specified number of times, i.e., the corresponding threshold in $\theta^{rare} = \left(\theta^{etype}, \theta^{act},\ \theta^{host},\ \theta^{part},\ \theta^{loc},\ \theta^{perio}\right)$; otherwise, it is a rare value. A common value's importance is evaluated by its frequency in important events, whereas a rare value's importance is estimated using *hidden correlation* [46,47], as defined below.

Given two events, one has a value $value_i$ on attribute $attr_1$, and the other has a value $value_j$ on attribute $attr_2$. If two events have an identical value $value_k$ on one of the remaining attributes, then we say $value_k$ is one *co-involved value* for $value_i$ and $value_j$. *Hidden correlation* [46] between $value_i$ and $value_j$, denoted as HC_{ij}, is defined as:

$$HC_{ij} = \frac{\sum_{k\in\mathbf{K}}\left(\left(\overline{v(i|k)} - \overline{v(i)}\right)\left(\overline{v(j|k)} - \overline{v(j)}\right)\right.}{\sqrt{\sum_{k\in\mathbf{K}}\left(\overline{v(i|k)} - \overline{v(i)}\right)^2}\sqrt{\sum_{k\in\mathbf{K}}\left(\overline{v(j|k)} - \overline{v(j)}\right)^2}}, \tag{17}$$

where $\overline{v(i)}$ and $\overline{v(j)}$ are the frequency at which events containing $value_i$ and $value_j$ are important, respectively; $\overline{v(i|k)}$ and $\overline{v(j|k)}$ indicate the frequency of events involving $value_i$ and $value_k$, $value_j$ and $value_k$ are important, respectively. $\mathbf{K}$ is the set of all co-involved values for $value_i$ and $value_j$. Given a rare value, common values under the same attribute are *alternative values*. The extent to which an alternative value matches the event is assessed by the *transition probability* (TP) [46]:

$$TP_i = \textstyle\sum_{j\in\mathbf{CV}} HC_{ij}, \tag{18}$$

$$i' = \arg\max_i TP_i,\ \ s.t.\ TP_i \geq \theta^{trans}, \tag{19}$$

where i represents the alternative value $value_i$, $\mathbf{CV}$ is the set of common values in the event, θ^{trans} is the threshold at which an alternative value can be selected. A rare value's importance is temporarily substituted by that of $value_{i'}$ if there exists $value_{i'}$ obtained by Eq. (18) and is estimated by its frequency in important events otherwise. For two conflicting events, we construct $\boldsymbol{d}^{fre} = \left(d_1^{fre}, d_2^{fre}, \ldots,\ d_6^{fre}\right)$, where d_i^{fre} is the importance difference between their i-th values.

Next, we describe how to establish order relations between values. An *Importance preference relation* (IPR) is a special case of strict partial order on a set of events, represented by the symbol $\succ$ (See Appendix A for proof) [48–50]. Given an important event ε_a and a normal event ε_b, we denote by $\varepsilon_a \succ \varepsilon_b$ the fact that the user prefers ε_a to ε_b, which is equivalent to $\left(\epsilon_a,\ \alpha_a, p_a^{host}, p_a^{part}, \mathcal{L}_a^{name}, \zeta_a\right) \succ \left(\epsilon_b,\ \alpha_b, p_b^{host}, p_b^{part}, \mathcal{L}_b^{name}, \zeta_b\right)$.

Property 1. *Provided that the operations are meaningful for an IPR they are applied to, the IPR still holds if adding or subtracting the same value to both sides of the relation.*

According to Property 1, we introduce $Filter(\varepsilon_a, \varepsilon_b)$ to filter the duplicate information by replacing the same value in ε_a as in ε_b with "null". We use IPR^L and IPR^R to denote the left and right-hand side in an IPR. The union of the left or right-hand side of two IPRs, represented by the symbol $\cup$, is a tuple combining elements of the corresponding positions.

Theorem 1. *Given IPR_i and IPR_j, the strict partial order still holds for the union of IPR_i^L and IPR_j^L, and the union of IPR_i^R and IPR_j^R, which is denoted as $IPR_i \cup IPR_j$:*

$$IPR_i^L \cup IPR_j^L \succ IPR_i^R \cup IPR_j^R. \tag{20}$$

If the left and right-hand sides correspond the same, then we say IPR_i is equal to IPR_j, denoted as $IPR_i = IPR_j$. The *original information* of IPR_i, denoted as $OI(IPR_i)$, takes the value null if there exists historical relations $IPR_m, IPR_n, \ldots, IPR_k$ whose union equals to IPR_i and takes the value itself otherwise.

We use $IPR_i^L \setminus IPR_j^R$ to denote relative complement of IPR_i^L in IPR_j^R, which is the tuple of elements in IPR_i^L but not in IPR_j^R.

Theorem 2. *The extra information inferred from IPR_i and IPR_j, denoted as $EI(IPR_i, IPR_j)$, is defined to be:*

$$EI\left(IPR_i, IPR_j\right) = \begin{cases} IPR_i^L \setminus IPR_j^R \succ IPR_i^R \setminus IPR_j^L, & \text{subValue}\left(IPR_j^L,\ IPR_i^R\right) \wedge \text{subValue}\left(IPR_j^R, IPR_i^L\right) \\ null, & otherwise \end{cases}, \tag{21}$$

where $subValue\left(IPR_j^L,\ IPR_i^R\right)$ takes the value True if all not-null elements of IPR_j^L are also elements of IPR_i^R, False otherwise.

For a tuple pair $\langle tp_1, tp_2\rangle$, IPR_i is a regular subIPR if it fits $\text{subValue}\left(IPR_i^L,\ tp_1\right) \wedge \text{subValue}\left(IPR_i^R,\ tp_2\right)$; IPR_i is an inverse subIPR if it meets $\text{subValue}\left(IPR_i^L,\ tp_2\right) \wedge \text{subValue}\left(IPR_i^R,\ tp_1\right)$.

IPR profile union. An IPR profile $\mathbf{\Gamma}_n$ is the transitive closure of all IPRs in $\mathbf{OIPR}_n$ and $\mathbf{EIPR}_n$, where $\mathbf{OIPR}_n$ and $\mathbf{EIPR}_n$ are sets of all original information and extra information derived from TI_{n-1} to TI_n, respectively [48]. An IPR profile union $\mathbf{\Omega}_n$ is the transitive closure of *IPR*s in IPR profile $\mathbf{\Gamma}_n$ and past *IPR*s in IPR profile union $\mathbf{\Omega}_{n-1}$, which is defined to be:

$$\mathbf{\Omega}_n = \begin{cases} \mathbf{\Gamma}_n, & n = 1 \\ \mathbf{\Gamma}_n \cup \mathbf{\Omega}_{n-1}, & otherwise \end{cases}. \tag{22}$$

We quantify the order relation between ε_i and ε_j by the following procedure:

1. Get a tuple pair $Filter\left(\varepsilon_i, \varepsilon_j\right), Filter\left(\varepsilon_j, \varepsilon_i\right)$ by filtering out duplicate information.
2. Look for regular subIPRs and inverse subIPRs of $Filter\left(\varepsilon_i, \varepsilon_j\right), Filter\left(\varepsilon_j, \varepsilon_i\right)$.
3. Construct $\boldsymbol{d}^{subIPR} = \left(d_1^{subIPR},\ d_2^{subIPR}, \ldots, d_6^{subIPR}\right)$ for each subIPR. For a subIPR IPR_k, d_i^{subIPR} is defined as:

$$d_i^{subIPR} = \begin{cases} 0, & i-\text{th element of } IPR_k^L \text{ is null} \\ U/N_{IPR_k}, & IPR_k \textit{ is a regular subIPR} \\ -U/N_{IPR_k}, & IPR_k \textit{ is an inverse subIPR} \end{cases}, \tag{23}$$

where U is a hyper-parameter, N_{IPR_k} is the number of not-null values in the IPR_k^L.

Construct $\boldsymbol{d}_n^{IPR} = \left(d_1^{IPR},\ d_2^{IPR}, \ldots, d_6^{IPR}\right)$, where d_i^{IPR} is the largest value on the *i*-th element obtained from all subIPRs.

Dvalue. We use *dvalue* $\boldsymbol{d}_n = \boldsymbol{d}_n^{fre} + \boldsymbol{d}_n^{IPR}$ to indicate the importance difference between two events in TI_n. Let $\mathbf{M}_n = (\boldsymbol{d}_1,\ \boldsymbol{d}_2, \ldots, \boldsymbol{d}_{n-1})$ denote the matrix that consists of dvalues for past *n-1* TIs. Given $\boldsymbol{d}_n$ and $\mathbf{M}_n$, $\mathbf{X} = (x_1,\ x_2, \ldots, x_{n-1})^T$ denote the coefficients corresponding to $\mathbf{M}_n$, and is defined as follows through the *sparse decomposition* (SD) process [1,51–53]:

$$\min_{\mathbf{X}} \|\mathbf{X}\|_1, \text{ s.t. } \mathbf{M}_n\mathbf{X} = \boldsymbol{d}_n, \tag{24}$$

where $\|\cdot\|_1$ is the L_1-regularization and gives as few non-zero coefficients as possible.

We then introduce how the system utilizes the previous strategies. The historical strategies can be used directly or after being modified by Trans$(\cdot)$, which is a function that transforms a strategy type using the following steps: First, swap action types on different events. Second, swap action types on each event's starting point and ending point and reverse adjustment directions. Two strategy types $c_{(i)}^{type}$ and $c_{(j)}^{type}$ are *inverse strategy types*, denoted as $c_{(i)}^{type} \leftrightarrow c_{(j)}^{type}$, if the following condition holds:

$$c_{(i)}^{type} \leftrightarrow c_{(j)}^{type}, \text{s.t.} \mathrm{Trans}\left(c_{(i)}^{type}\right) = c_{(j)}^{type}, \mathrm{Trans}\left(c_{(j)}^{type}\right) = c_{(i)}^{type}. \tag{25}$$

The set of all inverse strategy types is $\left\{c_{(1)}^{type} \leftrightarrow c_{(6)}^{type}, c_{(2)}^{type} \leftrightarrow c_{(8)}^{type}, c_{(3)}^{type} \leftrightarrow c_{(7)}^{type}, c_{(4)}^{type} \leftrightarrow c_{(9)}^{type}, c_{(10)}^{type} \leftrightarrow c_{(10)}^{type}, c_{(11)}^{type} \leftrightarrow c_{(16)}^{type}, c_{(12)}^{type} \leftrightarrow c_{(12)}^{type}, c_{(13)}^{type} \leftrightarrow c_{(14)}^{type}, c_{(15)}^{type} \leftrightarrow c_{(17)}^{type}\right\}$.

We say TI_n and TI_m have *inverse flags* if $\zeta_i \neq \zeta_j$ and ζ_i, ζ_j are identical with ζ_l, ζ_k, respectively. We train two calibrated support vector machine (SVM) [1,54,55] f_R and f_I on $\left\{\left(\boldsymbol{x}_{i,\,j}, y_{i,\,j}^{regular}\right)\right\}$ and $\left\{\left(\boldsymbol{x}_{i,\,j}, y_{i,\,j}^{inverse}\right)\right\}$, where i, j denotes TI_i, TI_j, $i \neq j$. $\boldsymbol{x}_{i,\,j}$ is the training sample that contains TCs, feasible strategy types, and flags within TI_i and TI_j. $y_{i,\,j}^{regular}$ and $y_{i,\,j}^{inverse}$ are true if feasible strategy types, flags, and the user's strategy type of TI_i and TI_j are all the same or inverse, respectively; false otherwise.

The *regular TC* and *inverse TC* between TI_m and TI_n, denoted as $\mathrm{TC}_{regular}(TI_m,\ TI_n)$ and $\mathrm{TC}_{inverse}(TI_m,\ TI_n)$, are given by:

$$\mathrm{TC}_{regular}(TI_m,\ TI_n) \equiv_{\mathrm{def}} \left[f_R(\boldsymbol{x}_{m,\,n}) \geq \theta^{TC} \vee TC_m == TC_n\right], \tag{26}$$

$$\mathrm{TC}_{inverse}(TI_m,\ TI_n) \equiv_{\mathrm{def}} \left[f_I(\boldsymbol{x}_{m,\,n}) \geq \theta^{TC}\right], \tag{27}$$

where TC_m and TC_n are temporal classes of TI_m and TI_m, respectively; θ^{TC} is the threshold for the SVM model.

Let $\mathrm{STI}_{rsc}(TI_m, TI_n)$ and $\mathrm{STI}_{isc}(TI_m, TI_n)$ denote the *regular similar case* and *inverse similar case* between TI_m and TI_n:

$$\mathrm{STI}_{rsc}(TI_m, TI_n) \equiv_{\mathrm{def}} \left[c_m^{user}\ is\ feasible \wedge x_m \geq \theta^{SD} \wedge \mathrm{TC}_{regular}(TI_m,\ TI_n)\right], \tag{28}$$

$$\mathrm{STI}_{isc}(TI_m, TI_n) \equiv_{\mathrm{def}} \left[\mathrm{Trans}(c_m^{user})\ is\ feasible \wedge -x_m \geq \theta^{SD} \wedge \mathrm{TC}_{inverse}(TI_m,\ TI_n)\right]. \tag{29}$$

Similar TIs between TI_m and TI_n, denoted as $\mathrm{STI}(TI_m, TI_n)$, is defined to be:

$$\mathrm{STI}(TI_m, TI_n) \equiv_{\mathrm{def}} [\mathrm{STI}_{rsc}(TI_m, TI_n) \vee \mathrm{STI}_{isc}(TI_m, TI_n)]. \tag{30}$$

We use **STI** to denote all similar TIs of TI_n. Let x_m^l, o_n^l denote score on $c_{(l)}^{type}$ obtained by TI_m and all similar TIs of TI_n, respectively. x_m^l, o_n^l are given by:

$$x_m^l = \begin{cases} |x_m|, & \left(\mathrm{STI}_{rsc}(TI_m, TI_n) \wedge c_m^{user} == c_{(l)}^{type}\right) \vee \left(\mathrm{STI}_{isc}(TI_m, TI_n) \wedge \mathrm{Trans}(c_m^{user}) == c_{(l)}^{type}\right) \\ 0, & otherwise \end{cases}, \tag{31}$$

$$o_n^l = \textstyle\sum_{m \in \mathbf{STI}} x_m^l. \tag{32}$$

$\hat{y}_n^l$, the possibility that the system recommends $c_{(l)}^{type}$, is given by:

$$\hat{y}_n^l = \frac{\exp\left(o_n^l\right)}{\sum_{k=1}^{17} \exp\left(o_n^k\right)}. \tag{33}$$

Let $\hat{\boldsymbol{y}}_n = \left(\hat{y}_n^1, \hat{y}_n^2, \ldots, \hat{y}_n^{17}\right)$ denote the *prediction probability distribution* of TI_n. Let $\boldsymbol{y}_n = \left(y_n^1, y_n^2, \ldots, y_n^{17}\right)$ denote the *label probability distribution*, where y_n^l is 1 if the user adopted $c_{(l)}^{type}$ and 0 otherwise. The system suggests the strategy type with the highest prediction probability returned by $Z(\hat{\boldsymbol{y}}_n)$:

$$Z(\hat{\boldsymbol{y}}_n) = c_{(l)}^{type}, \text{ where } l = \arg\max_k \hat{y}_n^k. \tag{34}$$

Algorithm 2 describes how to find similar TIs and generate strategies based on KB$(n, \mathbf{w}_n)$, where KB$(n, \mathbf{w}_n)$ specifies the contents of KB correspond to TI_n and $\mathbf{w}_n = \left(\theta_n^{rare}, \theta_n^{matrix}, \theta_n^{trans}, \theta_n^{SD}, \theta_n^{TC}\right)$ assigns values of thresholds.

Algorithm 2: Resolve Temporal Inconsistencies by Similar Circumstances

```
Input:        A conflict : $TI_n$
              Knowledge base : KB$(n, \mathbf{w}_n)$
Output:       System's suggestion : $C_{\text{sys}}$
Initialization:
    $C_{\text{sys}} \leftarrow$ [["hold", −1], ["hold", −1], ["hold", −1], ["hold", −1]]  # -1 indicates an invalid action
    STI← [] # indexes of all similar TIs
    $\boldsymbol{Dur}^{all\_c\_time} \leftarrow$ [] # adjustment lengths for all possible strategy types
1.  Get $\boldsymbol{d}_n^{fre}$ and $\boldsymbol{d}_n^{IPR}$, $\boldsymbol{d}_n \leftarrow \boldsymbol{d}_n^{fre} + \boldsymbol{d}_n^{IPR}$
2.  Sparse decomposition: $\min_{\mathbf{X}} \|\mathbf{X}\|_1$, s.t. $\mathbf{M}_n\mathbf{X} = \boldsymbol{d}_n$
3.  For $i = 1, 2, \ldots, n-1$ do:
4.      If $x_i \geq \theta^{SD}$ and TC$_{regular}(TI_i, TI_n)$:
5.          Get effect_flag and $\boldsymbol{Dur}^{c\_time}$ of $c_i^{user}$ # by Algorithm 1
6.      Elif $-x_i \geq \theta^{SD}$ and TC$_{inverse}(TI_i, TI_n)$:
7.          Get effect_flag and $\boldsymbol{Dur}^{c\_time}$ of Trans$\left(c_i^{user}\right)$ # by Algorithm 1
8.      Else:
9.          effect_flag $\leftarrow$ False
10.     If effect_flag is True:
11.         Update STI and $\boldsymbol{Dur}^{all\_c\_time}$
12. If exists similar TI(s) with $TI_n$:
13.     Generate prediction probability distribution $\hat{\boldsymbol{y}}_n$ # by Eq. (33)
14.     Choose a strategy type $c_{(l)}^{type}$ # by Eq. (34)
15.     Get corresponding time length of $c_{(l)}^{type}$, generate suggestion $C_{\text{sys}}$
16. Return $C_{\text{sys}}$
```

5.2.4. Generating Solutions with Strategy Distribution (GSSD)

The generic solution GSSD exploits the system-user strategy matrix $\mathbf{M}^{\text{ctype}}$:

$$\mathbf{M}^{\text{ctype}} = \begin{bmatrix} n_{1,1} & \cdots & n_{1,17} \\ \vdots & \ddots & \vdots \\ n_{17,1} & \cdots & n_{17,17} \end{bmatrix} = \left(n_{i,j}\right) \in \mathbb{R}^{17\times 17}, \tag{35}$$

where $n_{i,j}$ is the number of times the system suggested $c_{(i)}^{type}$ and the user adopt $c_{(j)}^{type}$. The main diagonal elements indicate the number of times the system and the user have agreed, while elements outside represent discrepancies. Let $\mathrm{P}^j = \frac{n_{j,j}}{\sum_{k=1}^{17} n_{j,k}}$ denote the *recommendation probability* that the system suggests $c_{(j)}^{type}$ $\left(\mathrm{P}^j = 0 \textit{ when } n_{j,j} = 0\right)$. The matrix is gradually populated, and the information it carries is partial and incomplete if the number of TIs is less than the threshold θ^{matrix}. In this case, the system randomly chooses one from all feasible strategies. Otherwise, the system suggests the strategy with the highest recommendation probability. Algorithm 3 formalizes the above process.

Algorithm 3: Generate a suggestion with strategy distribution

```
Input:   A conflict : $TI_n$
         The system-user strategy matrix: $\mathbf{M}^{\text{ctype}}$
         The threshold for $\mathbf{M}^{\text{ctype}}$: $\theta^{matrix}$
Output:  System's suggestion : $C_{\text{sys}}$
Initialization:
      $\boldsymbol{C}^{poss} \leftarrow []$ # all possible strategy types for $TI_n$
      $\boldsymbol{Dur}^{all\_c\_time} \leftarrow []$ # adjustment lengths for all possible strategy types
1.    For $l = 1, 2, \ldots, 17$ do: # find all feasible strategy types
2.        Get effect_flag and $\boldsymbol{Dur}^{c\_time}$ of $c_{(l)}^{type}$ # by Algorithm 1
3.        **If** effect_flag is True:
4.            Update $\boldsymbol{C}^{poss}$ and $\boldsymbol{Dur}^{all\_c\_time}$
5.    If $n \geq \theta^{matrix}$:
6.        $l \leftarrow \arg\max_j \mathrm{P}^j$, where $\mathrm{P}^j = \frac{n_{j,j}}{\sum_{k=1}^{17} n_{j,k}}$, $c_{(j)}^{type} \in C^{poss}$ $\left(\mathrm{P}^j = 0 \ when \ n_{j,j} = 0\right)$
7.    Else:
8.        Randomly choose a strategy type $c_{(l)}^{type} \in \boldsymbol{C}^{poss}$
9.    Get corresponding time length of $c_{(l)}^{type}$, generate system suggestion $C_{\text{sys}}$
10.   Return $C_{\text{sys}}$
```

5.3. Incremental Performance Improvement through Knowledge Refinement

As depicted in Figure 3, the system uses three learning components LC_1, LC_2 and LC_3 to refine the existing knowledge, where:

- LC_1 is the component that records and revises the event-related knowledge. When LC_1 is invoked, it first records the new event. After that, it derives original and extra information from the IPRs generated, based on the new event and historical events, and then updates the IPR profile union accordingly. Event accumulation and IPR profile union updates lead to better estimation of a TI's dvalue. Hence a subsequent update on historical TIs' dvalues is in order, which provides a better basis for using RSSC to resolve TIs.
- LC_2 is the component that records and revises the TI-related knowledge. When LC_2 is invoked, it records the new TI and maintains $\mathbf{M}^{\text{ctype}}$ by aggregating the system's and user's decisions over past TIs. As the number of historical TIs grows, the probability that the system can find the same or similar TI of the current TI increases, and thus the probability that the system uses RSIC and RSSC methods increases. The refinement of $\mathbf{M}^{\text{ctype}}$ helps the system gradually approach the user's decision preference in the genera case, which in turn makes the strategy generated by GSSD more acceptable to the user.
- LC_3 is the component that optimizes models. When LC_3 is triggered, it retrains the calibrated SVM and updates parameters w. The updates of calibrated SVM and parameters $\mathbf{w}$ enable the system to determine the similarity of TIs more accurately, thereby improving the success rate of solving TIs by RSSC.

We next define how the system learns from strategy discrepancies. If action types of the system's and the user's strategies are correspondingly the same, then we say they agree on the solution of TI_n, denoted as $c_n^{sys} = c_n^{user}$. Otherwise, we say there is a *strategy discrepancy* due to the incompatible action types, denoted as $c_n^{sys} \cdot c_n^{user}$. c_n^{sys} is obtained from $\hat{\boldsymbol{y}}_n$, which in turn is generated by the system model f:

$$f: \ TI_n \times \mathrm{KB}(n, \ \mathbf{w}_n) \rightarrow \hat{\boldsymbol{y}}_n. \tag{36}$$

The strategy discrepancy on TI_n is evaluated by weighted cross-entropy $WCE(\boldsymbol{y}_n, \hat{\boldsymbol{y}}_n)$:

$$\begin{aligned} WCE(\boldsymbol{y}_n, \hat{\boldsymbol{y}}_n) &= -\sum_{j=1}^{17}\left(y_n^j log\hat{y}_n^j + \left(1 - \mathrm{P}^j\right)\left(1 - y_n^j\right)log\left(1 - \hat{y}_n^j\right)\right) \\ &= WCE(\boldsymbol{y}_n, f(TI_n, \mathrm{KB}(n,\ \mathbf{w}_n))) \end{aligned} \tag{37}$$

Since the data corresponding to TI_n is deterministic, we address the knowledge deficiency from the parameter perspective. We present *loss* to measure the strategy discrepancies achieved on past n TIs with the specific parameters $\mathbf{w}^k \in \mathbf{W}$ ($\mathbf{W}$ is the set of all possible values of $\mathbf{w}$), which is defined as:

$$loss\left(n,\ \boldsymbol{w}^k\right) = \frac{1}{n}\sum_{i=1}^{n} WCE\left(\boldsymbol{y}_i, f\left(TI_i, KB\left(i,\ \boldsymbol{w}^k\right)\right)\right). \tag{38}$$

The strategy discrepancy leads the system to update parameters to the ones that obtain minimal loss:

$$\mathbf{w}_n^* = \arg\min_{\mathbf{w}^i} loss\left(n, \mathbf{w}^i\right). \tag{39}$$

6. Results

6.1. Experimental Setup and Metrics

To validate the effectiveness of our system, we carried out experiments on resolving TIs involving five participants in total (four males and one female). Generally, the number of events ranges from 3359 to 5110, and the amount of TIs ranges from 489 to 853. Table 2 gives participant information. See Appendix B for details of experimental data.

Table 2. Participant information.

User	Age	Sex	Profession	Educational Background
1	28	M	PhD student	Postgraduate
2	29	M	IT engineer	Bachelor
3	33	M	SIPI engineer	Bachelor
4	28	M	PhD student	Postgraduate
5	20	F	College student	High school

We adopt a heuristic method Base algorithm (BA), SC-based methods, and a traditional ML model SVM@n as benchmarks: BA randomly suggests a choose a strategy from the feasible ones. SC-based methods include SC_V1 (deals with TIs by RSIC and GSSD), SC_V2 (additionally employs the RSSC), and SC_V3 (the SmartCalendar system with complete components); SVM@n is an SVM model that starts from the $80-th$ TI and retrains every 20 more TIs.

We propose strategy type acceptance rate (SA) to measure the consistency between system recommendations and user decisions, which is defined to be:

$$\frac{1}{\mathrm{MA}}\sum_{t=0}^{\mathrm{MA}}\mathbb{I}\left(c_{n-t}^{sys} = c_{n-t}^{user}\right), \tag{40}$$

where $\mathbb{I}(\cdot)$ is an indicator function that takes the value 1 if the statement is true and takes 0 otherwise. We smooth out short-term fluctuations and emphasize longer-term trends through a moving average with a size of 200. A moving average is a series of averages of different fixed-size subsets of the complete dataset.

6.2. Experimental Results

First, we compare the distance between the SC-based methods' strategies and the user's strategies in terms of WCE. Figure 4 illustrates that the WCE of SC_V1 shows a steady decline. SC_V2 followed a similar trend, but the result was slightly lower than that of SC_V1, which validates the effectiveness of the RSSC module. Meanwhile, SC_V3 witnessed a significant fall. SC_V3′s lead over SC_V2 verifies the validity of knowledge refinements. Refer to Appendix B for a discussion of convergence.

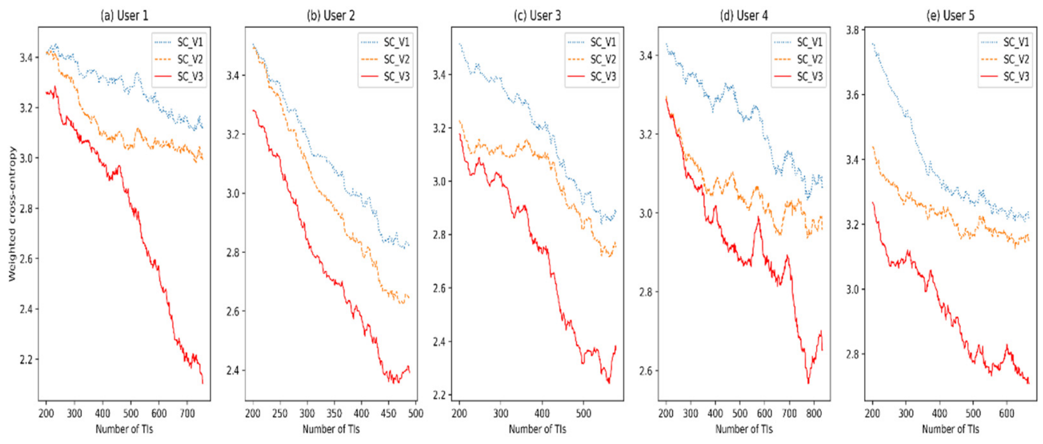

Figure 4. Weighted cross-entropy for five users.

Next, we evaluate the adaptability to user preferences of our approach, BA, and SVM@n concerning SA. Figure 5 shows that BA and SVM@n remained at a lower level and did not show an ascending trend on SA with increasing data, which indicates that neither the heuristic nor the traditional ML algorithm can be directly applied to the calendar scenario. In contrast, SC-based methods have steadily increased in SA. The SA of SC_V2 is higher than SC_V1, indicating that the system obtains a more accurate recommendation using RSSC. SC_V3 increased sharply and finally led SC_V2 by 26.5%, 9%, 15%, 10%, and 23% among five users, proving that the system gains greater adaptability to the dynamic and complex environment than other approaches.

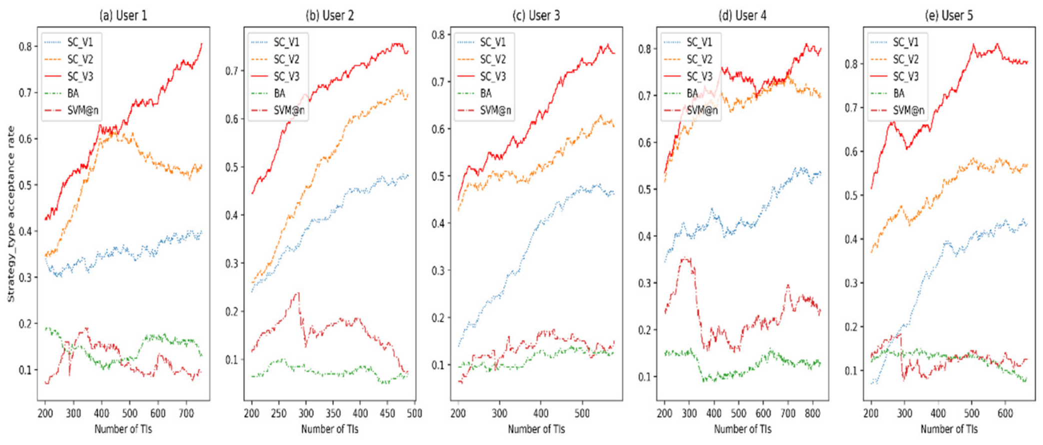

Figure 5. Strategy type acceptance rate for five users.

The system incrementally improves its performance through continuous knowledge refinement. Knowledge refinements can be classified as data accumulation and meta-knowledge updates. Since the occurrence of data is beyond the system's control, we focus on how the system achieves incremental performance improvement through meta-knowledge refinement.

We present how the meta-knowledge has been updated over time from the perspective of IPRs and parameters. As shown in Figure 6, the number of IPRs experienced a rapid surge in the first 20% of data and continued with a gradual increase. This is because, as the data accumulates, the probability that an incoming event has ever occurred in history increases. Accordingly, the average amount of origin information and additional information it can produce gradually decreases.

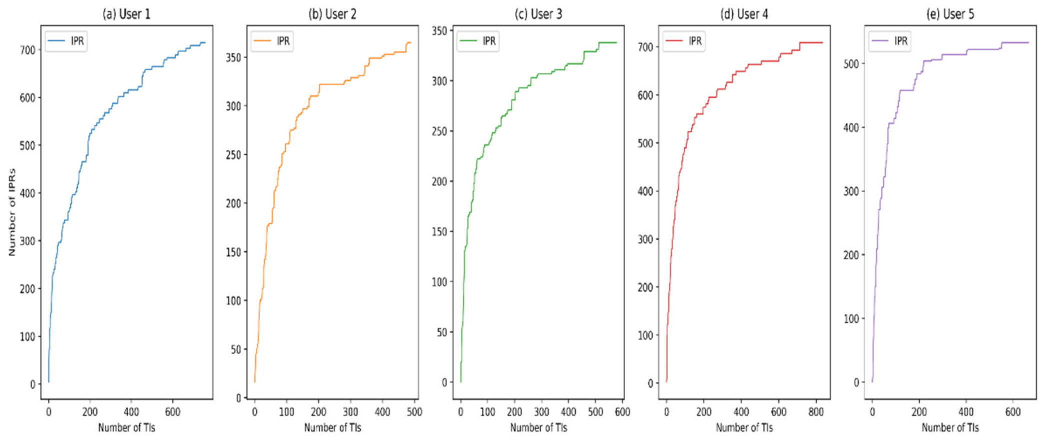

Figure 6. IPR updates for five users.

Figure 7 delineates the update of **w** over time for each user. The y-axis represents all sets of values of **w** and the star marker highlights the moment when **w** changes. For each user, parameters were updated multiple times and ended up with different values. The above result shows that no model works for every user at every stage, demonstrating the need for continuous learning.

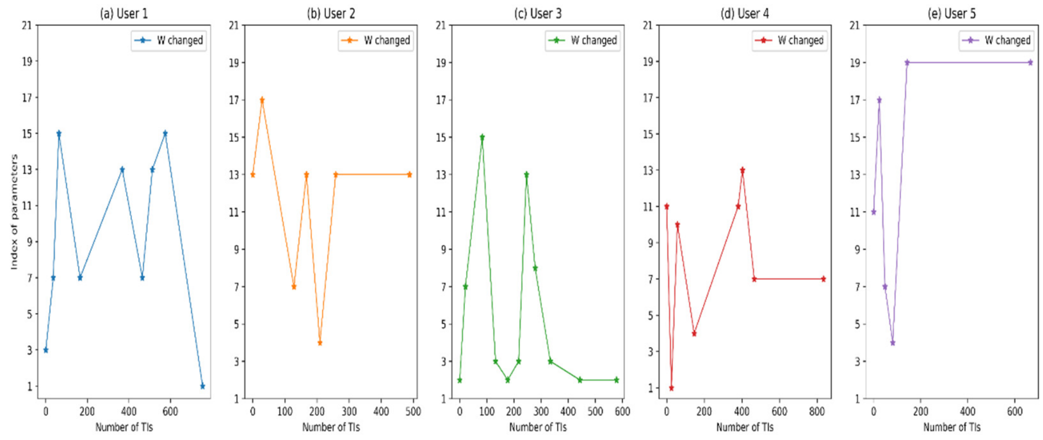

Figure 7. Parameter updates for five users.

We assess the performance of RSIC, RSSC, and GSSD in SC_V3 in terms of the percentage of successfully solved TIs. Table 3 depicts that for each 10% increase in data, RSSC obtained the highest improvement rate, followed by RSIC, and GSSD with the lowest. The growth in RSIC indicates that the data augmentation provides a good basis for the system to refer to history. In addition, the refinement in meta-knowledge further improves the performance of RSSC and GSSD, which is why SC_V3 outperforms SC_V2 in Figures 4 and 5.

Table 3. The increase in the percentage of TIs successfully solved by RSIC, RSSC, and GSSD for each 10% increase in data.

User	RSIC	RSSC	GSSD
1	0.00670	0.01842	0.01060
2	0.01228	0.01340	0.00167
3	0.00949	0.01060	0.00725
4	0.00614	0.00670	0.00446
5	0.00223	0.01507	0.00111

Figure 8 shows that the gap between RSSC and GSSD continues to narrow as data increases, and RSSC outraced GSSD on Users 2, 3, and 5. The fact that the system prefers RSSC over GSSD indicates that the system gradually approaches the user's decision preferences through continuous knowledge refinements. And RSSC is playing a more and more important role in resolving TIs. In summary, continuous knowledge refinement is indispensable and effective for a PeLA to acquire self-improvement capability.

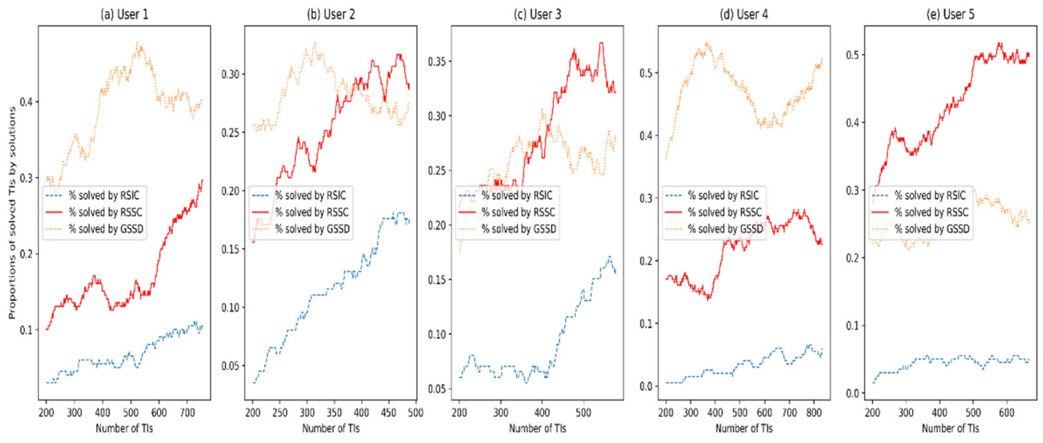

Figure 8. Percentage of TIs successfully solved by RSIC, RSSC, and GSSD for five users.

6.3. Discussions

STEP PL emphasizes incremental task performance improvement through sequential stimuli-driven learning episodes. In contrast to *STEP* PL, learning processes in LL and NEL are neither triggered by stimuli nor oriented toward incremental performance improvement, which makes these two paradigms inapplicable for developing an intelligent calendar system that can gradually adapt to the user's preference.

7. Conclusions and Future Work

In contrast to the one-off metaphor, *STEP* PL emphasizes accomplishing incremental task performance through continuous knowledge refinements. In this work, we investigated the problems that a long-term event scheduling system encounters when resolving TIs, and proposed a system SmartCalendar based on the *STEP* PL. First, we formally define events, complete temporal classes, and strategies. Then, we theoretically model the SmartCalendar system to detect, handle, and learn from TIs, enabling it to consistently improve its problem-solving performance. Finally, we conduct experiments to validate our approach and demonstrate that our approach outperforms the comparison algorithms in terms of strategy type acceptance rate and self-improvement ability. The collected dataset and prototype system are open source on the GitHub repository.

Future work can be pursued in the following directions:

1. *Expand the dataset*. The current dataset contains one year of events and TIs for five users. In future work, we plan to expand the dataset in terms of the number of

users and the time span to validate the system's adaptability to a more complex environment.

2. *Classify learning stimuli*. Learning stimuli play an essential role in a PeLA's performance improvement. In future work, we plan to classify learning stimuli from the knowledge perspective, which provides guidelines for people to develop PeLAs in other fields.

Author Contributions: Data curation, X.S., H.Q.; Writing— original draft, J.T.; Writing—review and editing, D.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research was partially funded by Macau Science and Technology Foundation under Grant 0025/2019/AKP.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The dataset is available at https://github.com/hensontang9/Temporal_conflicts, (accessed on 14 September 2022).

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

Appendix A.1. Proof of Strict Partial Order

For all $\varepsilon_i, \varepsilon_j,\ \varepsilon_k \in \boldsymbol{\varepsilon}$, a binary relation $\succ$ on a set ε satisfies the following conditions:
(Irreflexivity): ε_i cannot be more important than itself
(Transitivity): if the user prefers ε_i to ε_j and prefers ε_j to ε_k, then it is easy to know that ε_i is preferred to ε_k(Asymmetry): if the user prefers ε_i to ε_j, then ε_j cannot be preferred to ε_i
Therefore, a binary relation $\succ$ on a set $\boldsymbol{\varepsilon}$ is a strict partial order.

Appendix A.2. Proof of Theorem 1

Given IPR_i and IPR_j, relation still holds by adding IPR_j^L on both sides of IPR_i according to property 1, then we get

$$IPR_i^L \cup IPR_j^L \succ IPR_i^R \cup IPR_j^L$$

similarly, we have

$$IPR_j^L \cup IPR_i^R \succ IPR_j^R \cup IPR_i^R$$

Hence, we obtain the following by transitivity

$$IPR_i^L \cup IPR_j^L \succ IPR_i^R \cup IPR_j^L \succ IPR_i^R \cup IPR_j^R$$

Q.E.D.

Appendix A.3. Proof of Theorem 2

Given IPR_i and IPR_j that satisfy subValue$\left(IPR_j^L,\ IPR_i^R\right)$ and subValue$\left(IPR_j^R,\ IPR_i^L\right)$. By property 1, we get

$$IPR_j^L \cup IPR_i^R \setminus IPR_j^L \succ IPR_j^R \cup IPR_i^R \setminus IPR_j^L$$

which equals

$$IPR_i^R \succ IPR_j^R \cup IPR_i^R \setminus IPR_j^L$$

with transitivity, we get

$$IPR_i^L \succ IPR_i^R \succ IPR_j^R \cup IPR_i^R \setminus IPR_j^L$$

by assumption we have

$$\text{subValue}\left(IPR_j^L,\ IPR_i^R\right)$$

by property 1, we subtract IPR_i^L on both sides

$$IPR_i^L \setminus IPR_j^R \succ IPR_i^R \setminus IPR_j^L$$

Q.E.D.

Appendix B

It is very difficult and time-consuming to collect real calendar data: either let the user record each day's events, then the time span of the required experimental data determines the time needed to collect the data, or let the user recall historical events, which is almost impossible due to the forgetfulness of human beings. So, we used an indirect method to collect experimental data: First, users input their non-duplicate daily events. Next, the system generates a 365-day temporal sequence, and each event is scheduled for specific days according to its periodicity. The user is asked to enter solutions to the temporally conflicting events.

The experimental results show that the system does not converge on the WCE metric, which is related to two factors: meta-knowledge and data. Figure 7 reveals that 81.7% of the model parameter updates occurred on the first 400 TIs. This is because, on the one hand, fewer strategy discrepancies occur afterward, and on the other hand, the system is becoming more aware of user preferences, and the current parameters are already the best ones for a given data set. Nevertheless, Figures 5 and 8 delineate that the system's performance grows steadily. Therefore, data insufficiency is the main reason for the system not converging. However, increasing data volume brings a new problem: The longer the time span of the dataset, the higher the probability that the user's preference will change. Thus, we choose a time span of one year to balance the dataset being too small and causing preference changes. Assuming that the user preference remains unchanged, the system performance regarding WCE on more data afterward is expected to maintain the downward trend, then eventually converge.

References

1. Zhou, Z.-H. *Machine Learning*; Springer Nature: Berlin/Heidelberg, Germany, 2021.
2. Zhang, D. From One-off Machine Learning to Perpetual Learning: A STEP Perspective. In Proceedings of the 2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Miyazaki, Japan, 7–10 October 2018; pp. 17–23.
3. Jordan, M.I.; Mitchell, T.M. Machine Learning: Trends, Perspectives, and Prospects. *Science* **2015**, *349*, 255–260. [CrossRef] [PubMed]
4. Buckler, B. *A Learning Process Model to Achieve Continuous Improvement and Innovation*; The Learning Organization: Bingley, UK, 1996.
5. Mitchell, T.; Cohen, W.; Hruschka, E.; Talukdar, P.; Yang, B.; Betteridge, J.; Carlson, A.; Dalvi, B.; Gardner, M.; Kisiel, B.; et al. Never-Ending Learning. *Commun. ACM* **2018**, *61*, 103–115. [CrossRef]
6. Chen, Z.; Liu, B. Lifelong Machine Learning. *Synth. Lect. Artif. Intell. Mach. Learn.* **2018**, *12*, 1–207.
7. Silver, D.L.; Yang, Q.; Li, L. Lifelong Machine Learning Systems: Beyond Learning Algorithms. In Proceedings of the 2013 AAAI Spring Symposium Series, Palo Alto, CA, USA, 25–27 March 2013.
8. Chen, Z.; Liu, B. Topic Modeling Using Topics from Many Domains, Lifelong Learning and Big Data. In Proceedings of the International Conference on Machine Learning, PMLR, Bejing, China, 22–24 June 2014; pp. 703–711.
9. Mitchell, T.; Fredkin, E. Never-Ending Language Learning. In Proceedings of the 2014 IEEE International Conference on Big Data (Big Data), Washington, DC, USA, 27–30 October 2014; p. 1.
10. Zhang, D.; Gregoire, E. Learning through Overcoming Inconsistencies. In Proceedings of the 2016 27th International Workshop on Database and Expert Systems Applications (DEXA), Porto, Portugal, 5–8 September 2016; pp. 121–128.
11. Zhang, D. Learning through Overcoming Temporal Inconsistencies. In Proceedings of the 2015 IEEE 14th International Conference on Cognitive Informatics & Cognitive Computing (ICCI*CC) Beijing, China, 6–8 July 2015; pp. 141–148.
12. Zhang, D. Learning through Explaining Observed Inconsistencies. In Proceedings of the 2014 IEEE 13th International Conference on Cognitive Informatics and Cognitive Computing, London, UK, 18–20 August 2014; pp. 133–139.
13. Zhang, D. Learning through Overcoming Incompatible and Anti-Subsumption Inconsistencies. In Proceedings of the 2013 IEEE 12th International Conference on Cognitive Informatics and Cognitive Computing, New York, NY, USA, 16–18 July 2013; pp. 137–142.

14. Warren, T. Microsoft Is Merging Its Outlook and Sunrise Apps. Available online: https://www.theverge.com/2015/10/28/9627014/microsoft-ios-android-apps-sunrise-outlook-combined (accessed on 23 August 2022).
15. David, K. What I Learned About Productivity While Reinventing Google Calendar. Available online: https://observer.com/2017/01/what-i-learned-about-productivity-while-reinventing-google-calendar/ (accessed on 23 August 2022).
16. Darrell, E. Google Acquires Timeful To Bring Smart Scheduling To Google Apps. Available online: https://social.techcrunch.com/2015/05/04/google-acquires-timeful-to-bring-smart-scheduling-to-google-apps/ (accessed on 23 August 2022).
17. Zhang, D. On Temporal Properties of Knowledge Base Inconsistency. In *Transactions on Computational Science V*; Gavrilova, M.L., Tan, C.J.K., Wang, Y., Chan, K.C.C., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2009; Volume 5540, pp. 20–37. ISBN 978-3-642-02096-4.
18. Roorda, M.J.; Doherty, S.T.; Miller, E.J. Operationalising Household Activity Scheduling Models: Addressing Assumptions and the Use of New Sources of Behavioural Data. In *Integrated Land-Use and Transportation Models*; Emerald Group Publishing Limited: Bingley, UK, 2005.
19. Auld, J.; Mohammadian, A.K. Activity Planning Processes in the Agent-Based Dynamic Activity Planning and Travel Scheduling (ADAPTS) Model. *Transp. Res. Part A Policy Pract.* **2012**, *46*, 1386–1403. [CrossRef]
20. Auld, J.; Mohammadian, A.K.; Doherty, S.T. Modeling Activity Conflict Resolution Strategies Using Scheduling Process Data. *Transp. Res. Part A Policy Pract.* **2009**, *43*, 386–400. [CrossRef]
21. Doherty, S.T.; Nemeth, E.; Roorda, M.; Miller, E.J. Computerized Household Activity-Scheduling Survey for Toronto, Canada, Area: Design and Assessment. *Transp. Res. Rec.* **2004**, *1894*, 140–149. [CrossRef]
22. Javanmardi, M.; Fasihozaman Langerudi, M.; Shabanpour, R.; Mohammadian, A. An Optimization Approach to Resolve Activity Scheduling Conflicts in ADAPTS Activity-Based Model. *Transportation* **2016**, *43*, 1023–1039. [CrossRef]
23. One in Ten Rule. Available online: https://en.wikipedia.org/w/index.php?title=One_in_ten_rule&oldid=1095296451 (accessed on 29 August 2022).
24. Konur, S. A Survey on Temporal Logics. *arXiv* **2010**, arXiv:1005.3199.
25. Allen, J.F.; Ferguson, G. Actions and Events in Interval Temporal Logic. *J. Log. Comput.* **1994**, *4*, 531–579. [CrossRef]
26. Allen, J.F. Towards a General Theory of Action and Time. *Artif. Intell.* **1984**, *23*, 123–154. [CrossRef]
27. Lutz, C.; Wolter, F.; Zakharyaschev, M. Temporal Description Logics: A Survey. In Proceedings of the 2008 15th International Symposium on Temporal Representation and Reasoning, Montreal, QC, Canada, 16–18 June 2008; pp. 3–14.
28. Schiewe, P.; Schöbel, A. Periodic Timetabling with Integrated Routing: Toward Applicable Approaches. *Transp. Sci.* **2020**, *54*, 1714–1731. [CrossRef]
29. Borndörfer, R.; Lindner, N.; Roth, S. A Concurrent Approach to the Periodic Event Scheduling Problem. *J. Rail Transp. Plan. Manag.* **2020**, *15*, 100175. [CrossRef]
30. Liebchen, C.; Möhring, R.H. The Modeling Power of the Periodic Event Scheduling Problem: Railway Timetables—And Beyond. In *Algorithmic Methods for Railway Optimization*; Springer: Berlin/Heidelberg, Germany, 2007; pp. 3–40.
31. Liebchen, C.; Möhring, R.H. A Case Study in Periodic Timetabling. *Electron. Notes Theor. Comput. Sci.* **2002**, *66*, 18–31. [CrossRef]
32. Srirama, S.N.; Flores, H.; Paniagua, C. Zompopo: Mobile Calendar Prediction Based on Human Activities Recognition Using the Accelerometer and Cloud Services. In Proceedings of the 2011 Fifth International Conference on Next Generation Mobile Applications, Services and Technologies, Cardiff, UK, 14–16 September 2011; pp. 63–69.
33. Gkekas, G.; Kyrikou, A.; Ioannidis, N. A Smart Calendar Application for Mobile Environments. In Proceedings of the 3rd International Conference on Mobile Multimedia Communications, Nafpaktos, Greece, 27–29 August 2007; pp. 1–5.
34. Paniagua, C.; Flores, H.; Srirama, S.N. Mobile Sensor Data Classification for Human Activity Recognition Using MapReduce on Cloud. *Procedia Comput. Sci.* **2012**, *10*, 585–592. [CrossRef]
35. About Us | Calendly. Available online: https://calendly.com/about (accessed on 23 August 2022).
36. Halang, C.; Schirmer, M. Towards a User-Centred Planning Algorithm for Automated Scheduling in Mobile Calendar Systems. *INFORMATIK 2012*. 2012. Available online: https://www.researchgate.net/publication/235970541_Towards_a_User-centred_Planning_Algorithm_for_Automated_Scheduling_in_Mobile_Calendar_Systems (accessed on 29 August 2022).
37. Zaidi, A.K.; Wagenhals, L.W. Planning Temporal Events Using Point–Interval Logic. *Math. Comput. Model.* **2006**, *43*, 1229–1253. [CrossRef]
38. Dvorák, F.; Barták, R.; Bit-Monnot, A.; Ingrand, F.; Ghallab, M. Planning and Acting with Temporal and Hierarchical Decomposition Models. In Proceedings of the 2014 IEEE 26th International Conference on Tools with Artificial Intelligence, Limassol, Cyprus, 10–12 November 2014; pp. 115–121.
39. Chen, Z.; Ma, N.; Liu, B. Lifelong Learning for Sentiment Classification. *arXiv* **2018**, arXiv:1801.02808.
40. Liu, B. Lifelong Machine Learning: A Paradigm for Continuous Learning. *Front. Comput. Sci.* **2017**, *11*, 359–361. [CrossRef]
41. Chen, Z.; Hruschka, E.R.; Liu, B. Lifelong Machine Learning and Computer Reading the Web. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA USA, 13–17 August 2016; ACM: New York, NY, USA, 2016; pp. 2117–2118.
42. Carlson, A.; Betteridge, J.; Kisiel, B.; Settles, B.; Hruschka, E.R.; Mitchell, T.M. Toward an Architecture for Never-Ending Language Learning. In Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, Atlanta, GA, USA, 11–15 July 2010.

43. Betteridge, J.; Carlson, A.; Hong, S.A.; Hruschka Jr, E.R.; Law, E.L.; Mitchell, T.M.; Wang, S.H. Toward Never Ending Language Learning. In Proceedings of the AAAI Spring Symposium: Learning by Reading and Learning To Read, Stanford, CA, USA, 23–25 March 2009; pp. 1–2.
44. Venema, Y. Temporal Logic. *Blackwell Guide Philos. Log.* **2017**, 203–223. [CrossRef]
45. Brunello, A.; Sciavicco, G.; Stan, I.E. Interval Temporal Logic Decision Tree Learning. In Proceedings of the European Conference on Logics in Artificial Intelligence, Rende, Italy, 7–11 May 2019; Springer: Berlin/Heidelberg, Germany, 2019; pp. 778–793.
46. Mu, Y.; Xiao, N.; Tang, R.; Luo, L.; Yin, X. An Efficient Similarity Measure for Collaborative Filtering. *Procedia Comput. Sci.* **2019**, *147*, 416–421. [CrossRef]
47. Anand, D.; Bharadwaj, K.K. Utilizing Various Sparsity Measures for Enhancing Accuracy of Collaborative Recommender Systems Based on Local and Global Similarities. *Expert Syst. Appl.* **2011**, *38*, 5101–5109. [CrossRef]
48. Pereira, F.S.F.; Gama, J.; de Amo, S.; Oliveira, G.M.B. On Analyzing User Preference Dynamics with Temporal Social Networks. *Mach. Learn.* **2018**, *107*, 1745–1773. [CrossRef]
49. Jagerman, R.; Markov, I.; de Rijke, M. When People Change Their Mind: Off-Policy Evaluation in Non-Stationary Recommendation Environments. In Proceedings of the Twelfth ACM International Conference on web Search and Data Mining, Melbourne, Australia, 11–15 February 2019; pp. 447–455.
50. Li, C.; De Rijke, M. Cascading Non-Stationary Bandits: Online Learning to Rank in the Non-Stationary Cascade Model. *arXiv* **2019**, arXiv:1905.12370.
51. Cheng, H.; Liu, Z.; Hou, L.; Yang, J. Sparsity-Induced Similarity Measure and Its Applications. *IEEE Trans. Circuits Syst. Video Technol.* **2016**, *26*, 613–626. [CrossRef]
52. Ma, H.; Gou, J.; Wang, X.; Ke, J.; Zeng, S. Sparse Coefficient-Based k-Nearest Neighbor Classification. *IEEE Access* **2017**, *5*, 16618–16634. [CrossRef]
53. Cherian, A. Nearest Neighbors Using Compact Sparse Codes. In Proceedings of the International Conference on Machine Learning, PMLR, Bejing, China, 22–24 June 2014; pp. 1053–1061.
54. Pisner, D.A.; Schnyer, D.M. Support Vector Machine. In *Machine learning*; Elsevier: Amsterdam, The Netherlands, 2020; pp. 101–121.
55. Suthaharan, S. Support Vector Machine. In *Machine Learning models and Algorithms for Big Data Classification*; Springer: Berlin/Heidelberg, Germany, 2016; pp. 207–235.