The Hague, Netherlands

August 30, 2016

Proceedings of the 2nd Workshop on

Artificial Intelligence and Internet of Things (2nd AI-IoT 2016)

In conjunction with
ECAI 2016

The Hague, Netherlands

August 30, 2016

Proceedings of the 2nd Workshop on

Artificial Intelligence and Internet of Things (2nd AI-IoT 2016)

In conjunction with
ECAI 2016

Organization

Chair

Constantine D. Spyropoulos	NCSR “Demokritos”, Greece

Program Committee

Constantine D. Spyropoulos	NCSR “Demokritos”, Greece
Alexander Artikis	NCSR “Demokritos” & University of Piraeus, Greece
Payam Barnaghi	University of Surrey, United Kingdom
Amedeo Cesta	National Research Council of Italy, Italy
Alfio Ferrara	University of Milano, Italy
Amelie Gyrard	Insight Centre for Data Analytics, Ireland
Alexander Jungmann	University of Paderborn, Germany
Konstantinos Kotis	University of Piraeus, Greece
Katerina Marinagi	Technological Educational Institute of Chalkis, Greece
Stavros Perantonis	NCSR “Demokritos”, Greece
Georgios Pierris	NCSR “Demokritos”, Greece
Evangelos Pournaras	ETH Zurich, Switzerland
Ioannis Refanidis	University of Macedonia, Greece
Yannis Soldatos	Athens Information Technology, Greece
Evaggelos Spyrou	NCSR “Demokritos”, Greece
Martin Strohbach	AGT International, Switzerland
Grigorios Tsoumakas	Aristotle University of Thessaloniki, Greece
Grigorios Tzortzis	NCSR “Demokritos”, Greece
Nick Vassiliadis	Aristotle University of Thessaloniki, Greece
Ioannis Vlahavas	Aristotle University of Thessaloniki, Greece
George Vouros	University of Piraeus, Greece

Organizing Committee

Constantine D. Spyropoulos	NCSR “Demokritos”, Greece
Georgios Pierris	NCSR “Demokritos”, Greece
Grigorios Tzortzis	NCSR “Demokritos”, Greece

Table of Contents

Preface

The continuous international efforts to enable everyday devices to participate in the emerging Internet of Things (IoT) ecosystem has led to an explosive increase in the number of smart devices that surround us. However, our capacity as humans to meaningfully process, manage, control, and interact with them is limited by human nature, our interests and our technical fluency. The coming new digital market envisions an ambient environment where the physical world, computer-based systems and humans converge and seamlessly interoperate, resulting in an improved social and economic marketplace.

Collectively, the public sector, industry, academia, end-users, SMEs, and large corporations constantly feed the, already, high expectations of IoT. Artificial Intelligence (AI) has the capacity to facilitate the anticipated socio-economic transformation caused by the proliferation of IoT through innovative algorithms and techniques.

The *Artificial Intelligence and Internet of Things* (AI-IoT) series of workshops aims at providing the ground for disseminating new and interesting ideas on how AI can make valuable contribution in solving problems that the IoT ecosystem faces. The virtualization of devices and smart systems, the discoverability and composition of services, the interoperability of services, the distribution of resources, the management and event recognition of big stream data, and the development of algorithms for edge and predictive analytics are only a few of the problems that look for intelligent human-centric solutions that could find application in smart cities, smart farming, transportation, health, smart grid, tourism, etc.

The second installment of the workshop – 2nd AI-IoT 2016 – was co-located with ECAI 2016 in The Hague, Netherlands and featured a keynote by Prof. Dirk Helbing from ETH Zurich, Switzerland, entitled "*Towards Smarter Societies*" and five accepted papers, resulting in an intriguing technical program. Papers accepted in the workshop gave special emphasis in AI-related topics such as:

- Machine learning
- AI planning
- Reasoning under uncertainty
- Personalization
- Classification
- Real-time event recognition
- Multi-agent systems

that have been explored in smart societies, tele-assistance, smart tourism, embedded sensor fusion, for activity recognition in surveillance and security systems, and for detecting treads and abnormal activities in maritime surveillance.

Specifically in this proceedings the contributions of the accepted papers are as follows. In the "*Third Generation Teleassistance: Intelligent Monitoring Makes* *the Difference*", Rafael-Palou et al. propose an intelligent monitoring solution for elderly people, integrated in an IoT-based tele-assistance system, demonstrating how it contributes in offering enhanced support to both end-users and caregivers. Machine learning methods based on SVM are used for detecting interesting events and issuing alarms in case of an emergency. Results from deploying the system in real-life situations are presented. Marzal et al. in "*Temporal Goal Reasoning for Predictive Performance of a Tourist Application*", discuss a goal reasoning framework that identifies if the context information acquired from several external resources dictates a change in the execution of a temporal plan. TempLM, a temporal planner that uses temporal landmarks for planning with temporal deadlines, detects situations of future failures and opportunities in the plan execution. The capability of the planner to adapt to external events is showcased in a smart tourism scenario. Babli et al. in their paper entitled "*An Intelligent System for Smart Tourism Simulation in a Dynamic Environment*" present an AI planning-based system for the smart tourism domain, where the goal is to construct a personalized tourist agenda of places a tourist could visit according to his preferences. The system not only creates the agenda, but also monitors its execution in real-time through simulation. Emphasis is given in dynamically reacting to changes in the environment by adapting, if necessary, the tourist agenda, through reformulation of the planning problem, to reflect the new state of the environment in real-time. In "*Extending Naive Bayes with Precision-tunable Feature Variables for Resource-efficient Sensor Fusion*", Galindez Olascoaga et al. focus on the tradeoff between resource efficiency and inference accuracy, by tuning feature quality in sensing devices. An extension to the naive Bayes classifier is implemented and evaluated in sensor fusion tasks. The algorithm is capable of dynamically tuning feature precision as a function of the incoming data quality, the difficulty of the task and the resource availability. In the last paper, "*A Distributed Event Calculus for Event Recognition*", Mavrommatis et al. present a distributed approach for stream reasoning, called dRTEC, based on a dialect of event calculus. dRTEC employs the Apache Spark framework to perform scalable event recognition and detect significant patterns.

The organizers would like to thank the authors for submitting their work to the workshop, the members of the program committee for their valuable contribution in reviewing the papers and, of course, the numerous participants of the workshop.

Constantine D. Spyropoulos
Georgios Pierris
Grigorios Tzortzis

Organizers of 2nd AI-IoT 2016

`Workshop site: http://2nd-ai-iot2016.iit.demokritos.gr/`

Keynote

Speaker: Prof. Dirk Helbing, ETH Zurich, Switzerland

Title: Towards Smarter Societies

Abstract – As the recent triumph of alphaGo has shown, the exponential increase in computer power allows us now to solve challenging problems that seemed to be out of reach for a long time. So, would we eventually be able to build superintelligent computers that would be able to solve humanities' 21st century problems and run our society in an optimal way? Surprisingly, the answer is "no", because data volumes increase faster than processing power, and systemic complexity even faster. This has a number of implications: local knowledge, context as well as distributed computing and control will become more important. Science will be relevant again to decide what data to process and how, and how to collect the right kind of data in the first place. This talk will elaborate on a number of pitfalls in the areas of Data Science and AI, and it will make proposals how to use these technologies and the Internet of Things more successfully, with context-aware approaches.

Third Generation Teleassistance: Intelligent Monitoring Makes the Difference

Xavier Rafael-Palou[1] and **Carme Zambrana**[1] and **Stefan Dauwalder**[1] and **Enrique de la Vega**[2] and **Eloisa Vargiu**[1] and **Felip Miralles**[1]

Abstract. Elderly people aim to preserve their independence and autonomy at their own home as long as possible. However, as they get old the risks of disease and injuries increase making critical to assist and provide them the right care whenever needed. Unfortunately, neither relatives, private institutions nor public care services are viable long-term solutions due to the large amount of required time and cost. Thus, smart teleassistance solutions must be investigated. In particular, IoT paradigm helps in designing third generation teleassistance systems by relying on sensors to gather the more data as possible. Moreover, we claim that providing IoT solutions of intelligent monitoring improves the overall efficacy. In this paper, we presents an intelligent monitoring solution, fully integrated in a IoT-based teleassistance system, showing how it helps in giving better support to both end-users and carers. Thanks to intelligent monitoring, carers can instantly access to the relevant information regarding the status of the end-user, also receiving alarms in case of any anomaly or emergency situations have been detected.

1 Introduction

In the last decade, the Internet of Things (IoT) paradigm rapidly grew up gaining ground in the scenario of modern wireless telecommunications [6]. Its basic idea is the pervasive presence of a variety of things or objects (e.g, tags, sensors, actuators, smartphones, everyday objects) that are able to interact with each other and cooperate with their neighbors to reach common goals. IoT solutions have been investigated and proposed in several fields [7], such as automotive [17], logistics [19], agriculture [31], entertainment [18], and independent living [12].

Several research issues are still open: standardization, networking, security, and privacy [27]. We claim that research might also focus on intelligent techniques to improve IoT solutions thus making the difference with respect to classical systems. In other words, artificial intelligence algorithms and methods may be integrated in IoT systems: to allow better coordination and communication among sensors, through adopting multi-agent systems [1]; to adapt the sensor network according to the context, by relying, for instance, on deep learning techniques [16]; as well as to provide recommendations to the final users, by using data fusion and semantic interpretation [4].

Considering the dependency care sector as a case study, in this paper we show how intelligent monitoring techniques, integrated in a IoT-based teleassistance system (namely, eKauri[3]), help in providing better assistance and support to people that need assistance. eKauri is a teleassistance system composed of a set of wireless sensors connected to a gateway (based on Raspberry-pi) that collects and securely redirects them to the cloud. It is worth noting that eKuari is composed by the following kinds of sensors: one presence-illumination-temperature sensors (i.e., TSP01 Z-Wave PIR) for each room, and one presence-door-illumination-temperature sensor (i.e., TSM02 Z-Wave PIR) for each entry door. Intelligent monitoring in eKauri allows to detect the following events: leaving home; going back to home; receiving a visit; remaining alone after a visit; going to the bathroom; going to sleep; and awaking from sleep. In this paper, we focus on the contribution of the intelligent monitoring in eKauri, the interested reader may refer to [23] for a deep description of the system.

The rest of the paper is organized as follows. In Section 2, we briefly recall IoT solutions to teleassistance. Section 3 illustrates how intelligent monitoring improves teleassistance in the eKauri system. In Section 4, the main installations of eKauri are presented together with users' experience. Section 5 ends the paper summarizing the main conclusions.

2 Related Work

Teleassistance remotely, automatically and passively monitors changes in people's condition or lifestyle, with the final goal of managing the risks of independent living [9] [2]. In other words, thanks to teleassistance, end-users are connected with therapists and caregivers as well as relatives and family, allowing people with special needs to be independent.

There are several of efforts to utilize IoT-based systems for monitoring elderly people, most of which target only certain aspects of elderly requirements from a limited viewpoint. Gokalp and Clarke reviewed monitoring activities of daily living of elderly people comparing characteristics, outcomes, and limitations of 25 studies [15]. They found that adopted sensors are mainly environmental, except for accelerometers and some physiological sensors. Ambient sensors could not differentiate the subject from visitors, as opposed to wearable sensors [8] [5]. On the other hand, the latter could only distinguish simple activities, such as walking, running, resting, falling, or inactivity [3]. Moreover, wearable sensors are not suitable for cognitively impaired elderly people due to the fact that they are likely to be forgotten or thrown away [11] [14]. Their main conclusion regarding sensors is that daily living activity monitoring requires use of a combination of ambient sensors, such as motion and door sensors.

[1] eHealth Unit, EURECAT, Barcelona, email: {xavier.rafael, carme.zambrana, stefan.dauwalder, eloisa.vargiu, felip.miralles}@eurecat.org

[2] Technology Transfer Unit, EURECAT, Barcelona, email: enrique.delavega@eurecat.org

[3] www.ekauri.com

3 Intelligent Monitoring Makes the Difference

Filtering and analyzing data coming from teleassistance systems is becoming more and more relevant. In fact, a lot of data are continuously gathered and sent through the sensors. The role of therapists, caregivers, social workers, as well as relatives (hereinafter, carers) is essential for remotely assisting monitored users. On the one hand, the monitored user (e.g., elderly or disabled people) needs to be kept informed about emergencies as soon as they happen and s/he has to be in contact with therapists and caregivers to change habits and/or to perform some therapy. On the other hand, monitoring systems are very important from the perspective of carers. In fact, those systems allow them to become aware of user context by acquiring heterogeneous data coming from sensors and other sources. Thus, intelligent solutions able to understand all those data and process them to keep carers aware about their assisted persons are needed, providing also users empowerment.

In the following, we show how intelligent monitoring helps in: improving sensors reliability allowing better activity recognition; providing useful information to carers; and inferring quality of life of users.

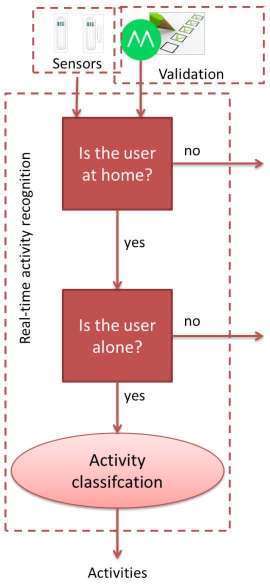


Figure 1. The hierarchical approach to presence detection.

3.1 Improving Sensors Reliability

Performance of IoT systems depends, among other characteristics, on the reliability of the adopted sensors. In the case of teleassistance, binary sensors are quite used in the literature and also in commercial solutions to identify user's activities. Binary sensors do not have the ability to directly identify people and can only present two possible values as outputs ("0" and "1"). Typical examples of binary sensors deployed within smart environments include pressure mats, door sensors, and movement detectors. A number of studies reporting the use of binary and related sensors have been undertaken for the purposes of activity recognition [26]. Nevertheless, sensor data can be considered to be highly dynamic and prone to noise and errors [25]. In the following, we present two solutions that rely on machine learning to improve reliability of sensors in presence detection and sleeping recognition, respectively.

3.1.1 Presence Detection

Detecting user's entering/leaving home can be done by relying on door sensors. Fusing data from door- and motions-sensors could help also in recognizing if the user received visits. Unfortunately, as said, sensors are not 100% reliable: sometimes they loose events or detect them several times. When sensors remain with a low battery charge they get worse. Moreover, also the Raspberry pi may loose some data or the connection with Internet and/or with the sensors. Also the Internet connection may stop working or loose data. Finally, without using a camera or wearable sensors we are not able to directly recognize if the user is alone or if s/he has some visits.

In order to solve this kind of limitations with the final goal of improving the overall performance of our IoT-based system that uses only motion and door sensors, we defined and adopt a two-levels hierarchical classifier (see Figure 1) [24]: the upper level is aimed at recognizing if the user is at home or not, whereas the lower is aimed at recognizing if the user is really alone or if s/he received some visits.

The goal of the classifier at the upper level is to improve performance of the door sensor. In fact, it may happen that the sensor registers a status change (from closed to open) even if the door has not been opened. This implies that the system may register that the user is away and, in the meanwhile, activities are detected at user's home. On the contrary, the system may register that the user is at home and, in the meanwhile, activities are not detected at user's home. To solve, or at least reduce, this problem, we built a supervised classifier able to recognize if the door sensor is working well or erroneous events have been detected. First, we revise the data gathered by the sensor-based system searching for anomalies, i.e.: (1) the user is away and at home some events are detected and (2) the user is at home and no events are detected. Then, we validated those data by relying on Moves, an app installed and running on the user smartphone[4]. In fact, Moves, among other functionality, is able to localize the user. Hence, using Moves as an "oracle" we build a dataset in which each entry is labeled depending on the fact that the door sensor was right (label "1") or wrong (label "0").

The goal of the classifier at the lower level is to identify whether the user is alone or not. The input data of this classifier are those that has been filtered by the upper level, being recognized as positives. To build this classifier, we rely on the novelty detection approach [20] used when data has few positive cases (i.e., anomalies) compared with the negatives (i.e., regular cases); in case of skewed data.

The hierarchical approach was part of the EU project BackHome[5]. To train and test it, we consider a window of 4 months for training and evaluation (training dataset) and a window of 1 month for the test (testing dataset). Experiments have been performed at each level of the hierarchy. First, we performed experiments to identify the best supervised classifier to be used at the upper level of the hierarchy. The best performance has been obtained by relying on the SVM (with $\gamma = 1.0$ and $C = 0.452$). Subsequently, we applied the novelty detection algorithm on the data filtered by the classifier at the upper level, to validate the classifier at the lower one. Finally, we measure the performance of the overall approach. We compared the overall results with those obtained by using the rule-based approach in both levels of the hierarchy. Results are shown in Table 1 and point out

[4] https://www.moves-app.com/
[5] www.backhome-fp7.eu

that the proposed approach outperforms the rule-based one with a significant improvement.

Table 1. Results of the overall hierarchical approach with respect to the rule-based one.

Metric	Rule-based	Hierarchical	Improv.
Accuracy	0.80	0.95	15%
Precision	0.68	0.94	26%
Recall	0.71	0.91	20%
F_1	0.69	0.92	23%

3.1.2 Sleep Recognition

We defined the sleeping activity as the period which begins when the user goes to sleep and ends when the user wakes up in the morning. Sleep recognition is aimed at reporting the following information: (i) the time when the user went to sleep and woke up; hereinafter we will refer to them as *go to sleep time* and *wake up time*, respectively; (ii) the number of sleeping activity hours; and (iii) the number of rest hours, which are sleeping activity hours minus the time that the user spent going to the toilet or performing other activities during the night.

Let us note that the simplest way to recognize sleeping activities is relying on a rule-based approach. In particular, the following rules may be adopted: the user is in the bedroom; the activity is performed at night (e.g., the period between 8 pm to 8 am); the user is inactive; and the inactivity duration is more than half an hour. Unfortunately, when moving to the real-world, some issues arise: user movements in the bed might be wrongly classified as awake; rules assumed all users wake up before 8 A.M., which is a strong assumption; and the approach cannot distinguish if the user is, for instance, in the bedroom watching TV or reading a book, thus classifying all those actions as sleeping.

In order to overcome those limitations, an SVM (Radial Basis Function kernel, with $C = 1.0$, $\gamma = 1.0$) has been adopted to classify the periods between two bedroom motions in two classes, awake and sleep [32]. Let us note that awake corresponds to the period in which the user goes to another room; performs activities in the bedroom; or stays in the bedroom with the light switched on. Otherwise, the activity is sleep.

Experiments, performed from May 2015 to January 2016 in 13 homes in Barcelona, show that the adopted machine learning solution is able to recognize when the user is performing her/his sleeping activity. In particular, the proposed approach reaches an F1 of 96%. Moreover, the adopted classifier is able to easily detect the *go to sleep time*, the *wake up time*, the number of sleeping activity hours and the number of rest hours. Figure 2 shows the comparisons between the ground truth (obtained by questionnaires answered by the users) and the results obtained with the machine learning approach (based on an SVM classifier). The plot has as temporal axis (axis x) and each coordinate in axis y represents nights in the dataset. The figure shows, in red, the sleep activity hours according to the ground truth and, in blue, the sleep activity hours calculated by the system. As both sleep activity hours of the same night are plotted in the same y coordinate, if the ground truth and the results coincide the color turns purple. If the *go to sleep time* and/or *wake up time* do not coincide, there is a text next to the corresponding side with the difference between the time coming from the ground truth and that coming from the results. In the middle of each bar there is the total time which results differ from the baseline.

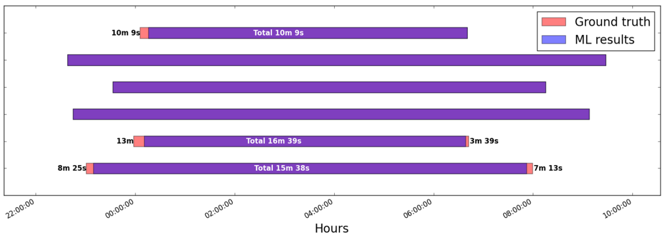
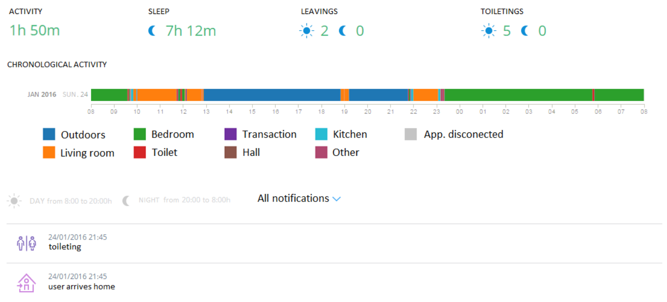


Figure 2. Comparison between the ground truth and the machine-learning (SVM) one.

3.2 Providing Feedback to Carers

The role of carers is essential for remotely assisting people that need assistance. Thus, intelligent monitoring able to understand gathered data and process them to keep carers aware about their assisted persons are needed [13].


ACTIVITY
1h 50m
SLEEP
7h 12m
LEAVINGS
2 0
TOILETINGS
5 0
CHRONOLOGICAL ACTIVITY
JAN 2016 SUN. 24

🟦 Outdoors
🟧 Living room
🟩 Bedroom
🟥 Toilet
🟪 Transaction
🟫 Hall
Kitchen
Other
⬜ App. disconected
DAY from 8:00 to 20:00h
NIGHT from 20:00 to 8:00h
All notifications
24/01/2016 21:45
toileting
24/01/2016 21:45
user arrives home

Figure 3. The main information given to carers through the healthcare center.

Thanks to the user-centered approach from the above-mentioned projects, we designed friendly and useful interfaces for accessing and visualizing relevant data and information. In particular, carers identified as the most relevant the following information (see Figure 3, first line on the top): time spent making activities, time spent sleeping, number of times the user leaves the home (during both day and night), and number of times the user goes to toilet (during both day and night). Moreover, they considered relevant to visually show the rooms where the user stayed time after time during a day (see Figure 3, central part) or during a period (e.g., the last month, as shown in Figure 4). They also want to be informed about all the notifications, chronologically ordered (see Figure 3, on the bottom). Finally, they want to access to some statistics to be aware about the evolution of user's habits in order to act accordingly.

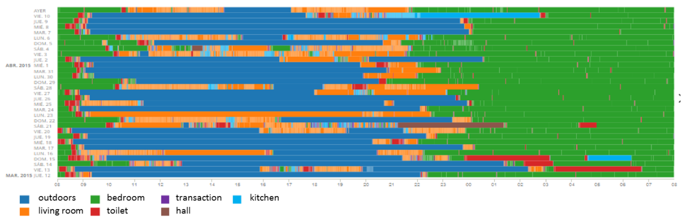


Figure 4. 1 month reporting.

To highlight the relevance of providing suitable information to carers, let us mention here two cases that happened during Barcelona installations in collaboration with Centre de Vida Independent[6]. *Case-1.* A woman with Alzheimer and heart problems needs continuously assistance and, thus, a caregiver visits her daily. One day, eKauri detected that no visits were received, an alarm was generated and the caregiver called. The caregiver confirmed that she did not go to visit the user that day. *Case-2.* During the afternoon, a user is accustomed to go out for a walk. One day, she stayed in the bedroom. eKauri detected the change in her habit and a caregiver called her. Actually, she had a problem with a knee and she could not walk. A physiotherapist was asked to go to visit her.

3.3 Assessing Quality of Life of Users

In the dependency care sector, analyzing data gathered by sensors may help in improving teleassistance systems in becoming aware of user context. In so doing, they would be able to automatically infer user's behavior as well as detect anomalies. In this direction, we studied a solution aimed at automatically assessing quality of life of people [29]. The goal is twofold: to provide support to people in need of assistance and to inform therapists, carers and families about the improvement/worsening of quality of life of monitored people.

First, we defined a Visual Analogic Scale (VAS) QoL questionnaire composed of the following items: *MOOD*, *HEALTH*, *MOBILITY*, *SATISFACTION WITH CARE*, *USUAL ACTIVITIES* (which includes *SLEEPING*), and *PAIN/DISCOMFORT*. Those items have been categorized in two families: monitorable and inferable. Monitorable items can be directly gathered from sensors without relying on direct input from the user. Inferable items can be assessed by analyzing data retrieved by the system when considering activities performed by the user not directly linked with the sensors.

We performed experiments on two monitorable items (i.e., *MOBILITY* and *SLEEPING*) and one inferable (i.e., *MOOD*). In particular, we are able to detect and acknowledge the location of the user over time as well as the covered distance in kilometers and the places where s/he stayed. At the same time, we can detect when the user is sleeping as well as how many times s/he is waking up during the night. Merging and fusing the information related to *MOBILITY* and *SLEEPING*, we may also infer the overall *MOOD*.

The corresponding QoL assessment system is composed of a set of sub-modules, each one devoted to assess a specific QoL item; namely: *MOBILITY*-assessment module; *SLEEPING*-assessment module; and *MOOD*-assessment module. Each sub-module is composed of two parts: Feature Extractor and Classifier. The Feature Extractor receives as input the list of notifications $\{n\}$ and the list of activities $\{a\}$ and extracts the relevant features $\{f\}$ to be given as input to the Classifier. The Classifier, then, uses those features to identify the right class Cl. This information will be then part of the overall summary Σ.

Each Feature Extractor works with its proper list of features:

- *MOBILITY*: number of times the user left home, total time performing outdoor activities, total time performing activities (both indoors and outdoors), total time of inactivity, covered distance, number of performed steps, number of visited places, number of burned calories.
- *SLEEPING*: total sleeping time, hour the user went to sleep, hour the user woke up, number of times the user went to the toilet during the night, time spent at the toilet during the night, number of time the user went to the bedroom during the night, time spent at the bedroom during the night, number of sleeping hours the day before, number of sleeping hours in the five days before.
- *MOOD*: number of received visits, total time performing outdoor activities, total time performing activities (both indoors and outdoors), total time of inactivity, covered distance, number of performed steps, number of burned calories, hour the user went to sleep, hour the user woke up, number of times the user went to the toilet during the night, time spent at the toilet during the night, number of time the user went to the bedroom during the night, time spent at the bedroom during the night, number of sleeping hours the day before, number of sleeping hours in the five days before. The Classifier is a supervised multi-class classifier built by using data previously labeled by the user and works on five classes, Very Bad, Bad, Normal, Good, and Very Good.

[6] http://www.cvi-bcn.org/en/

Under the umbrella of BackHome, we tested our approach with 3 users with severe disabilities (both cognitive and motor) living at their own real homes [30]. Although the system was evaluated by using as ground truth answers given to QoL questionnaires that is an approach completely subjective that depends on the particularity of each monitored user, after only 3 weeks of testing, the approach seemed convincing. Results presented in this paper show that *MOBILITY*, *SLEEPING*, and *MOOD* can be inferred with a high accuracy (0.76, 0.72, and 0.81, respectively) by relying on an automatic QoL assessment system. Let us note that *SLEEPING* was the method with the lowest performance. This is due to the fact that, currently, the system uses only motion sensors. Higher performances could be expected when combining motion sensors with other ones, such as mat-pressure or light sensors. *MOBILITY* achieved higher performance results than *SLEEPING* especially when outdoor and indoor features are merged together. In fact, using only outdoor features was not as reliable as combining with indoor. This can be due to the reliability of the GPS system embedded in the smartphone that made some errors in identifying when the user was really away. Let us also note that this is an important result because disable people in general spend a lot of time at their home. Finally, *MOOD* reported the highest performances. Although at a first instance this could be surprising, this fact might be explained considering the intrinsic correlation between *SLEEPING* and *MOBILITY*, as highlighted by the questionnaire compiled daily by the users. It is worth noting that higher performances could be expected considering also social networking activities performed by the user.

4 Users' Experience

The proposed solution has been developed according to a user-centered design approach in order to collect requirements and feedback from all the actors (i.e., end-users and their relatives, professionals, caregivers, and social workers). For evaluation purposes, the system has been installed in two healthy-user homes in Barcelona (control users).

The system has been used in the EU project BackHome to monitor disabled people. BackHome was an European R&D project that focuses on restoring independence to people that are affected by motor impairment due to acquired brain injury or disease, with the overall aim of preventing exclusion [21] [22]. In BackHome, information gathered by the sensor-based system is used to provide context-awareness by relying on ambient intelligence [10]. Intelligent monitoring was used in BackHome to study habits and to automatically assess QoL of people. The BackHome system ran in 3 end-user's home in Belfast.

In collaboration with Centre de Vida Independent[7], from May 2015 to January 2016, eKauri was installed in Barcelona in 13 elderly people' homes (12 women) over 65 years old [28]. To test eKauri, monitored users were asked to daily answer to a questionnaire composed of 20 questions (12 optional). Moreover, they daily received a phone-call by a caregiver who manually verifies the data. All detected events were shown in the Web applications and revised by therapists and caregivers. Feedback from them has been used to improve the interface and add functionality.

Although, at least at the beginning, users were a little bit reticent, during the monitored period they felt comfortable with the services provided by eKauri. In particular, they really appreciated the fact that it is not-intrusive and that it allows them to follow their normal lives. In the case of CVI, people also be grateful for being called by phone. In other words, it is important to provide a system that may become part of the home without losing social interactions. Thus, a teleassistance system does not substitute the role of caregivers. On the other side, carers recognized eKauri as a support to detect users' habits helping in diagnosing user's conditions and her/his decline, if any.

Currently, eKauri is installed in 40 elderly people's homes in the Basque Country in collaboration with Fundación Salud y Comunidad[8].

5 Conclusions

Considering the dependency care sector as a case study, in this paper we highlighted how intelligent monitoring techniques, integrated in eKauri, an IoT-based teleassistance system, allow to better provide assistance and support to people that need assistance. In particular, we focused on the power of intelligent monitoring in improving sensor reliability, activity recognition, feedback provided to carers, as well as quality of life of final users. As a matter of fact, results about independent home evaluation of eKauri show a good acceptance of the system by both home users and caregivers. Being promising, the potential socio-economic impact of the exploitation of the system, as well as barriers and facilitators for future deployment, have to be analyzed before going to the market.

Summarizing, our main conclusion is that time is ripe to adopt IoT in the real world and that intelligent monitoring makes the difference in providing feedback to the users. However, to become pervasive, in particular in the dependency care sector, solutions must be taken into account the role of the final users in each phase of the development. Moreover, even if users at home and caregivers give a positive feedback, one step ahead might be performed to allow that stakeholders will take value from third generation teleassistance systems. It means that, as technological providers, we must put into effect concrete solutions that give a twist in adapting innovative strategies.

Acknowledgments

This research was partially funded by the European Community, under the BackHome project (grant agreement n. 288566 - Seventh Framework Programme FP7/2007-2013), and the CONNECARE project (grant agreement n. 689802 - H2020-EU.3.1). This paper reflects only the authors' views and funding agencies are not liable for any use that may be made of the information contained herein.

[7] http://www.cvi-bcn.org/en/
[8] https://www.fsyc.org/

REFERENCES

[1] Charilaos Akasiadis, Evaggelos Spyrou, Georgios Pierris, Dimitris Sgouropoulos, Giorgos Siantikos, Alexandros Mavrommatis, Costas Vrakopoulos, and Theodoros Giannakopoulos, 'Exploiting future internet technologies: the smart room case', in *Proceedings of the 8th ACM International Conference on PErvasive Technologies Related to Assistive Environments*, p. 85. ACM, (2015).

[2] Stelios Andreadis, Thanos G Stavropoulos, Georgios Meditskos, and Ioannis Kompatsiaris, 'Dem@ home: Ambient intelligence for clinical support of people living with dementia', in *Proc. 1st Workshop on Semantic Web Technologies in Pervasive and Mobile Environments (SEMPER2016)*, (2016).

[3] Urs Anliker, Jamie A Ward, Paul Lukowicz, Gerhard Tröster, Francois Dolveck, Michel Baer, Fatou Keita, Eran B Schenker, Fabrizio Catarsi, Luca Coluccini, et al., 'Amon: a wearable multiparameter medical monitoring and alert system', *Information Technology in Biomedicine, IEEE Transactions on*, **8**(4), 415–427, (2004).

[4] Alexander Artikis, Panagiotis D Bamidis, Antonis Billis, Charalampos Bratsas, Christos Frantzidis, Vangelis Karkaletsis, Manousos Klados, Evdokimos Konstantinidis, Stasinos Konstantopoulos, Dimitris Kosmopoulos, et al., 'Supporting tele-health and ai-based clinical decision making with sensor data fusion and semantic interpretation: The usefil case study', in *International Workshop on Artificial Intelligence and NetMedicine*, p. 21, (2012).

[5] Louis Atallah, Benny Lo, Raza Ali, Rachel King, and Guang-Zhong Yang, 'Real-time activity classification using ambient and wearable sensors', *Information Technology in Biomedicine, IEEE Transactions on*, **13**(6), 1031–1039, (2009).

[6] Luigi Atzori, Antonio Iera, and Giacomo Morabito, 'The internet of things: A survey', *Computer networks*, **54**(15), 2787–2805, (2010).

[7] Debasis Bandyopadhyay and Jaydip Sen, 'Internet of things: Applications and challenges in technology and standardization', *Wireless Personal Communications*, **58**(1), 49–69, (2011).

[8] NP Bidargaddi, A Sarela, et al., 'Activity and heart rate-based measures for outpatient cardiac rehabilitation', *Methods of information in medicine*, **47**(3), 208–216, (2008).

[9] Peter Bower, Martin Cartwright, Shashivadan P Hirani, James Barlow, Jane Hendy, Martin Knapp, Catherine Henderson, Anne Rogers, Caroline Sanders, Martin Bardsley, et al., 'A comprehensive evaluation of the impact of telemonitoring in patients with long-term conditions and social care needs: protocol for the whole systems demonstrator cluster randomised trial', *BMC health services research*, **11**(1), 184, (2011).

[10] Eloi Casals, José Alejandro Cordero, Stefan Dauwalder, Juan Manuel Fernández, Marc Solà, Eloisa Vargiu, and Felip Miralles, 'Ambient intelligence by atml: Rules in backhome', in *Emerging ideas on Information Filtering and Retrieval. DART 2013: Revised and Invited Papers; C. Lai, A. Giuliani and G. Semeraro (eds.)*, (2014).

[11] Marie Chan, Eric Campo, and Daniel Estève, 'Assessment of activity of elderly people using a home monitoring system', *International Journal of Rehabilitation Research*, **28**(1), 69–76, (2005).

[12] Angelika Dohr, R Modre-Opsrian, Mario Drobics, Dieter Hayn, and Günter Schreier, 'The internet of things for ambient assisted living', in *2010 Seventh International Conference on Information Technology*, pp. 804–809. Ieee, (2010).

[13] Juan Manuel Fernández, Marc Solà, Alexander Steblin, Eloisa Vargiu, and Felip Miralles, 'The relevance of providing useful information to therapists and caregivers in tele*', in *DART 2014: Revised and Invited Papers; C. Lai, A. Giuliani and G. Semeraro (eds.)*, (in press).

[14] Céline Franco, Jacques Demongeot, Christophe Villemazet, and Nicolas Vuillerme, 'Behavioral telemonitoring of the elderly at home: Detection of nycthemeral rhythms drifts from location data', in *Advanced Information Networking and Applications Workshops (WAINA), 2010 IEEE 24th International Conference on*, pp. 759–766. IEEE, (2010).

[15] Hulya Gokalp and Malcolm Clarke, 'Monitoring activities of daily living of the elderly and the potential for its use in telecare and telehealth: a review', *TELEMEDICINE and e-HEALTH*, **19**(12), 910–923, (2013).

[16] Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic, and Marimuthu Palaniswami, 'Internet of things (iot): A vision, architectural elements, and future directions', *Future Generation Computer Systems*, **29**(7), 1645–1660, (2013).

[17] Wu He, Gongjun Yan, and Li Da Xu, 'Developing vehicular data cloud services in the iot environment', *Industrial Informatics, IEEE Transactions on*, **10**(2), 1587–1595, (2014).

[18] Chih-Lin Hu, Hung-Tsung Huang, Cheng-Lung Lin, Nguyen Huu Minh Anh, Yi-Yu Su, and Pin-Chuan Liu, 'Design and implementation of media content sharing services in home-based iot networks', in *Parallel and Distributed Systems (ICPADS), 2013 International Conference on*, pp. 605–610. IEEE, (2013).

[19] Stephan Karpischek, Florian Michahelles, Florian Resatsch, and Elgar Fleisch, 'Mobile sales assistant-an nfc-based product information system for retailers', in *Near Field Communication, 2009. NFC'09. First International Workshop on*, pp. 20–23. IEEE, (2009).

[20] Markos Markou and Sameer Singh, 'Novelty detection: a review?part 1: statistical approaches', *Signal processing*, **83**(12), 2481–2497, (2003).

[21] Felip Miralles, Eloisa Vargiu, Stefan Dauwalder, Marc Solà, Gernot Müller-Putz, Selina C. Wriessnegger, Andreas Pinegger, Andrea Kübler, Sebastian Halder, Ivo Käthner, Suzanne Martin, Jean Daly, Elaine Armstrong, Christoph Guger, Christoph Hintermüller, and Hannah Lowish, 'Brain computer interface on track to home', *The Scientific World Journal - Advances in Brain-Computer Interface*, (submitted).

[22] Felip Miralles, Eloisa Vargiu, Xavier Rafael-Palou, Marc Solà, Stefan Dauwalder, Christoph Guger, Christoph Hintermüller, Arnau Espinosa, Hannah Lowish, Suzanne Martin, et al., 'Brain computer interfaces on track to home: Results of the evaluation at disabled end-users's homes and lessons learnt', *Frontiers in ICT*, **2**, 25, (2015).

[23] Xavier Rafael-Palou, Eloisa Vargiu, Stefan Dauwalder, and Felip Miralles, 'Monitoring and supporting people that need assistance: the backhome experience', in *DART 2014: Revised and Invited Papers; C. Lai, A. Giuliani and G. Semeraro (eds.)*, (in press).

[24] Xavier Rafael-Palou, Eloisa Vargiu, Guillem Serra, and Felip Miralles, 'Improving activity monitoring through a hierarchical approach', in *The International Conference on Information and Communication Technologies for Ageing Well and e-Health (ICT 4 Ageing Well)*, (2015).

[25] Anand Ranganathan, Jalal Al-Muhtadi, and Roy H Campbell, 'Reasoning about uncertain contexts in pervasive computing environments', *IEEE Pervasive Computing*, **3**(2), 62–70, (2004).

[26] Emmanuel Munguia Tapia, Stephen S Intille, and Kent Larson, *Activity recognition in the home using simple and ubiquitous sensors*, Springer, 2004.

[27] Chun-Wei Tsai, Chin-Feng Lai, and Athanasios V Vasilakos, 'Future internet of things: open issues and challenges', *Wireless Networks*, **20**(8), 2201–2217, (2014).

[28] Eloisa Vargiu, Stefan Dauwalder, Xavier Rafael-Palou, Felip Miralles amd Alejandra Millet Pi-Figueres, Lluïsa Pla i Masip, and Cèlia Riera Brutau, 'Monitoring elderly people at home: Results and lessons learned', in *16th International Conference for Integrated Care 2016, Barcelona, May 23-25*, (2016).

[29] Eloisa Vargiu, Juan Manuel Fernández, and Felip Miralles, 'Context-aware based quality of life telemonitoring', in *Distributed Systems and Applications of Information Filtering and Retrieval. DART 2012: Revised and Invited Papers. C. Lai, A. Giuliani and G. Semeraro (eds.)*, (2014).

[30] Eloisa Vargiu, Xavier Rafael-Palou, and Felip Miralles, 'Experimenting quality of life telemonitoring in a real scenario', *Artificial Intelligence Research*, **4**(2), p136, (2015).

[31] Duan Yan-e, 'Design of intelligent agriculture management information system based on iot', in *Intelligent Computation Technology and Automation (ICICTA), 2011 International Conference on*, volume 1, pp. 1045–1049. IEEE, (2011).

[32] Carme Zambrana, Xavier Rafael-Palou, and Eloisa Vargiu, 'Sleeping recognition to assist elderly people at home', *Artificial Intelligence Research*, **5**(2), 64–70, (2016).

Temporal goal reasoning for predictive performance of a tourist application

Eliseo Marzal, Jesus Ibañez, Laura Sebastia, Eva Onaindia [1]

Abstract. Many real-work smart environments make use of IoT to be provided with context-aware services. Additionally, these environments require making decisions based on predictions about future actions. This involves the use of goal-directed behaviour which may need reasoning about new goals. This paper is devoted to analyze when a new goal can be formulated. Once a plan has been computed for a given problem, exogenous events can change the environment so that a failure in the plan is caused or an opportunity arises. This paper present a goal reasoning framework where context information acquired from several external sources determines a change that may affect the execution of the current plan. This change may cause a failure or an opportunity. We show how the planning system, namely TempLM, is able to predict both failures and opportunities thanks to the analysis of the Temporal Landmarks Graph and the Temporal Propositions Graph built for the given problem.

1 INTRODUCTION

One key feature in the application of innovative technologies like IoT in smart environments is the capability of providing context-aware services. Besides real-world information, this requires anticipatory behaviour through reasoning; that is, making decisions based on predictions and expectations about future actions [1]. Particularly, many real-world applications involve unanticipated changes that may bring an alteration of the current process or a future impact on the application or even an opportunity to include some new functionality.

The purpose of a planning application is to achieve a goal through the execution of a plan or course of actions. The arrival of an unexpected environmental event introduces a new piece of information that was not taken into account during the construction of the plan and that may affect the active plan in several ways. Typically, the first reaction is to check if the plan is no longer executable and, if so, apply a repair mechanism or replanning to fix the failure that prevents the active plan from being normally executed. A second consequence is that the unanticipated event provokes a future anomaly in the plan execution. A third and more interesting derivation is whether the new data brings an opportunity to achieve a goal that is not currently contemplated in the active plan.

Goal-directed behaviour is a hallmark of intelligence aimed at discovering the changes that can be applied in the goal of an application in view of the information collected by some unanticipated events. A dynamic formulation of new goals is very helpful in situations where: a) the agent's interests are threaten and a rational anomaly response must be provided; b) goals are no longer achievable and a graceful degradation in the goal achievement is a convenient action; or c) goal achievement in the future is jeopardized, what affects future performance [14]. Thereby, goal-directed reasoning can be regarded as a context-aware responsiveness technique.

This paper is particularly devoted to analyze the first step of a goal formulation process; that is, to answer the question '*when a new goal can be formulated*?'. In some applications, opportunistic behaviour is applied when the sensory input triggers some enabling conditions to accomplish a task, and reactive plans are adopted to detect problems and recover from them automatically as well as to recognize and exploit opportunities [2]. In dynamic and complex environments, like robotics, opportunities are predicted and executed immediately in order to provide quick responsiveness but there is no usually anticipation of future situations.

In less dynamic and reactive environments, typically, goal formulation is considered when an anomaly is detected and/or the system is self-motivated to explore its actions in the world [14]. One approach that has been used to predict or anticipate future plan failures is Case-Based Planning (CBP). In CBP, when a plan fails, it is usually stored with the justification of its failure. This information is then retrieved from the case memory when looking for a similar situation which produced a failure in the past. CBP may be applied before the generation of a plan to anticipate possible problems and avoid situations that failed in the past, or after a plan has been produced to eliminate plans which may fail [13]. CBP presents though several limitations in its application to smart environments: a) predicting future failures is subject to finding an identical case in the case memory; and b) CBP allows only for anticipating a failure but not for detecting opportunities of pursuing a better goal or a new goal.

In this paper, we present a goal reasoning framework that traces the execution of a temporal plan and identifies if the context information acquired from several sources determines a change in the plan goals. Particularly, the reasoner detects situations of future failures and opportunities in the plan execution in the context of temporal planning with deadlines. The framework draws upon TempLM, an approach based on temporal landmarks to handle temporal planning problems with deadlines [11, 12], and we will show how the reasoner works on a temporal plan of a tourist application.

[1] Universitat Politècnica de València, Valencia, Spain, Email: {emarzal,jeibrui,lstarin,onaindia}@dsic.upv.es

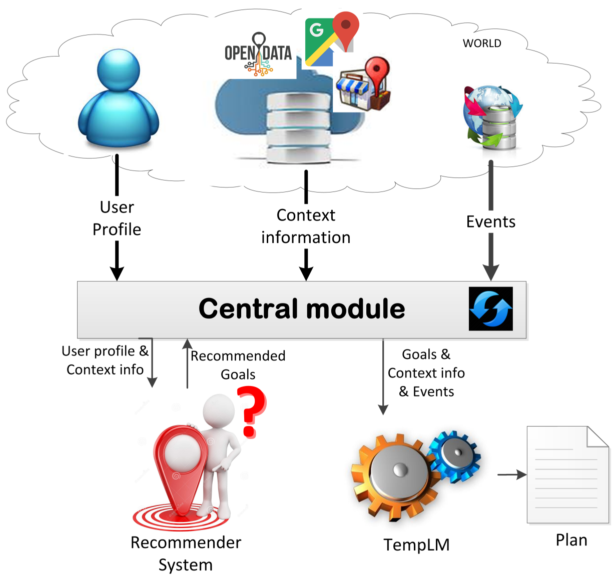


Figure 1. Architecture

This paper is organized as follows. First, a motivating example is introduced, as well as the architecture of our system. Then, some basic definitions referring to automated planning and the main characteristics of our planner TempLM are given. Section 4 introduces new definitions about exogenous events and section 5 explains the analysis that TempLM performs in order to detect future failures or opportunities caused by exogenous events. Finally, section 6 concludes and outlines some future work.

2 TOURIST APPLICATION EXAMPLE

In order to illustrate the foundations and contributions of this paper, a problem in the context of smart tourist will be used. **Smart tourism** involves several components that are supported by information and communication technologies (ICT) [9]. On one hand, it refers to *Smart Destinations*, which are cases of smart cities that apply smart city principles to support mobility, resource availability and allocation, sustainability and quality of life/visits. Second, the *Smart resident/visitor experience* focuses on technology-mediated tourism experiences and their enhancement through personalization, context-awareness and real-time monitoring [4]. Finally, *Smart business* refers to the complex business ecosystem that creates and supports the exchange of touristic resources and the co-creation of the tourism experience. These smart systems include a wide range of technologies in direct support of tourism such as decision support systems and the more recent recommender systems, context-aware systems, autonomous agents searching and mining Web sources, ambient intelligence, as well as systems that create augmented realities.

In this sense, our system aims to improve the resident/visitor experience by reacting in advance to changes in the environment that may cause failures or opportunities in the previously computed agenda. The architecture of our system, shown in Figure 1, is composed of the following modules:

- The **Central module** is the core of the system. It is in charge of generating the initial plan, considering the user profile and the context information. Additionally, it listens to new events that may require to update this plan.
- The **Recommender System** selects the recommended visits for a tourist, given her user profile and the context information.
- The TempLM **planner** develops two main tasks: in first place, it receives the goals computed by the recommender system and the context information and it builds the initial plan for the tourist; then, every time a new event is received by the central module, TempLM analyses it to determine whether the plan needs to be updated.

In this paper, we will focus on the second task of TempLM, that is, on the analysis of events to detect failures or opportunities in the plan.

An example of the problem that we are facing is the following. A tourist arrives to Valencia (Spain) and she is staying at Caro Hotel. She uses our system to obtain a personalized agenda for her visit. First, the recommender system analyses her user profile to select a set of recommended visits with a recommended duration. Let us assume that the user is recommended to visit the Lonja for 1.5 hours, the Cathedral and the Central Market for 2 hours, respectively. These recommended goals, some other user preferences related to the time windows when she prefers to perform the activities along with information about the context, such as the opening hours of the places to visit and the geographical distances between places, are compiled into a planning problem that is formulated as an *Automated Planning Problem* [8], with durative actions and deadlines. This problem is solved by our planner TempLM.

Figure 2 shows the plan obtained for this tourist. The segments at the bottom represent the opening hours of places (i.e. the Lonja is open from 10h until 19h) or the time windows of preferences indicated by the user (i.e. the time for having lunch ranges from 14h to 16h). The green boxes represent the actions in the plan. Specifically, in this domain, three types of actions can be performed: visiting an attraction, having lunch and moving from one place to another. The duration of these actions is determined by the corresponding parameters, that is, the attraction to visit or the origin and destination of the movement action, respectively. For example, the action `(visit T Lonja)` takes from 10:09h to 11:29h. In addition, the visit action must consider the opening hours/preference time window; for example, the action `(visit T Centralmarket)` can only be performed from 15:30h until 19h. The whole plan must fit into the available time of the user indicated in Figure 2 as the timeline, that is, from 10h until 19h. A more detailed description of the compilation of this problem and domain can be found in [10].

If we consider this context, there are some events that may occur during the visit of our tourist. For example:

- The Lonja may close earlier or open later, that is, its available time window may change; this may imply that the visit action has to finish before than expected or it has to be delayed, respectively.
- The Ricard Camarena restaurant may be fully-booked, which implies that another restaurant in the area must be selected.

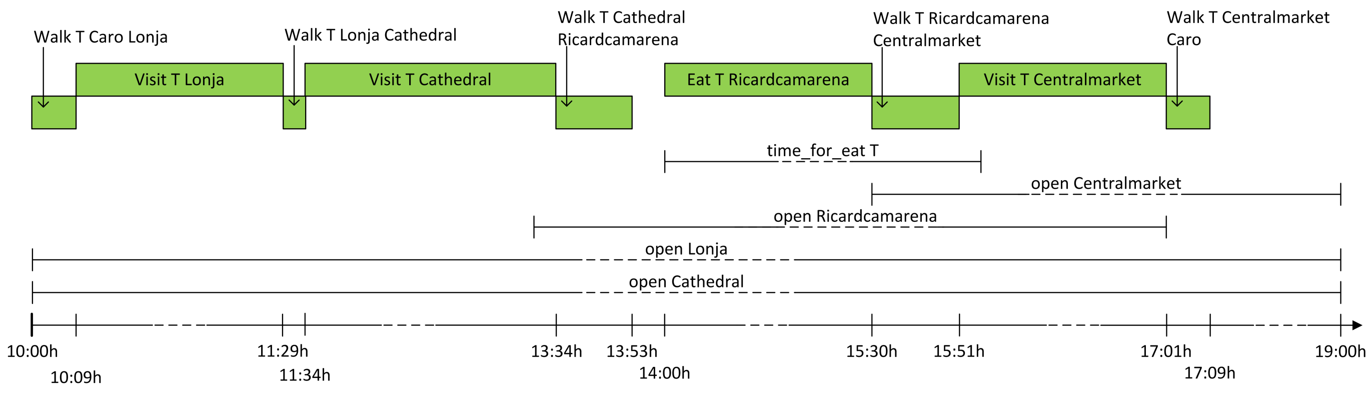


Figure 2. Plan in execution

- The user may take longer to walk from his hotel to the Lonja, so the visit action must be delayed.

These exogenous events are received by the central module from the external data sources and are analyzed by TempLM in order to react in consequence. There are some events that may cause a plan failure (i.e. the attraction is closed) or a plan modification (i.e. the user gets later that expected to an attraction), whereas others may cause the appearance of a free time slot that can be used to include an affordance/opportunity. In this current work, we will focus on the detection of both failures and time slots for opportunities, and we will give some hints about how they can be solved or exploited.

3 BACKGROUND

This section introduces some planning concepts and then it summarizes the main characteristics of our planner TempLM.

3.1 Planning concepts

Definition 3.1 *A* **temporal planning problem with deadline constraints** *is a tuple $\mathcal{P} = \langle P, O, I, G, D\rangle$, where P is the set of propositions, O is the set of ground actions, I and G are sets of propositions that represent the initial state and the goal description and D is a set of deadline constraints of the form (p, t), denoting that proposition p must be achieved within t time units.*

For example, a proposition in I can be (be T Caro), indicating that initially the tourist is at the Caro hotel. It is important to note that, apart from the propositions and functions that are initially true, the initial state I may also contain *timed initial literals (TILs)*. TILs, which were first introduced in PDDL2.2[6], are a very simple way of expressing that a proposition becomes true or false at a certain time point. A TIL can be represented as a pair (p, t), where p is a (positive or negative) proposition and t is a time point. Specifically, if p is a positive proposition, then t indicates the time point at which p becomes true and if it is a negative proposition, then t indicates the time point at which p becomes false. For example, ((not (open Lonja)),19h) is a TIL in I that indicates that the Lonja will close at 19h.

Definition 3.2 *A* **simple durative action** *$a \in O$ is defined as a tuple $(pre_{\vdash}, eff_{\vdash}, pre_{\leftrightarrow}, pre_{\dashv}, eff_{\dashv}, dur)$[5]:*

- *$pre_{\vdash}$ ($pre_{\dashv}$) are the start (end) conditions of a: at the state in which a starts (ends), these conditions must hold.*
- *$eff_{\vdash}$ ($eff_{\dashv}$) are the start (end) effects of a: starting (ending) a updates the world state according to these effects. A given collection of effects $eff_x, x \in \{\vdash, \dashv\}$ consists of:*
  - *eff_x^-, propositions to be deleted from the world state;*
  - *eff_x^+, propositions to be added to the world state*
- *$pre_{\leftrightarrow}$ are the invariant conditions of a: these must hold at every point in the open interval between the start and end of a.*
- *dur is the duration of a.*

An example of an action is shown here:
Action: `Eat(?t: tourist, ?r: restaurant)`
$pre_{\vdash} = \{$ (free_table ?r) $\}$, $pre_{\dashv} = \emptyset$
$pre_{\leftrightarrow} = \{$ (open ?r), (time_for_eat ?t), (be ?t ?r) $\}$
$eff_{\vdash} = \emptyset, eff_{\dashv} = \{$ (eaten ?t) $\}$
$dur =$ (eat_time ?t ?r)

Definition 3.3 *A* **temporal plan** *Π is a set of pairs (a, t), where $a \in O$ and t is the start execution time of a.*

The temporal plan for the running example is shown in Figure 2.

Definition 3.4 *Given a temporal plan Π, the* **induced plan** *Π^* for Π is the set of pairs defined as* [7]*:*

$$\Pi^* = \{(a_{\vdash}, t), (a_{\dashv}, t + dur(a))\}, \quad \forall (a, t) \in \Pi$$

For simplicity, we will refer to any pair in Π^* as an "action". For example, Π^* would include (`(walk T Caro Lonja)`$_{\vdash}$, 10:00) and (`(walk T Caro Lonja)`$_{\dashv}$, 10:09). In the induced plan, we only consider the start and the end time points of the actions in the original temporal plan because these are the time points interesting for building the state resulting from the execution at a certain time point t, as shown below.

Definition 3.5 *Given a state S_t defined as a set of propositions that are true at time t and a pair $(a_x, t) \in \Pi^*$, an action a_x is* **applicable** *in S_t if $pre_x(a) \cup pre_{\leftrightarrow}(a) \subseteq S_t$.*

With this definition, only the start and the end time points of the actions are considered. In order to mitigate the fact that the overall conditions are not checked during the execution of the action, we check them at the start and the end of the action, although this is not consistent with the definition of a durative action in PDDL2.1 (where the *overall* conditions of an action a starting at t have to be fulfilled during the interval $[t + \varepsilon, t + dur(a) - \varepsilon]$).

Definition 3.6 *Given a time point t, let Π_t^* be the subset of actions $(a, t') \in \Pi^*$ such that $t' < t$. A* **temporal state** *S_t is composed by a set of propositions $p \in P$ and a set of TILs (denoted by Γ_t) resulting from applying the actions in Π_t^* from I (denoted by $I \rightarrow_{\Pi_t^*} S_t$), that is, it is assumed that the actions in Π_t^* are applicable in the corresponding state. Formally, we define S_t recursively, where $S_0 = I$:*

$$S_t = S_{t-1} - \left(\bigcup_{\forall (a_x, t') \in \Pi_t^*} eff_x^-(a) \right) \cup \left(\bigcup_{\forall (a_x, t') \in \Pi_t} eff_x^+(a) \right)$$

For example, the state $S_{10:09}$ reached after the execution of the action ((walk T Caro Lonja)$_{\dashv}$,10:09h) will be the same as the initial state but $S_{10:09}$ will contain the new location of the tourist (be T Lonja). It is important to notice that if we compute the state $S_{10:05}$, then the location of the user is unknown because the proposition (be T Caro) is deleted at the start time of the execution of the action and the proposition (be T Lonja) is not added until the end time of the action.

Definition 3.7 *The* **duration** *(makespan) of an induced plan Π^* executed from the initial state of the problem is $dur(\Pi^*) = max_{\forall (a_{\dashv}, t) \in \Pi^*} (t) - T_i$; i.e., the end time of the action that finishes last minus the time point at which the plan starts T_i, assuming that all the actions in Π^* are applicable in the corresponding state.*

For instance, $dur(\Pi^*)$ in Figure 2 is 7 h. and 9 min., because the last action ends at 17:09h and the plan starts at 10h.

Definition 3.8 *An induced plan Π^* is a* **solution for a temporal planning problem with deadline constraints** *$\mathcal{P} = \langle P, O, I, G, D \rangle$ if the following conditions hold:*

1. *$G \subseteq S_g$, where $I \rightarrow_{\Pi_t^*} S_g$, where $t = dur(\Pi^*)$*
2. *$\forall (p, t) \in D : \exists t' \leq t : p \in S_{t'}$, where $I \rightarrow_{\Pi_{t'}^*} S_{t'}$*

This definition indicates that it is not only necessary that a plan Π^* reaches the goals, but also that all the propositions present in D are achieved before the corresponding deadline.

3.2 TempLM

TempLM [11] is a temporal planning approach for solving planning problems with deadlines that has demonstrated an excellent behaviour in the detection of unsolvable problems and the resolution of overconstrained problems. It draws upon the concept of temporal landmark, which is a proposition that is found to necessarily happen in a solution plan in order to satisfy the deadlines of the problem goals. The set of temporal landmarks extracted from a problem along with their relationships form a temporal landmarks graph (TLG) that is conveniently used to take decisions during the construction of the solution plan and for guiding the search process.

Definition 3.9 *A* **Temporal Landmarks Graph (TLG)** *is a directed graph $G = (V, E)$ where the set of nodes V are landmarks and an edge in E is a tuple of the form $(l_i, l_j, \prec_{\{n,d\}})$ which represents a necessary or dependency ordering constraint between the landmarks l_i and l_j, denoting that l_i must happen before l_j in a solution plan.*

Landmarks are also annotated with various *temporal intervals* that represent the validity of the corresponding temporal proposition ([11]):

- The **generation interval** of a landmark is denoted by $[min_g(l), max_g(l)]$. $min_g(l)$ represents the earliest time point when landmark l can start in the plan. $max_g(l)$ represents the latest time point when l must start in order to satisfy D.
- The **validity interval** of a landmark l is denoted by $([min_v(l), max_v(l)])$ and it represents the longest time that l can be in the plan.
- The **necessity interval** of a landmark l is denoted by $([min_n(l), max_n(l)])$ and it represents the set of time points when l is required as a condition for an action to achieve other landmarks.

These intervals are given some initial values that are then updated by means of a propagation method, as explained in [11]. In order to be consistent, both the generation interval and the necessity interval must fall into the validity interval. Figure 3 shows a part of the initial TLG built for this problem. The validity interval of a landmark is represented by a segment, the max_g is indicated by a small bold vertical line (min_g is always equal to min_v) and the necessity interval is represented by a green box inside the segment. For example, the validity interval of the landmark (visited T Cathedral) is [12:05h, 19h] and the $max_g = 19h$; the necessity interval for this landmark is empty.

TempLM applies a search process in the space of partial plans in order to find a solution plan for a problem $\mathcal{P}$. A node in the search tree is defined as $n = (\Pi, S_t, TLG_\Pi)$, where Π is the conflict-free partial plan of n, S_t is the state reached at time $t = dur(\Pi)$ after the execution of Π from I and TLG_Π is the refined TLG after taking into account the information in Π. Given a node n of the search tree, a successor of n results from adding an executable action a to the partial plan of n, provided that a does not cause conflicts with the actions of n. Hence, the plan of the successor node will contain a newly added action, information which can be exploited to find new temporal landmarks in the TLG [12].

4 CHANGES IN THE ENVIRONMENT

This section introduces some definitions related to the management of changes in the environment. We assume that in our system changes happen due to exogenous events.

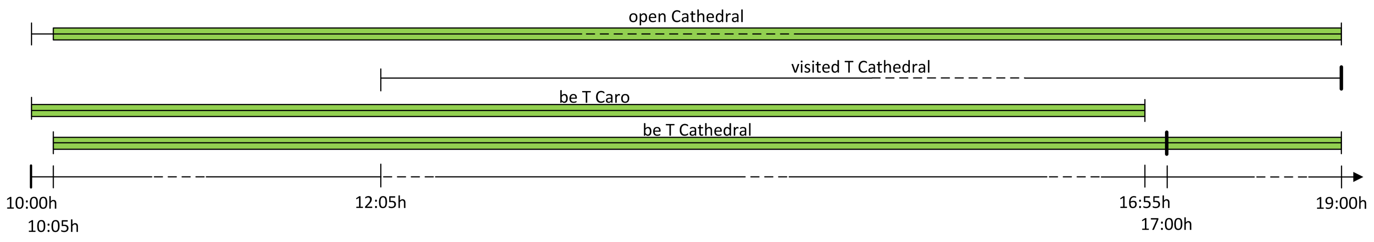


Figure 3. Initial partial TLG for this problem

Definition 4.1 *We define an* **exogenous event** θ *as a tuple* (t_a, p, t_c)*, where* t_a *is the* arrival time point*, that is, the time point when the exogenous event is received in the system, and* (p, t_c) *denotes a TIL.*

For example, let ((not (open Centralmarket)),17h) be a TIL in I that indicates that the Central Market will close at 17h. If at 14h it is known that the Central Market will close one hour earlier, the exogenous event that will be received by the system is the following: (14h, (not (open Centralmarket)), 16h) and it will invalidate the previous TIL.

Definition 4.2 *We say that two exogenous events* (t_a, p, t_c) *and* (t'_a, p', t'_c) *are* linked[2] *if they refer to the same proposition* $(p = p')$ *and their arrival time is the same* $(t_a = t'_a)$*.*

We can indicate a modification in the actual duration of an action with two linked exogenous events that refer to the proposition(s) that will be achieved as a result of the execution of the action. The first one deletes a proposition p at the time it was expected and the second event adds that proposition at the new time. For example, if at 13h it is known that the action `(eat T RicardCamarena)` to be executed from 14h to 15:30h will take 30 minutes more than expected, these two linked exogenous events are received by the system: (13h, (not (eaten T)), 15:30h) and (13h, (eaten T), 16h). The first one deletes the expected event of the tourist having finished lunch at 15:30h and the second adds the proposition at the time when the tourist will actually finish having lunch.

We denote by t_{cur} the current execution time when an event θ is received and by $S^o_{t_{cur}}$ the current (observed) state. Therefore, $S^o_{t_{cur}}$ will contain the propositions that are true in the current world and the functions with their actual value in the current world. Moreover, let $\Gamma_{t_{cur}-\varepsilon}$ be the set of TILs in the state at the time immediately prior to t_{cur}. $S^o_{t_{cur}}$ will contain the TIL in θ plus the TILs in $\Gamma_{t_{cur}-\varepsilon}$ that are consistent with θ. For example, as Figure 2 shows, $\Gamma_{14h-\varepsilon}$ contains, among others, the TILs in Table 1 (top). If the event θ =(14h, (not (open Centralmarket)), 17:30h) is received, then Γ_{14h} in $S^o_{t_{cur}}$ will be updated as shown in Table 1 (bottom). That is, the time window in which the Central Market remains open is updated from [15:30h, 19h] to [15:30h, 17:30h].

Definition 4.3 *We define the* **executed partial plan at** t_{cur}*, denoted by* Π^*_{ex}*, as:*

$$\Pi^*_{ex} = \{(a, t) \in \Pi^* : t < t_{cur}\}$$

[2] For simplicity, we denote by θ a single event or two linked events.

Table 1. TILs at 14h

$\Gamma_{14h-\varepsilon}$
((open Centralmarket), 15:30h),
((not (open Centralmarket)), 19h),
((not (open Ricardcamarena)), 17h), ...
Γ_{14h}
((open Centralmarket), 15:30h),
((not (open Centralmarket)), 17:30h),
((not (open Ricardcamarena)), 17h), ...

Definition 4.4 *We define the* **remaining partial plan at** t_{cur}*, denoted by* Π^*_r*, as:*

$$\Pi^*_r = \{(a, t) \in \Pi^* : t \geq t_{cur}\}$$

Definition 4.5 *We define the* **resulting state of** Π^*_{ex} *(*$S_{t_{cur}}$*) as the state reached after executing the actions in* Π^*_{ex}*, that is:* $I \rightarrow_{\Pi^*_{ex}} S_{t_{cur}}$

Definition 4.6 *We define the* **difference** *between two states* S_i *and* S_j *as:*

$$Diff(S_i, S_j) = (S_i - S_j) \cup (S_j - S_i)$$

Definition 4.7 *A* **discrepancy** *is a non-empty difference between the state that should have been reached with the executed part of the plan* $S_{t_{cur}}$ *and the current observed state* $S^o_{t_{cur}}$*, that is:* $Diff(S_{t_{cur}}, S^o_{t_{cur}}) \neq \emptyset$*.*

Obviously, the discrepancy between these two sets will at least contain the TIL from the event θ. For example, if we consider the example in Table 1, the event that arrives at 14h is reflected in the resulting Γ. Discrepancies may cause failures or opportunities in the plan.

Definition 4.8 *We say that there is a* **failure in the plan** *if:*

1. *There is a discrepancy at* t_{cur}*, and*
2. $S^o_{t_{cur}} \rightarrow_{\Pi^*_r} S_g : G \not\subseteq S_g$*, that is, the plan does not reach the goal due to this discrepancy.*

Note that Π^* was a solution for the initial planning problem, that is, it was executable from I. This definition indicates that there has been a change in the environment at a certain time point between I and t_{cur} which causes a failure in the execution of action a and, consequently, a failure in the execution of the plan.

Definition 4.9 *We say that there is an* **opportunity in the plan** *if:*

1. *There is a discrepancy at t_{cur}, and*
2. *$S^o_{t_{cur}} \rightarrow_{\Pi^*_r} S_g : G \subseteq S_g$, that is, the plan reaches the goal in spite of this discrepancy, and*
3. *this discrepancy causes a different execution of the remaining plan, that is, there is a time point t where the state reached from $S_{t_{cur}}$ is different from the state reached from $S^o_{t_{cur}}$; formally, given $t \in [t_{cur}, dur(\Pi^*)]$, let Π' be the set of actions in Π^*_r scheduled to start between t_{cur} and t; we define S^o_t and S_t as $S^o_{t_{cur}} \rightarrow_{\Pi'} S^o_t$ and $S_{t_{cur}} \rightarrow_{\Pi'} S_t$, respectively; the execution of the remaining plan is different if $Diff(S_t, S^o_t) \neq Diff(S_{t_{cur}}, S^o_{t_{cur}})$.*

An opportunity in this case is defined as a discrepancy that still allows to reach the goals but that causes a difference in the execution of the original plan.

It could be the case that a discrepancy does not cause a failure or an opportunity, because it affects an object that is not considered in the current plan.

5 DETECTION OF FAILURES AND OPPORTUNITIES IN TempLM

The TLG that TempLM builds to solve a planning problem gives an schema of which propositions must be achieved in the plan, and when they must be achieved in order to reach the goals on time. Additionally, a Temporal Proposition Graph, which gives an exact picture of the propositions that are achieved in the plan and when they are achieved, is defined:

Definition 5.1 *A* **Temporal Proposition Graph (TPG)** *for a given induced plan Π^* is a representation of the time interval in which a proposition $p \in P$ is true during the execution of Π^*. This is denoted as the validity interval of the proposition p. A TPG also defines the generation and necessity intervals of a proposition, with the same meaning as for a landmark in the TLG.*

Figure 4 shows the TPG for the plan in Figure 2. In this case, all the propositions that appear during the execution of the plan are represented. It can be observed that the validity interval of the propositions in the TPG is much smaller than in the TLG (see Figure 3), given that the TLG is just an estimation, whereas the TPG is an accurate representation of the solution plan. For example, in Figure 3, proposition (be T Cathedral) ranges from 10:05h until 19h, whereas in Figure 4 it is shrunk to the interval [11:34h, 13:34h].

The aim of this section is to explain how both the TLG and the TPG can be used in order to detect failures and opportunities due to exogenous events that may arrive during the execution of a plan, and also to give some hints about how they can be solved or exploited.

As explained in section 2, when a new exogenous event is detected by the central module, TempLM receives the current execution time t_{cur}, the current observed state $S^o_{t_{cur}}$ and the exogenous event θ as input. Then, TempLM computes the state that should have been reached as a result of the execution of the plan until time t_{cur}, that is, it computes $S_{t_{cur}}$. The next step is to analyze whether the event θ has caused a

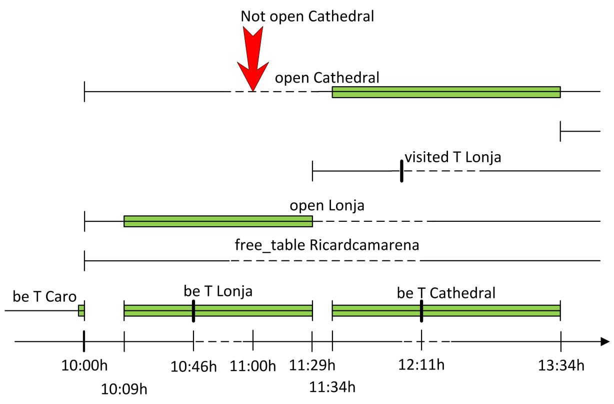


Figure 5. Example of future failure

discrepancy between $S_{t_{cur}}$ and $S^o_{t_{cur}}$, as indicated in Definition 4.7. In case of a discrepancy between both states, several cases can be found.

Case 1. The discrepancy does not affect the plan
This case corresponds to a situation where the exogenous event $\theta = (t_a, p, t_c)$ affects a proposition that has not any influence in the plan. This means that the proposition p that causes the discrepancy is not present in the TPG of the plan and it is not mutex [3] with any other proposition in the TPG. Formally, TempLM does not detect any failure or opportunity if:

$$p \notin TPG_{\Pi^*} \wedge \neg \exists p' \in TPG_{\Pi^*} : mutex(p, p')$$

For example, let us assume that this exogenous event (10, (not (free_table Cantinella)), 13h) is received, indicating that the Cantinella restaurant will not have free tables at 13h. This does not affect the current plan, given that the restaurant where the tourist is going to have lunch is a different one.

Case 2. A failure is detected
The second analysis is aimed at detecting a failure caused by an exogenous event $\theta = (t_a, p, t_c)$. We distinguish two situations: when the event causes a present failure, that is, $t_a = t_c = t_{cur}$ or when it causes a future failure, that is, $t_a = t_{cur} < t_c$. In both cases, a proposition $p' : mutex(p, p')$ which belongs to the TPG is deleted before expected when it is still needed to reach the goals. Formally, TempLM detects a failure if:

$$p' \in TPG_{\Pi^*} \wedge t_c < max_n(p')$$

For example, if at $t_{cur} = 10h$ an event indicating that the Cathedral will be closed at 11h arrives, expressed as $\theta =$(10, (not (open Cathedral)), 11h), a failure is caused because $t_c =$11h, $mutex(($not(openCathedral)$), ($openCathedral$))$ and $t_c < max_n($open Cathedral$) =$13:34h, as it can be observed in Figure 5. In this case, TempLM is able to easily predict a future failure due to an exogenous event.

Once the failure is detected, two situations can occur: (1) the problem is unsolvable or (2) the problem can be solved using other elements defined in the context. In order to distinguish these situations, TempLM finds the node $(\Pi, S, TLG_{\Pi^*_{ex}})$

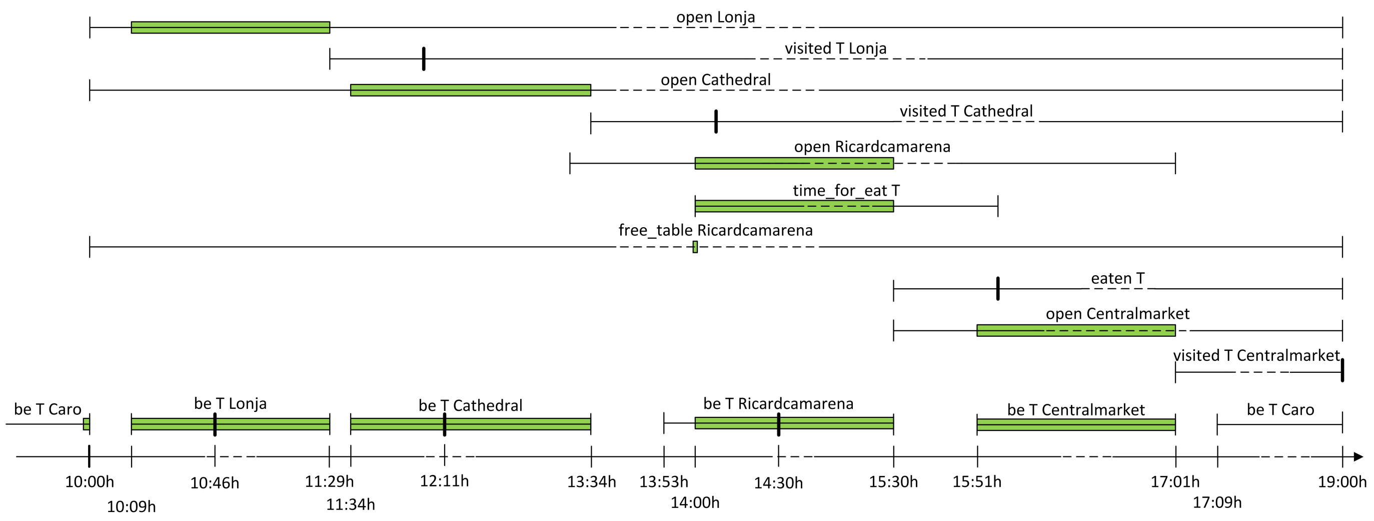


Figure 4. Temporal Proposition Graph of the plan

in the search space built when solving the original problem, where $S = S_{t_{cur}}$ computed from Π^*_{ex}. If the TLG in this node is updated with the information about the event θ, this new information is propagated across the TLG (obtaining $TLG^{\theta}_{\Pi^*}$) and an inconsistency [11] between two intervals of a given proposition is found, then the problem is unsolvable. It is important to remark that the propositions in the TLG are necessary to solve the problem (they are landmarks); therefore, if an inconsistency is found in the intervals associated to a landmark, then the problem is unsolvable. Formally:

$$\exists p \in TLG^{\theta}_{\Pi^*} : max_g(p) \notin [min_v(p), max_v(p)]$$

For example, if the event (10, (not (open Cathedral)), 11h) arrives, then the problem is unsolvable because, due to θ, the value of the validity interval of (open Cathedral) changes, i.e. max_v(open Cathedral) $= 11h$. This information is propagated to the proposition (visited T Cathedral) in the TLG^*_{Π}, which updates max_g(visited T Cathedral) $= 11h$, indicating that, in order to reach the goals, the Cathedral must be visited before 11h. Given that this value does not belong to the validity interval of (visited T Cathedral), as it can be observed in Figure 3, there is an inconsistency in these intervals. Therefore, we can affirm that the problem, after the arrival of the event, is unsolvable. In fact, there is not a plan that satisfies the goals, because even in the case that the Cathedral is the first visit, it takes 2 hours and then, it would last until 12:05h, which is later than the time of closing.

In any other case, TempLM performs the search of a new plan starting from Π^*_{ex}, that is, Π^*_r is discarded and substituted by the eventually new plan found. If, for example, an event indicating that the Ricard Camarena restaurant will be fully-booked at 13:45h (θ =(13, (not (free_table Ricardcamarena)),13:45h) arrives, then TempLM detects a failure because t_c =13:45h $<$ max_n(free_table Ricardcamarena) =14:00h. In this situation, TempLM would be able to find a different solution plan, because having lunch at Ricard Camarena restaurant is not a hard goal; the hard goal is (eaten T), which can be achieved by having lunch in any other restaurant. Therefore, a new plan from Π^*_{ex} in which the tourist has lunch in a different restaurant could be found by TempLM.

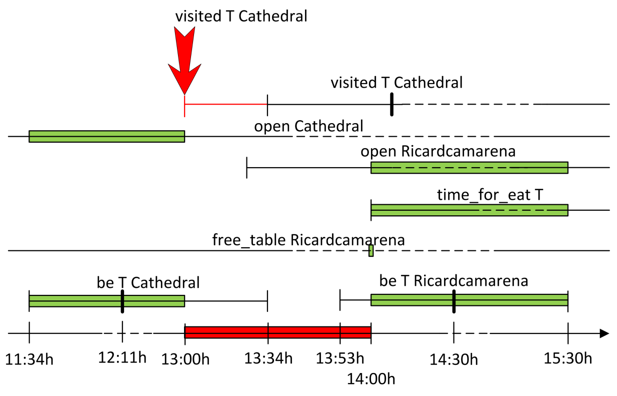


Figure 6. First example of a future opportunity

Case 3. An opportunity is detected

The last analysis is devoted to detect an opportunity when a new exogenous event $\theta = (t_a, p, t_c)$ arrives. In this case, p is a proposition that is achieved before than expected, which permits to consider an action as finished before its complete execution. Formally, TempLM detects an opportunity if:

$$p \in TPG_{\Pi^*} \wedge t_c < min_v(p)$$

For example, the arrival of an event in which the visit to the Cathedral finishes at 13h, i.e. θ =(13, (visited Cathedral), 13h), causes the detection of an opportunity. The red arrow in Figure 6 indicates the new time point at which the proposition (visited Cathedral) is achieved and the red rectangle indicates

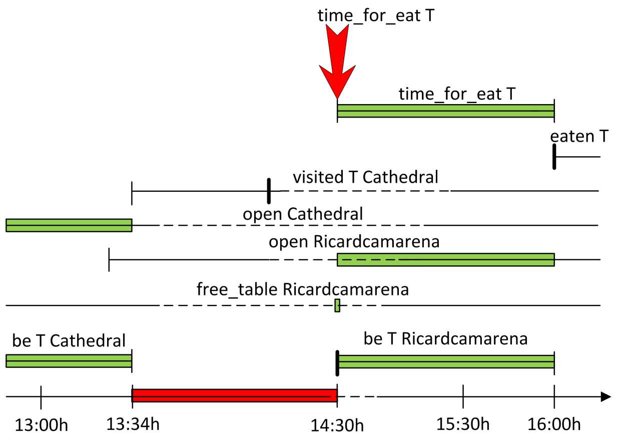


Figure 7. Second example of a future opportunity

the available interval of time thanks to this event (which can be considered as an opportunity). It can be observed that, in this example, t_{cur} =13h < min_v((visited Cathedral)) =13:34h and the necessity interval of (open Cathedral) and (be T Cathedral) are shrunk because (visited Cathedral) has been achieved before than expected. At this moment, the recommender system could be invoked to obtain a new visit (a new goal) to be performed during the time interval between 13h and 14h, before (be T Ricardcamarena). TempLM would obtain a plan to reach this new goal and then the remaining original plan would be executed.

There is another case where TempLM can detect an opportunity:

$$p \in TPG_{\Pi^*} \wedge t_c > min_v(p) \wedge$$

$$\neg \exists q \in TPG^{\theta}_{\Pi^*} : max_g(q) \notin [min_v(q), max_v(q)]$$

This case represents the situation when a proposition is achieved later than expected, but it does not affect the achievement of the remaining goals. That is, there is not any proposition q whose intervals are inconsistent after updating the TPG_{Π^*} with the exogenous event and propagating the information across the TPG_{Π^*} to obtain $TPG^{\theta}_{\Pi^*}$.

For example, let us assume that the tourist now prefers to have lunch between 14:30h and 16h, instead of between 14h and 16h, but the action still takes 90 minutes. This situation triggers the following exogenous event: θ =(13, (time_for_eat T), 14:30h). This event causes that TempLM detects an opportunity because min_v(time_for_eatT) =14:00h and t_c =14:30h; this provokes that min_v(time_for_eat T) is updated and this information is propagated to min_v(eatenT) (max_g(eaten T) still belongs to the corresponding validity interval). Additionally, the necessity intervals of (time_for_eat T), (open Ricardcamarena) and (be T Ricardcamarena) are also updated, as it is shown in Figure 7. The available time for the opportunity, also shown in Figure 7, can be used to reach a new goal recommended by the recommender system, as explained above.

Thanks to the analysis presented in this section, we have been able to show how TempLM is able to predict future failures or opportunities.

6 CONCLUSIONS AND FURTHER WORK

This paper has introduced a goal reasoning framework where context information acquired from several external sources determines a change that may affect the execution of the current plan. This change may cause a failure or an opportunity. We have shown how the planning system, namely TempLM, is able to predict both failures and opportunities thanks to the analysis of the Temporal Landmarks Graph and the Temporal Propositions Graph built for the given problem.

As for further work, our aim is to implement in TempLM the analysis described in this paper. Then, we will be able to test the system in a real environment. This will imply an analysis to decide which events should be considered among the huge amount of context information that is supplied by external sources.

ACKNOWLEDGMENTS

This work has been partially supported by Spanish Government Project MINECO TIN2014-55637-C2-2-R.

REFERENCES

[1] E. Aarts and J.L. Encarnaao, *True visions: the emergence of ambient intelligence*, New York. Springer, 200.
[2] M. Beetz and H. Grosskreutz, 'Probabilistic hybrid action models for predicting concurrent percept-driven robot behavior', *J. Artif. Intell. Res. (JAIR)*, **24**, 799–849, (2005).
[3] A. Blum and M. Furst, 'Fast planning through planning graph analysis', *Artificial Intelligence*, **90**(1-2), 281–300, (1997).
[4] D. Buhalis and A. Amaranggana, 'Smart tourism destinations enhancing tourism experience through personalisation of services', in *Information and Communication Technologies in Tourism 2015*, 377–389, Springer, (2015).
[5] A. J. Coles, A. I. Coles, M. Fox, and D. Long, 'Colin: Planning with continuous linear numeric change', *Journal of Artificial Intelligence Research*, **44**, 1–96, (2012).
[6] S. Edelkamp and J. Hoffmann, 'Pddl2. 2: The language for the classical part of the 4th international planning competition', *4th International Planning Competition (IPC'04)*, (2004).
[7] M. Fox and D. Long, 'Pddl2. 1: An extension to pddl for expressing temporal planning domains.', *J. Artif. Intell. Res.(JAIR)*, **20**, 61–124, (2003).
[8] M. Ghallab, D. Nau and P. Traverso, *Automated Planning. Theory and Practice.*, Morgan Kaufmann, 2004.
[9] U. Gretzel, M. Sigala, Z. Xiang, and C. Koo, 'Smart tourism: foundations and developments', *Electronic Markets*, **25**(3), 179–188, (2015).
[10] J. Ibañez, L. Sebastia, and E. Onaindia, 'Planning tourist agendas for different travel styles', in *Proceedings of the European Conference on Artificial Intelligence (ECAI)*, p. in press, (2016).
[11] E. Marzal, L. Sebastia, and E. Onaindia, 'On the use of temporal landmarks for planning with deadlines', in *Proc. of the 24th International Conference on Automated Planning and Scheduling*, pp. 172–180. AAAI Press, (2014).
[12] E. Marzal, L. Sebastia, and E. Onaindia, 'Temporal landmark graphs for solving overconstrained planning problems', *Knowledge-Based Systems*, (2016).
[13] L. Spalazzi, 'A survey on case-based planning', *Artificial Intelligence Review*, **16**, 3–36, (2001).
[14] S. Vattam, M. Klenk, M. Molineaux, and D. W. Aha, 'Breadth of approaches to goal reasoning: A research survey', Technical report, DTIC Document, (2013).

An Intelligent System for Smart Tourism Simulation in a Dynamic Environment

Mohannad Babli, Jesús Ibáñez, Laura Sebastiá, Antonio Garrido, Eva Onaindia [1,2]

Abstract. In this paper, we present a smart tourism system that plans a tourist agenda and keeps track of the plan execution. A Recommendation System returns the list of places that best fit the individual tastes of the tourist and a planner creates a personalized agenda or plan with indication of times and durations of visits. The key component of the system is the simulator in charge of the plan monitoring and execution. The simulator periodically updates its internal state with information from open data platforms and maintains a snapshot of the real-world scenario through live events that communicate sensible environmental changes. The simulator builds a new planning problem when an unexpected change affects the plan execution and the planner arranges the tourist agenda by calculating a new plan.

1 INTRODUCTION

The exponential growth of the Internet of Things (IoT) and the surge of open data platforms provided by city governments worldwide is providing a new foundation for travel-related mobile products and services. With technology being embedded on all organizations and entities, the application of the smartness concept to address travellers' needs before, during and after their trip, destinations could increase their competitiveness level [2].

Smart tourism differs from general e-tourism not only in the core technologies of which it takes advantage but also in the approaches to creating enhanced at-destination experiences [8]. In the work [14], authors identify the requirements of smart technology integration in personalized tourism experiences including information aggregation, ubiquitous mobile connectedness and real time synchronization.

Many tourism applications provide a personalized tourist agenda with the list of recommended activities to the user [12, 13, 5, 16, 15]. In many cases, these systems provide a dynamic interaction that allows the user to interact with such agenda by adding or removing activities or changing their order. Additionally, the use of GPS in mobile devices allows recommender systems to locate the future user's location and recommend the most interesting places to visit. However, most of these applications work with fixed and static information throughout the execution of the activities. In other words, they do not easily react before changes in the world; for instance, a museum that closes, a restaurant which is fully booked, a bus route that is now diverted, etc. This has critical implications on the way tourists regard their experiences. Activities, even pre-designed in advance, must be dynamically adapted and personalized in real time. One essential prerequisite for smart technology is real time synchronization, which implies that information is not limited to a-priori collection but can be collected and updated in real time [14].

Creating an agile and adaptable tourist agenda to the dynamic environment requires tracing the plan and checking that activities happen as expected. This involves plan monitoring and possibly finding a new tourist agenda organization in case some particular activity can not be realized. In this paper, we relate our experience with a context-aware smart tourism simulator.

From the monitoring and simulation perspective, there exist many frameworks for different programming languages that support discrete event-based simulators (e.g. `http://jamesii.informatik.uni-rostock.de/jamesii.org`, `http://desmoj.sourceforge.net/home.html`, `http://simpy.readthedocs.io/en/latest/`). Although they can be programmed for very particular scenarios, they fail to take a general domain description and simulate its behavior in highly dynamic environments. At this stage, planning technology can be very valuable. The Planning Domain Definition Language (PDDL) provides a simple way to define the physics of the domain (a tourism domain in our case, although it is valid for any other scenario) and the particular problem that instantiates such a domain. In PDDL we can define the activities to be executed similarly to rules, with their preconditions, effects and other interesting features like duration, cost, reward, etc. The result of using planning in a tourism domain is a plan, represented as the agenda of activities the user will follow. The plan needs to be validated, executed and adapted, if necessary, to new information. *VAL* is a plan validation tool [9] that can be used to validate and simulate a successful plan execution. However, *VAL* does not consider the dynamic changes of the world and, consequently, it cannot react to them.

In this work, we present a smart tourism system that attempts to overcome the previous limitations. Particularly, we use a PDDL tourism description that can be easily adapted to many different scenarios with new activities, preconditions, effects and points of interest. We run a planner to obtain a plan and we simulate the plan execution like in a real context, dynamically simulating changes in the environment. The simulator reacts to the changes by checking whether the real

[1] Universitat Politècnica de València, Valencia, Spain, Email: {mobab, jeibrui, lstarin, agarridot, onaindia}@dsic.upv.es

[2] This work has been partially supported by Spanish Government Project MINECO TIN2014-55637-C2-2-R

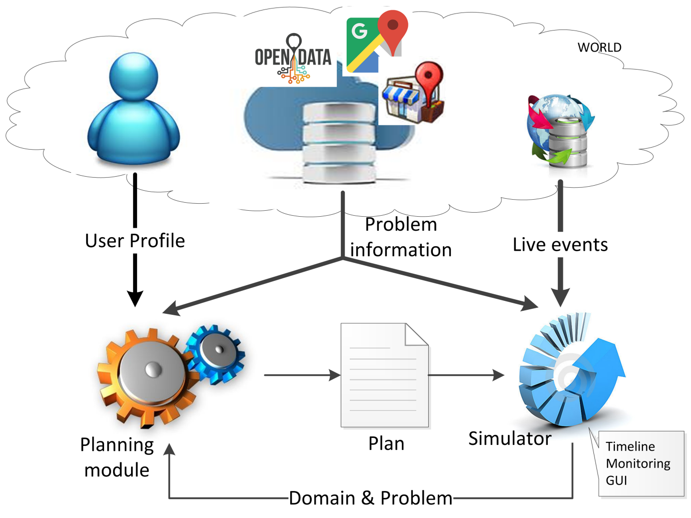


Figure 1. GLASS Architecture

world matches the expected one or not and reformulating the PDDL problem if a failure in the plan is detected, while trying to reuse as many of the original set of recommended activities as possible.

This paper is organized as follows. Next section outlines the architecture of the smart tourism system. Section 3 presents the planning description of the tourism domain and highlights the main components of a planning problem. Section 4 describes the simulator behaviour with a special emphasis on the reformulation of a planning problem. Section 5 presents a case of study of a tourist plan in the city of Valencia in Spain and last section concludes.

2 GLASS ARCHITECTURE

This work is part of the ongoing GLASS[3] (Goal-management for Long-term Autonomy in Smart citieS) project applied to a tourism domain. The idea here is to apply different strategies for dividing the set of goals (i.e. tourist recommendations based on previous plans executions by other tourists) for each user by using different utility recommendations systems. The GLASS architecture is shown in Figure 1. As can be seen, the architecture simply consists of a two-process loop: planning module and simulation+monitoring that share common information.

On the one hand, the input information is retrieved from different data sources. First, we need the user profile with the explicit interests of the user, the goals and preferences (such as points of interest he/she wants to visit), and temporal constraints. Second, we need to access a different set of databases that identify and categorize the points of interest (e.g. museums, restaurants, etc.), their timetable, and geographic sources to find out routes, distances and times between points. Currently, we use standard APIs, such as Google Places[4] and Directions[5] for this, but other open databases would be also valid, such as OpenStreetMap[6]. Third, there exists a snapshot of the environment or real-world scenario where the plan is executed. Since this world is highly dynamic and can change frequently (e.g. the opening hours of a museum has changed, or a restaurant is fully booked and the duration for having lunch is longer than expected), we get the new information as live events.

On the other hand, the planning module takes the user profile and the problem information to create a planning scenario in a PDDL format, as described in Section 3. We need to model the user preferences, constraints and the actions that the user can do, such as visit or move. The output of this is a plan, as a sequence of actions the user has to execute. As a proof of concept, in GLASS we actually simulate that execution rather than having a real execution that would require a true group of tourists equipped with sensoring information to their current geographic positions, pending and already satisfied goals, etc. This simulation process, more detailed in Section 4, takes the plan and creates a timeline structure to run a timed events based execution. It simulates and monitors the resulting states of the world, according to the changes in the plan, that is the effects that actions provoke and, probably, being also modified by the live events. This simulation is shown in a specially designed Graphical User Interface that shows what is happening at any time. If the expected state is different to the real state, i.e. a discrepancy has been discovered, because some live events prevent the remaining actions in the plan from being executed, a (re)planning module becomes necessary. The idea is to reuse the same planning module, thus closing the loop, with a new PDDL domain+problem specification to adapt the plan to the new emerging scenario.

3 PLANNING MODULE

The main goal of our system is to provide a personalized plan to a given tourist. This resulting plan has to reflect the preferences of the tourist according to his/her profile (demographic classification, the places visited by the user in former trips and the current visit preferences). Moreover, in order to build this plan, the duration of the activities to perform, the opening hours of the places to visit and the geographical distances between places (time to move from one place to another) needs also to be considered. Thus, solving this problem requires the use of a planning system capable of dealing with durative actions to represent the duration of visits; temporal constraints to express the opening hours of places and soft goals for the user preferences. Soft goals will be used to denote the preferable visits of the user, the non-mandatory goals that we wish to satisfy in order to generate a good plan that satisfies the user but that do not have to be achieved in order for the plan to be correct. In our case, the goal of visiting a recommended place according to the user profile (the list of potential places that the user can visit is returned by a Recommender System), is defined as a soft goal (more details in section 3.2). Among the few automated planners capable of handling temporal planning problems with preferences, we opted for OPTIC [1] because it handles the version 3.0 of the popular Planning Domain Definition Language (PDDL) [7], including soft goals.

[3] More info at `http://www.plg.inf.uc3m.es/~glass`
[4] More info at `https://developers.google.com/places/web-service`
[5] More info at `https://developers.google.com/maps/documentation/directions`
[6] More info at `http://wiki.openstreetmap.org/wiki/API`

All the information required by OPTIC to build the plan is compiled into a planning problem encoded in PDDL3.0 language, as described in the following sections.

3.1 Initial state

The initial state of a planning problem describes the state of the world when the plan starts its execution. The initial state must reflect the opening hours of the places to visit, the distances between them, the initial location of the user, etc. Some information is expressed with predicates and functions, while other information is represented by Timed Initial Literals (TILs). TILs, which were first introduced in PDDL2.2, are a very simple way of expressing a restricted form of exogenous events that become true or false at given time points [3].

The predicate `(be tourist ?l)` is used to represent the location of the user and the pair of TILs `(at 0 (active tourist))` and `(at` $t_{available}$ `(not (active tourist)))` determine the available time of the user for the tour, where $t_{available}$ is the difference between the time when the tour starts and finishes. The time indicated in the TILs is relative to the starting time of the plan; that is, $t_{available} = 540$ refers to 7pm if the plan starts at 10am. Another pair of TILs is used to define the time window in which the tourist prefers to have lunch. For example, if the preference is between 2pm and 4pm, the TILs are `(at 240 (time_for_eat tourist))` and `(at 360 (not (time_for_eat tourist)))`.

The duration of a particular visit `?v` for a tourist `?t` is defined through the numeric function `(visit_time ?v ?t)`. Assigning a value to a numeric function gives rise to a numeric-valued *fluent*; for example, `(= (visit_time Lonja tourist) 80)` (details about calculating the duration of the visit are shown in the following section). The list of available restaurants is given through the predicate `(free_table ?r)`; for example, `(free_table ricard_camarena)`. For each restaurant, we define the time slot in which it serve meals, which may depend on the type of restaurant, closing time of the kitchen or other factors. Both, places to visit and restaurants, have an opening hour and a closing hour that are specified by a TIL: `(at` t_{open} `(open` a`))` and `(at` t_{close} `(not (open` a`)))`, to indicate when the place/restaurant is not longer available. For example, `(at 0 (open Lonja))`, `(at 540 (not (open Lonja)))`.

The distance between two locations `?a` and `?b` is defined by the function `(moving_time ?a ?b)`, which returns the time in minutes needed to travel from `?a` to `?b` by using the travel mode preferred by the user. The time to move between two places is represented through a numeric fluent (e.g., `(= (moving_time caro_hotel Lonja) 9)`), where the value 9 is taken from Google Maps.

3.2 Goals and preferences

We handle two types of goals: *hard goals*, that represent the realization of an activity that the user has specified as mandatory (e.g., the final destination at which the user wants to finish up the tour `(be tourist caro_hotel)`); and *soft goals or preferences*, that represent the realization of a desirable but non-compulsory activity (e.g., visiting the Lonja `(preference v3 (visited tourist Lonja))`). Preferences are expressed in PDDL3.0 so we need to define how the satisfaction, or violation, of these preferences will affect the quality of a plan. The penalties for violation of preferences (costs) will be handled by the planner in the plan metric to optimize at the time of selecting the best tourist plan; i.e., the plan that satisfies the majority of the tourist preferences and thereby minimizes the penalties for violation.

The objective is to find a plan that achieves all the hard goals while minimizing a plan metric to maximize the preference satisfaction; that is, when a preference is not fulfilled, a penalty is added to the metric. Specifically, we define penalties for non-visited POIs and for travelling times.

The *penalty for non-visited places* is aimed to help the planner select the activities (tourist visits) with a higher priority for the user. Given a plan Π, this penalty is calculated as the ratio between the priority of the activities not included in Π and the priority of the whole set of activities recommended to the user (RA):

$$P_{non\_visited} = \frac{\sum_{a \in RA-\Pi} Pr^a}{\sum_{a \in RA} Pr^a}$$

For example, if the priority for visiting the Lonja is 290, and the sum of the priorities of all the visits is 2530, the penalty for not visiting the Lonja would be expressed in PDDL as: `( / (* 290 (is-violated v3)) 2530)`. The priority of the activities (Pr^a) is calculated by a hybrid Recommender System (RS) which returns a value between 0 and 300 according to the estimated degree of interest of the user in activity a. The value of Pr^a is also used by the RS to return a time interval that encompasses the minimum and maximum recommendable visit duration following a normal distribution $N\left(\mu_a, \sigma_a^2\right)$, where μ_a represents the average visit duration for a typical tourist [10]. Thus, the higher the value of Pr^a, the longer the visit duration.

The *penalty for movements* enforce a reduction in the time spent in travelling from one location to another, so that closer activities are visited consecutively. This penalty is calculated as the duration of all *move* actions of Π (Π_m):

$$P_{move} = \frac{\sum_{a \in \Pi_m} dur(a)}{dur(\Pi)}$$

The function `(total_moving_time tourist)` accumulates the time spent in transportation actions, so this penalty would be defined in PDDL as: `( / (total_moving_time tourist) 540)`. The plan metric to be minimized by the planner is expressed as the sum of both penalties: $P_{total} = P_{non\_visited} + P_{move}$.

3.3 Actions

We define three actions in the tourism domain. The action to **move** from one location to another is defined in Figure 2. It takes as parameters the user `?per`, the initial location `?from` and the destination `?to`. The duration of the action is set to the estimated/actual time to go from `?from` to `?to`, which is stored in the database. The preconditions for this action to be applicable are: (1) the user is at location `?from` and (2) the time window for the available time of the user is active during the whole execution of the action. The effects of the action assert that (1) the user is not longer at the initial location, (2) the user is at the new location at the end of the action and (3)

```
(:durative-action move
  :parameters (?per - person ?from - location
              ?to - location)
  :duration (= ?duration (moving_time ?to ?from) )
  :condition
    (and
      (at start (be ?per ?from))
      (over all (active ?per)))
  :effect
    (and
      (at start (not (be ?per ?from)))
      (at start (walking ?per))
      (at end (be ?per ?to))
      (at end (not (walking ?per)))
      (at end (increase (total_moving_time ?per)
                (moving_time ?from ?to)))))
```

Figure 2. Action move of the tourism domain

the time spent in move actions is modified according to the movement duration. In order to indicate the position of the user during the execution of the action, a `walking` predicate is asserted at the start of the action and deleted at the end of the action. In this paper, we only consider *walking* as the move action; however, more transportation modes according to the user's preferences can be included; e.g., cycling, driving, and public transport, as in [10] .

The action to **visit** a place is defined in Figure 3, whose parameters are the place to visit `?mon` and the user `?per`. The duration of the action is defined by the function `(visit_time ?mon ?per)`. The conditions for this action to be applicable are: (1) the user is at `?mon` during the whole execution of the action; (2) `?mon` is open during the whole execution of the action and (3) the time window for the available time of the user is active. The effects of the action assert that the place is visited.

The action to perform the activity of **eat** is defined in Figure 4, whose parameters are the user `?pers` and the restaurant `?loc`. The duration of the action is defined by the function `(time_for_eat ?pers)` and specified by the user. To apply this action, the following conditions must hold: (1) the user is at `?loc` during the whole execution of the action; (2) `?loc` is open during the whole execution of the action; (3) the restaurant has a free table and (4) both the time window for the time to have lunch defined by the user and the available time are active. The effects of the action assert that the user has had lunch.

4 SIMULATOR

The objective of the simulator is to execute the plan and monitor that everything works as expected. To accomplish this, we first need to create the structures to perform the simulation. We use a timed event simulation, where events occur at particular times through a timeline, possibly provoking changes in the world state. During the plan monitoring, we check the predicates and functions and we visually show the plan trace in a specially designed GUI. In case a failure that prevents an action of the plan from being executed is found during the plan simulation, we activate a replanning mechanism that requires a knowledge-based reformulation of the new planning scenario. Next, we describe these tasks in more detail.

```
(:durative-action visit
  :parameters (?per - person ?mon - monument)
  :duration (= ?duration (visit_time ?mon ?per))
  :condition
    (and
      (at start (be ?per ?mon))
      (over all (be ?per ?mon))
      (over all (active ?per))
      (over all (open ?mon))
  :effect
    (and
      (at end (visited ?per ?mon))))
```

Figure 3. Action visit of the tourism domain

```
(:durative-action eat
  :parameters (?pers - person ?loc - restaurant)
  :duration  (= ?duration (eat_time ?pers ?loc))
  :condition
    (and
      (at start (free_table ?loc))
      (at start (be ?pers ?loc))
      (over all (be ?pers ?loc))
      (over all (active ?pers))
      (over all (open ?loc))
      (over all (time_for_eat ?pers)))
  :effect
    (and
      (at end (eaten ?pers))))
```

Figure 4. Action eat of the tourism domain

4.1 Timed event simulation: the timeline

A timeline is a simple structure that contains a collection of unique timed events in chronological order that represents a sequence of world states and that need to be monitored. The timeline is generated with the actions of the plan, the problem information and the live events, as depicted in Figure 1. A timed event is an event that happens at time t and contains the following information: (1) the *start*, *over all* or *end* conditions to be checked at t; (2) the *start* or *end* effects to be applied at t; (3) TILs that represent exogenous events but that are defined as part of the problem information, so they are known at the time of the plan simulation; and (4) *live events*, that dynamically appear during the executing/monitoring process and so they are unknown a priori. This way, a timeline encapsulates the information about the plan (irrespective of it is a sequential or parallel one), TILs and live events[7], and the corresponding world states. The time scale of the timeline will depend on the granularity of the plan and the periodic steps we want to use for monitoring the timed events. In our implementation, live events can be manually supplied or they can be retrieved from a datasource that keeps information about the real world.

Given a plan with two actions (`move` and `visit`), of duration 20 and 60, respectively, and a live event that indicates the

[7] The information about the plan, problem information and live events is modeled in PDDL format. We used PDDL4J (`https://github.com/pellierd/pddl4j`), an open source library that facilitates the development of JAVA tools for automated planning based on the PDDL language

```
0.00 move person1 hotel museum [20.00]
20.01 visit person1 museum [60.00]
90.00 live event (not (open museum1))
```

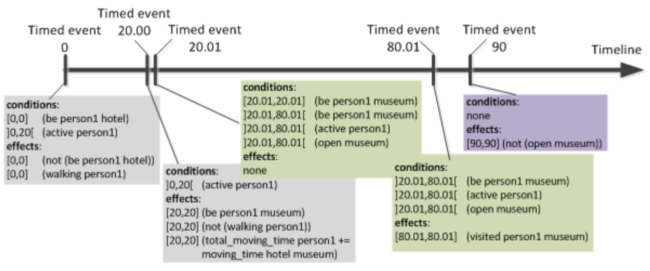
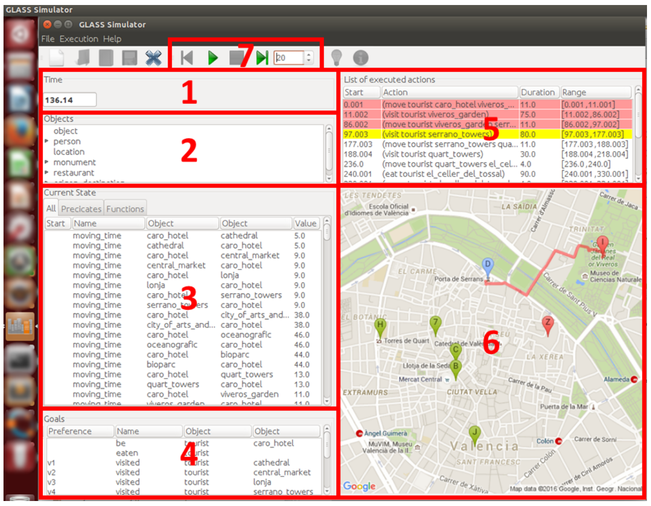

Timed event
0
Timed event
20.00
Timed event
20.01
Timed event
80.01
Timed event
90
Timeline
conditions:
[0,0] (be person1 hotel)
]0,20[(active person1)
effects:
[0,0] (not (be person1 hotel))
[0,0] (walking person1)
conditions:
]0,20[(active person1)
effects:
[20,20] (be person1 museum)
[20,20] (not (walking person1))
[20,20] (total_moving_time person1 += moving_time hotel museum)
conditions:
[20.01,20.01] (be person1 museum)
]20.01,80.01[(be person1 museum)
]20.01,80.01[(active person1)
]20.01,80.01[(open museum)
effects:
none
conditions:
]20.01,80.01[(be person1 museum)
]20.01,80.01[(active person1)
]20.01,80.01[(open museum)
effects:
[80.01,80.01] (visited person1 museum)
conditions:
none
effects:
[90,90] (not (open museum))

Figure 5. An example of a timeline with five timed events. Note the closed interval for the *at start/end* conditions and effects, and the open interval for the *over all* conditions



Time

136.14

Objects

object
person
location
monument
restaurant

Current State

All | Precicates | Functions

Start	Name	Object	Object	Value
	moving_time	caro_hotel	cathedral	5.0
	moving_time	cathedral	caro_hotel	5.0
	moving_time	caro_hotel	central_market	9.0
	moving_time	central_market	caro_hotel	9.0
	moving_time	caro_hotel	lonja	9.0
	moving_time	lonja	caro_hotel	9.0
	moving_time	caro_hotel	serrano_towers	9.0
	moving_time	serrano_towers	caro_hotel	9.0
	moving_time	caro_hotel	city_of_arts_and...	38.0
	moving_time	city_of_arts_and...	caro_hotel	38.0
	moving_time	caro_hotel	oceanografic	46.0
	moving_time	oceanografic	caro_hotel	46.0
	moving_time	caro_hotel	bioparc	44.0
	moving_time	bioparc	caro_hotel	44.0
	moving_time	caro_hotel	quart_towers	13.0
	moving_time	quart_towers	caro_hotel	13.0
	moving_time	caro_hotel	viveros_garden	11.0

Goals

Preference	Name	Object	Object
	be	tourist	caro_hotel
	eaten	tourist	
v1	visited	tourist	cathedral
v2	visited	tourist	central_market
v3	visited	tourist	lonja
v4	visited	tourist	serrano_towers

List of executed actions

Start	Action	Duration	Range
0.001	(move tourist caro_hotel viveros_...	11.0	[0.001 ,11.001]
11.002	(visit tourist viveros_garden)	75.0	[11.002 ,86.002]
86.002	(move tourist viveros_garden serr...	11.0	[86.002 ,97.002]
97.003	(visit tourist serrano_towers)	80.0	[97.003 ,177.003]
177.003	(move tourist serrano_towers qua...	11.0	[177.003 ,188.003]
188.004	(visit tourist quart_towers)	30.0	[188.004 ,218.004]
236.0	(move tourist quart_towers el_cel...	4.0	[236.0 ,240.0]
240.001	(eat tourist el_celler_del_tossal)	90.0	[240.001 ,330.001]

Figure 6. Simulator graphical user interface

museum is closed (not open) at time 90, the resulting timeline is shown in Figure 5.

4.2 Plan execution simulation

The simulation of the plan execution requires to set the size of the execution step that will be applied along the timeline explained in section 4.1. We can set the step size to the time granularity of the planner or choose a larger size. The smaller the size of the execution step, the more frequently access to external databases (Google APIs) to acquire new information and update the real-world state. Thus, the execution step size specified by the user determines the update frequency of the internal state of the simulator with respect to the real-world state. If changes frequently occur in the domain, a small execution step will result in a more reliable simulation with a proactive behavior. The simulation state is also updated at each timed event, checking the conditions of the actions in the state and applying the effects of the event. Additionally, the simulator also interacts with the real-world through live events, which may in turn modify or create new timed events in the timeline.

The simulation of the plan execution starts at time zero, with an initial state equal to the real-world state, and the simulator advances through the timeline in every execution step (see Figure 5). The simulator checks that conditions are satisfied, the current state matches the expected state, and then updates the current state accordingly — this whole process is visually shown in our GUI, described below. More specifically, every execution step involves two main tasks:

1. processing the live events for changes and update the respective timed events
2. for every unprocessed timed event within the current step: (1) update the simulation state with the TILs and effects of the live events; (2) check the conditions of the timed event to find differences between the current state and the expected state; and (3) update the state with the effects of the actions.

If a difference between the current state and the expected state is found and this difference leads to a situation where the plan is no longer executable, then a failure has been detected. In such a case, the GUI informs the user about the cause of the failure: the action that has failed and the conditions that have been violated. For instance, in the example of Figure 5, let us suppose there is a live event at time 60 that indicates the museum will no longer be open from 60 onwards. In this case, the *overall* condition `]20.01,80.01[ (open museum)` is violated, which means the `visit` action cannot be successfully executed. Then, we need to invoke the replanning module as described in Section 4.4

4.3 Graphical User Interface

The graphical user interface (GUI) has been designed to provide information about the internal state of the simulator during the whole plan execution simulation and provides mechanisms to control the next step of the simulation. The GUI is specifically designed to offer a smart-city orientation. It includes six distinguishable GUI parts:

1. Figure 6-section 1 shows the current simulation time
2. Problem objects (Figure 6-section 2): it displays the planning problem objects along with their types. This static information will not change over the simulation process.
3. Current state (Figure 6-section 3): This graphical section contains the PDDL description of the current state, which can change after an action starts or ends, when a live event arrives or when a user introduces a manual change (TILs). Propositions and numeric fluents of the current state can be separately consulted in two different tabs.
4. Figure 6-section 4 shows the problem goals. In later refinements, we intend to show the goals that are expectedly to be achieved with the plan under execution.
5. Figure 6-section 5 shows the dynamic list of plan actions, their start time, the objects involved in the action execution and the action duration. In addition, actions are shown with a representative colour: actions currently in execution are shadowed in yellow, past or already executed actions in red, and future actions in white.
6. Representative map (Figure 6-section 6): The map depicts with location icons the relevant places involved in the plan

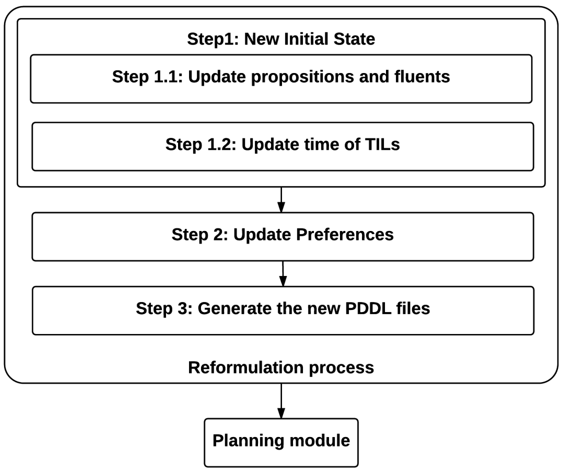

Step1: New Initial State
Step 1.1: Update propositions and fluents
Step 1.2: Update time of TILs
Step 2: Update Preferences
Step 3: Generate the new PDDL files
Reformulation process
Planning module

Figure 7. Reformulation steps

(places to visit, restaurant, hotel). These location icons change their colour when the corresponding action is executed. The map also displays distances between locations.

7. Simulation control buttons (Figure 6-section 7): In the middle of the top menu, the interface displays four buttons to run the simulator step by step (the step size is defined by the user), continue with the simulation, stop the simulation and reset the simulation.

4.4 Reformulating the planning problem

Figure 7 shows the steps of the reformulation procedure.

Step 1: Create the New Initial State. The initial state will comprise the information known by the simulator at the time of creating the new problem. This includes the information of the current simulation state plus the information about future TILs; that is currently known information about some future events. Thereby, the occurrence time of the future TILs must also be updated.

Step 1.1: Update propositions and fluents. This step refers to the update of the current state. The propositions and fluents after the failure are retrieved from the current world state. However, this might not be an accurate state since we do not have sensing actions that provide us with a precise picture of the real world. This may be particularly problematic when an *overall* or an *at end* condition of an action is violated and the action has *at end* effects. Let us take as an example the action `(move tourist caro_hotel viveros_garden)`, with an *at start* effect `(at start (not (be tourist caro_hotel)))`, an *at end* effect `(at end (be tourist viveros_garden))`, and an *overall* condition `(over all (active tourist))`. Due to a failure that resulted from a live event which violated the previously mentioned *overall* condition, the tourist is neither in `caro_hotel` because the *at start* effect were already executed, nor in `viveros_garden` because the *at end* effects were not yet executed due to the failure. For simplicity, and because of the lack of sensing actions in our current implementation, when a failure happens due to an *overall* or an *at end* condition violation, we will calculate the new initial state by simply rolling back the *at start* effects of the failing action (if any).

Step 1.2: Update the time of TILs. When the new problem is reformulated, we invoke the OPTIC planner, which resets the time of execution and generates a plan starting from time equal to zero. Consequently, we need to update the occurrence time of the TILs to the result of its original time minus the failure time. Let us assume that a failure occurred at time 100, and that we have the TIL planned at time 235 `(at 235 (not (open la_paella)))`, meaning that the restaurant *la paella* will close at 235. Therefore, in the new initial state formulation, its time will be 135 (235 minus 100); and it will be updated to `(at 135 (not (open la_paella)))`.

Step 2: Update preferences. When a failure occurs, we come across a situation where we can distinguish two types of preferences or soft goals:

1. Goals that have already been achieved at the time of the failure by the actions that have been successfully executed before the failure. These preference goals along with their penalties will not be included in the new reformulated problem.
2. Goals that have not been satisfied, and which can in turn be divided into two sets:
   (a) The problem goals that were not included in the original plan;
   (b) The problem goals included in the original plan that have not been satisfied yet due to the failure.
   - For the set of goals in (a), the penalties are kept intact as they were originally defined in the problem file.
   - For the set of goals in (b), we want to keep the plan stability metric [4] similar to the concept of minimal perturbation [11], which is why we increase the penalties of these goals in the new reformulated problem. Particularly, we opt for assigning a relatively high priority to these pending goals (twice as much as the maximum penalty among all goals), in order to potentially enforce these goals in the new plan. We have thus opted for applying a *stability strategy* that gives more priority to goals that were already included in the original plan than goals that were not. Other strategies such as keeping a higher level of stability with respect to the failed prior plan can also be adopted. In the case of a tourism domain, we think that maintaining the original agenda of the tourist as far as possible is more advisable.

     For example, let us consider that the preference `(preference v2 (sometime (visited tourist central_market)))` is one of the soft goals in the set (b); this preference indicates that sometime during the execution of the plan the tourist wishes to visit the `central_market`. Assuming that the penalty of this preference in the original problem file was 270, and that the highest among all preferences was 300, the new penalty for preference `v2` will be 600.

Finally, two points are worth mentioning. First, we learn the soft goals that the planner decided to pursue in the original plan by simply executing the plan without any live events. Second, the pending hard goals of the original problem are kept as hard goals in the new problem file.

Step 3: Generate the new PDDL files. The last step of the reformulation process consists in generating the new PDDL files. In principle, the domain file remains unchanged, unless we wish to necessarily include some particular action in the new plan. In this case, we would need to encode a *dummy effect* that triggers the corresponding action. Otherwise, only the problem file is generated taking into consideration the modifications discussed in step 1 and step 2.

In future implementations of the simulator, we will test a Constraint Programming approach [6] for reformulating the planning problem, as in [15], and compare the performance when relying on a scheduler rather than a planner.

5 CASES OF STUDY

The aim of this section is to show the behaviour of our simulation system with a representative example. We have a tourist who wishes to make a one-day tour in the city of Valencia. Initially, the system retrieves a set of recommended places according to the user profile (table 1, column 1) and a set of restaurants. The list of recommended places is calculated by a Recommender System through the user profile. This list of places comes along with a recommendation value (Table 1, column RV) according to the interest degree of the user in the particular place. This value will be used by the planning module to obtain a plan that fits the user's likes.

The tour (plan) for the user calculated by the planner is shown in the left snapshot of Figure 8. The visits included in the plan are marked with a red location icon in the snapshot. The tour starts from the origin location of the tourist, i.e., the hotel in which the user is staying at (green location icon), and includes six visits to monuments (red icons) and one stop at a restaurant (orange icon).

The simulator starts the plan execution simulation with the information provided above. Let us assume that at time 1:55 pm, a live event is received, `(at 235 (not (free_table el_celler_del_tossal)))`, indicating that the restaurant chosen by the planner, *el celler del tossal*, is completely full and has no available table. At the time the live event arrives, the tourist has already visited the first three monuments (1. Viveros garden; 2. Serrano towers; 3. Quart towers), and he is currently at the location of the restaurant *el celler del tossal*. When the user learns the restaurant is fully booked, the simulator detects a failure because the action `(eat tourist el_celler_del_tossal)` is not executable. Then, the simulator reformulates a new planning problem in order to obtain a plan that solves the failure:

1. *Initial state*: the current location of the tourist is the point at which the previous plan failed; i.e., the restaurant *el celler del tossal*. Since the new initial state is initialized to time zero ($t_{ini} = 0$), the simulator updates the time of the TILs in the current state, namely, the opening and closing time of places, the time slot for having lunch and the TIL `(at` $t_{available}$ `(not (active tourist)))`, where $t_{available}$ is set to the new time the tourist must get back to the hotel from $t_{ini} = 0$. Additionally, the fluent `(total_moving_time tourist)` is updated with the total time the tourist has spent in moves around the city.
2. *Goals*: the places that have already been visited (the first three monuments) are removed from the goal list. The new set of goals includes two lists: (a) the *pending goals* of the failed plan; that is, have lunch (4. have lunch) and the three remaining monuments that have not been yet visited by the user (5. Lonja, 6. Central market, 7. Town Hall); plus (b) the goals of the original problem goals that were not included in the first plan.

Regarding penalties of the goals, the list of goals in (b) are included in the new planning problem with their original recommended values (see the non-bold values RV' in column 2 of Table 1). As for the pending goals of (a), the penalty of these goals is increased with respect to their penalty in the first plan (see the bold values RV' in column 2 of Table 1 for the three pending monuments) accordingly to the stability concept explained in section 4.4.

The simulator invokes OPTIC and obtains a new plan, displayed in the middle snapshot of Figure 8. A few things must be noted in this new plan:

1. OPTIC suggests a new restaurant (orange icon labeled with number 4) which is rather far away from the prior restaurant. The reason is that we have only included in the planning problem the 10-top restaurants in Valencia suggested by *Trip Advisor*, and the closest one to the prior restaurant is the one shown in the second map.
2. The places included in the new plan are marked with green location icons as well as the paths between places. We can observe that the new plan maintains the visit to Town Hall (now represented with the green icon numbered as 5) and to the Lonja (now indicated with the green icon with number 6). However, the visit to Central Market has been discarded in this new plan. This is likely due to the longer distance to the new restaurant.

The simulation continues. Let us assume that when the tourist is visiting the Town Hall, a new live event announcing the building closes before the scheduled closing time (`(at 140 (not (open town_hall)))`) is received, the current time being equal to 140 . A new failure is detected in the middle of the execution of `(visit tourist town_hall)` due to a violation of an *overall* condition. As we explained in section 4.4, the simulator applies a rollback process to obtain the current state before the last visit action but preserving the simulation time. Then, in the new reformulated problem, the user is located at the Town Hall, he has not visited the Town Hall and the live event causes the proposition `(open town_hall)` to be removed from the initial state. Note that the goal `(visited tourist town_hall)` will be included as a goal in the new problem to maintain the plan stability but since the Town Hall is no longer open for visits, the planner will not include this goal in the new plan. The penalties of the goals for this third problem are shown in column 3 of Table 1.

The third plan (right snapshot of Figure 8), shown in light blue colour, retrieves the visit to the Central market that was eliminated from the second plan. The new plan suggests visiting La Lonja (5. la Lonja) and then the Central market (6. Central market). Finally, the user returns to the hotel.

6 CONCLUSIONS AND FURTHER WORK

We have presented a context-aware smart tourism simulator that keeps track of the execution of a tourist agenda. The sim-

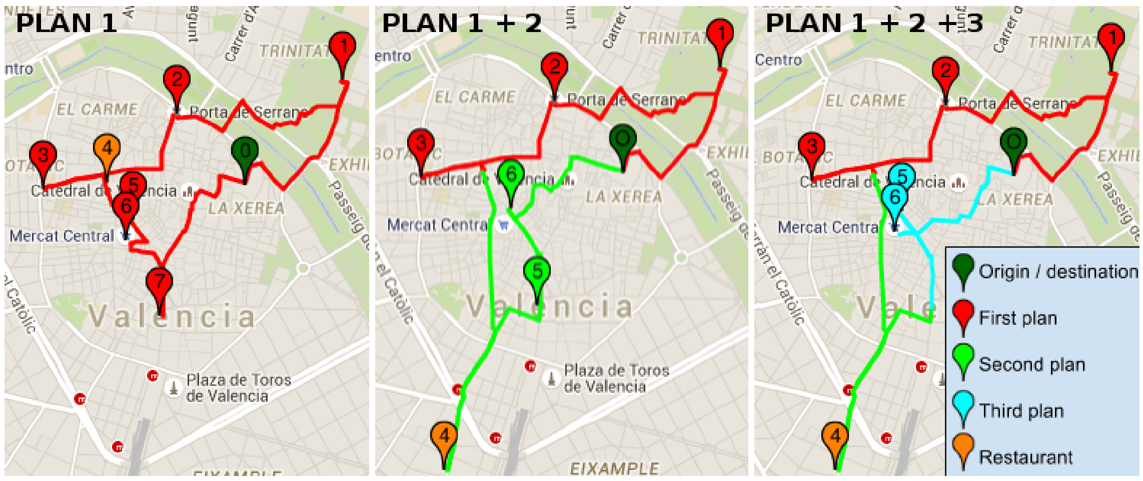


Figure 8. The three simulated plans. Icons in PLAN1: (0) Caro hotel, (1)Viveros Garden, (2) Serrano towers, (3) Quart towers, (4) El Celler del Tossal (RESTAURANT), (5)Lonja, (6) Central market, (7) Town hall. Icons in PLAN2: 1,2,3 are the same as PLAN1, (4) the Pederniz (RESTAURANT), (5) Town hall, (6)Lonja. Icons in PLAN3: 1,2,3,4 are the same as PLAN2, (5) Lonja, (6) Central market

PLACES	**RV**	**RV'**	**RV"**
Cathedral	280	280	280
Central market	270	**600**	**600**
Lonja	290	**600**	**600**
Serrano towers	250	—	—
City of arts and sciences	280	280	280
Oceanografic	300	300	300
Bioparc	210	210	210
Quart towers	200	—	—
Viveros garden	250	—	—
Town hall	200	**600**	**600**

Table 1. Recommended places

ulator periodically updates its internal state with real-world information and receives sensible environmental changes in the form of live events. Events are processed in the context of the plan and in case of failure, a new planning problem is formulated. This involves creating the new initial state and updating the time of timed events. In the case of study, we have shown how the user can track the plan execution through a GUI that automatically displays the plan under execution.

As for future work, we intend to endow the system with a pro-active behaviour, analyzing the incoming of live events that entail a future failure in the plan.

REFERENCES

[1] J. Benton, A. J. Coles, and A.I. Coles, 'Temporal planning with preferences and time-dependent continuous costs', in *ICAPS*, (2012).
[2] D. Buhalis and A. Amaranggana, 'Smart tourism destinations enhancing tourism experience through personalisation of services', in *Proc. Int. Confernece on Information and Communication Technologies in Tourism*, pp. 553–564, (2013).
[3] S. Edelkamp and J. Hoffmann, 'PDDL2.2: The language for the classical part of the 4th International Planning Competition', *IPC 04*, (2004).
[4] M. Fox, A. Gerevini, D. Long, and I. Serina, 'Plan stability: Replanning versus plan repair', in *ICAPS*, volume 6, pp. 212–221, (2006).
[5] I. Garcia, L. Sebastia, E. Onaindia, and C. Guzman, '*E-Tourism*: a tourist recommendation and planning application', *International Journal on Artificial Intelligence Tools*, **18**(5), 717–738, (2009).
[6] A. Garrido, M. Arangu, and Eva. Onaindia, 'A constraint programming formulation for planning: from plan scheduling to plan generation', *Journal of Scheduling*, **12**(3), 227–256, (2009).
[7] A. Gerevini, P. Haslum, D. Long, A. Saetti, and Y. Dimopoulos, 'Deterministic planning in the 5th International Planning Competition: PDDL3 and experimental evaluation of the planners', *Artificial Intelligence*, **173**(5-6), 619–668, (2009).
[8] U. Gretzel, M. Sigala, Z. Xiang, and C. Koo, 'Smart tourism: foundations and developments', *Electronic Markets*, **25**(3), 179–188, (2015).
[9] R. Howey, D. Long, and M. Fox, 'VAL: automatic plan validation, continuous effects and mixed initiative planning using PDDL', in *16th IEEE ICTAI*, pp. 294–301, (2004).
[10] J. Ibañez, L. Sebastia, and E. Onaindia, 'Planning tourist agendas for different travel styles', in *ECAI*, (2016).
[11] S. Kambhampati, 'Mapping and retrieval during plan reuse: A validation structure based approach.', in *AAAI*, pp. 170–175, (1990).
[12] F. Martínez-Santiago, F. J. Ariza-López, A. Montejo-Ráez, and A. U. López, 'Geoasis: A knowledge-based geo-referenced tourist assistant', *Expert Syst. Appl.*, **39**(14), 11737–11745, (2012).
[13] I. Mínguez, D. Berrueta, and L. Polo, 'Cruzar: An application of semantic matchmaking to e-tourism', *Cases on Semantic Interoperability for Information Systems Integration: Practices and Applications. Information Science Reference*, (2009).
[14] B. Neuhofer, D. Buhalis, and A. Ladkin, 'Smart technologies for personalized experiences: a case study in the hospitality domain', *Electronic Markets*, **25**(3), 243–254, (2015).
[15] I. Refanidis, C. Emmanouilidis, I. Sakellariou, A. Alexiadis, R.-A. Koutsiamanis, K. Agnantis, A. Tasidou, F. Kokkoras, and P. S. Efraimidis, 'myVisitPlanner ^{gr}: Personalized Itinerary Planning System for Tourism', in *SETN*, pp. 615–629. Springer, (2014).
[16] P. Vansteenwegen, W. Souffriau, G. V. Berghe, and D. V. Oudheusden, 'The city trip planner: An expert system for tourists', *Expert Syst. Appl.*, **38**(6), 6540–6546, (2011).

Extending naive Bayes with precision-tunable feature variables for resource-efficient sensor fusion

Laura Isabel Galindez Olascoaga[1] and **Wannes Meert**[2] and **Herman Bruyninckx**[3] and **Marian Verhelst**[1]

Abstract. Resource-constrained ubiquitous sensing devices suffer from the fundamental conflict between their limited hardware resources and the desire to continuously process all incoming sensory data. The data's representation quality has an immediate impact on both aspects. This paper strives to enable resource-aware and resource-tunable inference systems, which are capable of operating in various trade-off points between inference accuracy and resource usage. We present an extension to naive Bayes that is capable of dynamically tuning feature precision in function of incoming data quality, difficulty of the task and resource availability. We also develop the heuristics that optimize this tunability. We demonstrate how this enables much finer granularity in the resource versus inference accuracy trade-off space, resulting in significant resource efficiency improvements in embedded sensor fusion tasks.

1 INTRODUCTION

The Internet of Things (IoT) paradigm is on the rise, promising an important contribution to tackling societal challenges through improved distributed sensing capabilities. This paradigm is expected to significantly impact application scenarios like e-health and domotics, which already benefit from the widespread availability of embedded sensing devices (i.e., smartphones, activity trackers and service robots) that can reliably gather and process a massive amount of data. The main enabling factor of this vision is the ability to seamlessly integrate several technological solutions which motivates the desire to run complex inference tasks in-situ and in a distributed manner. Yet, smart embedded devices' processing abilities are held back by their limited resource availability, both in terms of energetic as well as computational resources [2]. This creates a fundamental conflict between the desire to fuse information from more and more always-on sensors in embedded devices, and the inability of these embedded devices to process all incoming data continuously and at high precision. This conflict is currently circumvented by running most sensor fusion and sensory inference tasks in the cloud [1, 4]. Yet, this has important consequences towards the system's latency and the user's privacy [5]. Moreover, it does not solve the excessive power consumption spent by the always-on sensors and the wireless link [8].

Efficiently running inference tasks on the devices themselves calls for awareness of the real-time embedded platform's resource limitations. This is in sharp contrast with most state-of-the-art inference approaches, which focus on maximizing information gain and inference accuracy [4] without taking the actual hardware footprint of online inference into account. To facilitate effective sensory fusion inference tasks inside embedded devices, this paper strives to enable resource-aware and resource-tunable inference systems, which are capable of operating in various trade-off points between inference accuracy and resource usage. Such performance-tunability can be realized by dynamically reallocating resources across sensory features in accordance to the task relevance and complexity. Recent techniques, such as feature-cost aware inference [4, 7], perform hardware-cost aware feature selection to minimize the overall resource cost of feature extraction. Additionally to feature-cost one can also adapt known machine learning models such that they are efficiently run on embedded systems by preferring integer operators [25], considering the trade-off between number of operations and accuracy [19], reducing the precision [29] and the value range [9] of the features , or decomposing the model and distributively running the inference task on different processing units [14, 16, 10]. In addition, recent efforts have attempted to integrate such machine learning models and techniques under embedded hardware efficient frameworks [14]. These optimization techniques result in a fixed resource usage versus performance operating trade-off and, as a result, fail to exploit all the resource saving opportunities that the hardware platform can provide. To overcome these limitations, this paper introduces the following innovations:

1. *Feature precision-tunability:* Instead of only selecting or deselecting a sensory feature, embedded platforms can also tune the precision of sensors and sensory feature extraction (i.e. by changing their resolution or number of bits) in return for hardware resource savings through techniques such as approximate computing and approximate sensing. This allows them to dynamically trade-off feature quality for resource efficiency. In this paper, we will extend the naive Bayes fusion model to such feature-precision tunability.
2. *Run-time accuracy-resource-awareness:* Instead of offline feature or precision selection, a dynamic approach should allow run-time accuracy-resource tunability. This requires the creation of a single fusion model, capturing all feature precision-tunability states, which can be explored and traversed at run-time. The resulting model enables run-time adaptations of feature precision in function of incoming data quality, in function of the difficulty of the inference task, or in function of the instantaneous resource availability in the embedded system.

We present an extension to naive Bayes that is capable of dynamic feature precision tunability, as well as heuristics to optimize this tunability. We demonstrate how this enables much finer granularity in

[1] MICAS – Department of Electrical engineering, KU Leuven, email: Laura.Galindez@esat.kuleuven.be
[2] DTAI – Department of Computer Science, KU Leuven
[3] PMA – Department of Mechanical Engineering, KU Leuven

[4] In this paper we refer to inference accuracy as the percentage of correctly predicted queries from a test set.

the resource versus inference accuracy trade-off space, resulting in significant resource efficiency improvements in embedded sensor fusion tasks. These performance tuning capabilities are of up most relevance in data-dense and resource-constricted environments like the IoT. Therefore, we demonstrate the functionality of our proposed techniques in sensor fusion datasets related to applications relevant to this paradigm.

The paper is structured as follows. In Section 2 we give a theoretical background of naive Bayes classifiers and Bayesian Networks (BN) and we explain how precision tuning enables resource-efficiency in the current context. Section 3 gives the details of the proposed feature precision-tunable BN and explains how parameter learning and inference are performed in it. In Section 4 we propose a heuristic that uses the proposed BN to select optimal operating points in the resource versus inference accuracy space and we evaluate the trade-off it achieves by performing experiments on four data corpora in Section 5. Finally, we discuss the results as well as our contributions and future work in Section 6.

2 BACKGROUND

2.1 Bayesian Networks

Bayesian Networks (BN) are directed acyclic graphs that compactly encode a joint probability distribution over a set of random variables [24]. Consider a set of random variables $\mathbf{U} = \{F_1, \ldots, F_n\}$ where each feature F_i may take values from a finite set $Val(F_i)$. A Bayesian Network for a set of random variables $\mathbf{U}$ is formally defined as the pair $B =< G, \Theta >$. The first component G represents the graph which encodes conditional independence assumptions. The nodes represent variables F_i and its arcs represent the probabilistic dependencies between variables. The second component Θ represents the set of conditional probability distributions $\Pr(F_i|\Pi_{F_i})$ that quantify the network where Π_{F_i} denotes the parents of F_i.

The joint probability distribution defined by the network B is given by

$$Pr(F_1, ..., F_n) = \prod_{i=1}^{n} Pr(F_i|\Pi_{F_i}). \tag{1}$$

Inference in BNs, $\Pr(\mathbf{Q}|\mathbf{E} = \mathbf{e})$, can be performed by assigning values $\mathbf{e}$ to variables $\mathbf{E}$ that are observed and by summing out variables $\mathbf{U}\backslash(\mathbf{Q} \cup \mathbf{E})$ that are not part of the query $\mathbf{Q}$.

2.2 Bayesian Network Classifiers

Classification is the task of assigning a class label C to instances described by a set of features $F_1, ..., F_n$. Such a task can be tackled by a Bayesian network where one of the random variables, C, is considered the class and the other random variables, $F_1, \ldots, F_n$, represent the features. The task is now to find the most likely value for the class variable C:

$$c = \arg\max_c \Pr(C = c|F_1 = f_1, \ldots, F_n = f_n), \tag{2}$$

where c is the current class and f_i is the observed value for feature F_i.

A widely used type of Bayesian classifier is the naive Bayes classifier [12]. The main assumption is that every feature is independent of the other features given that the class is known. The graphical structure of the naive Bayes network is shown in Figure 1. This assumption allows for efficient learning and inference as it simplifies Equation 2 to

$$c = \max_c \Pr(F_1 = f_1|C = c) \ldots \Pr(F_n = f_n|C = c) \Pr(C = c)$$

by applying the rule of Bayes and conditional independence. Despite the independence assumption, naive Bayes classifiers perform surprisingly good. This makes them one of the most effective and efficient inductive learning algorithms [33, 6].

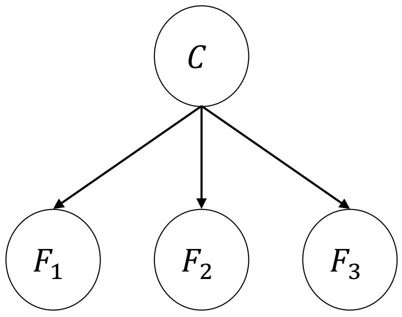


Figure 1. Graphical representation of the naive Bayes classifier

2.3 Resource awareness and precision tuning

Embedded hardware platforms have to operate under very scarce resources. First and foremost, their miniaturization results in very limited battery capabilities which motivates the quest for high energy efficient designs and methodologies. Moreover, due to size, cooling and cost restrictions, the computational bandwidth of these devices is extremely scarce. This has sparked an enormous amount of research into adaptive hardware over the last decade. Under this paradigm, resource consumption can be tuned at run-time to be lower at the expense of reduced quality sensor streams or computations. This dynamic trade-off is achievable in several ways:

1. *Noisy sensors:* The amount of noise present in sensory measurement strongly depends on the amount of energy spent in the sensor front-end (in its filters, amplifiers, etc). By tolerating more statistical noise on the measurement result, energy can be saved [15, 3].
2. *Stochastic computing*: The resulting accuracy of digital computations can be traded off against processing resource usage and energy consumption by using stochastic or approximate computing techniques. In stochastic computing, for example, numbers are represented by bit-streams that can be processed by very simple circuits such as standard logic gate arrays. These implementations allow a limited amount of errors (stochastic noise) in the digital embedded calculations in return for a more efficient implementation [21].
3. *Reduced precision sensing and computing:* Instead of applying aforementioned stochastic techniques to dynamically trade feature quality for resource savings, significant resource savings are also achievable by simply limiting the amount of bits with which sensor values are sampled and digitally processed for feature extraction. Standard hardware platforms digitize sensory values at fixed precision and typically process them with 16-bit resolution. Recent works present the development precision-tunable digitizers, as well as digital processing platforms capable of dynamically

adjusting the precision with which internal computations are performed [23, 20].

Within this paper, we focus on enabling the feature quality versus resource trade-off through the latter technique: computations with variable precision features. The results are transferable to the discussed alternatives, which is left for future work. Under variable precision computations, the extracted features $\mathbf{U} = \{F_1, ..., F_n\}$ are each computed and represented by a tunable amount of bits. The number of bits representing the feature, directly impacts the set of values a feature F_i can take on and will be referred to as $F_{i,m}$ with m the number of bits. More specifically, when using a m-bit representation, the feature can take $|\,Val(F_{i,m})| = 2^m$ possible values.

As we are interested in studying resource efficiency in sensory applications we construct these m-bit representations by following a signal processing quantization approach [27]. Mapping of the original feature to an m-bit representation is based on comparisons with decision levels t_k. If the feature value is between t_k and t_{k+1} it gets mapped to a quantization level l_k, where the number of levels must be equal to 2^m. In this paper, the decision levels t_k are derived from the feature value range and the set of lower precision decision levels is a subset of the higher precision decision levels (see also Section 3).

State-of-the-art implementations show that under such computational precision tuning, the resource cost (here expressed in terms of energy per computational operation) scales more than quadratically with the computational precision, expressed in terms of number of bits [22]. In this paper, we will assume that the feature cost (denoted $T_{i,m}$) of feature F_i computed with precision m-bits is equal to

$$T_{i,m} = \alpha_i \cdot m^2. \tag{3}$$

where α_i is the feature-dependent cost of the nominal precision feature.

3 VARIABLE FEATURE PRECISION NAIVE BAYES

Tuning feature precision enables a wide range of resource dependent operation points.We make the probabilistic relations required by classification tasks explicit in a Bayesian Network (BN) where features can be observed at different precision levels.

3.1 Model Structure

The proposed BN represents a naive Bayes classifier that has multiple versions of the same feature in each leaf, as shown by Figure 2.

Feature versions with the highest precision ($F_{i,m}$) are directly linked to the class variable C. There are no direct links between lower precision feature versions $F_i \setminus F_{i,m}$ and the class variable since the relation between these versions is deterministic. The proposed structure encodes the following joint probability distribution over the multiple feature version sets $F_i = \{F_{i,m}, \ldots, F_{i,0}\}$ and the class variable C

$$Pr(C, F_{1,m}, ..., F_{n,0}) = \\ \prod_{i=1}^{n} \prod_{b=0}^{m-1} Pr(F_{i,b}|F_{i,b+1}) \cdot Pr(F_{i,m}|C) \cdot Pr(C). \tag{4}$$

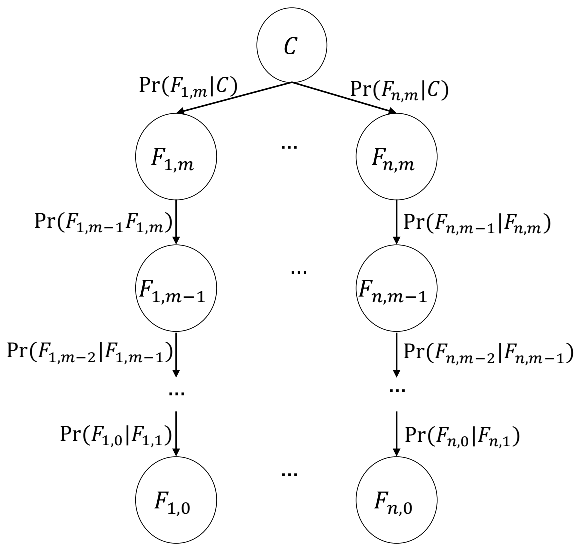


Figure 2. Naive Bayes model extended with multiple feature quality versions

3.2 Parameter Learning

We assume that the studied features were generated by a continuous distribution [24], therefore, we model the conditional probabilities between features of highest precision and classes $Pr(F_{i,m}|C)$ as Gaussian distributions [18]. Even though the model includes $n \times (m + 1)$ parameters θ, only the conditional probabilities between features of highest precision and classes $Pr(F_{i,m}|C)$, must be trained since the conditional probabilities between lower precision features $Pr(F_{i,b}|F_{i,b+1})$ are deterministic. Once we have knowledge of the decision levels t_k that generated the lower precision features, we are able to systematically add them to the tails of the proposed naive Bayes structure.

3.3 Inference

At any given time, every feature F_i is observed at only one of the precision options b_i depending on the current resource consumption desires and constraints. Given observation $o = \{f_{1,b_1}, f_{2,b_2}, ..., f_{n,b_n}\}$, classification is performed by estimating the class posterior probability given by

$$Pr(C|o) \sim \prod_{i=1}^{n} Pr(fi, b_i|C) \cdot Pr(C). \tag{5}$$

This implies that, for every observed feature, its lower precision versions are not observed, while their higher precision versions are marginalized.

Consider the example depicted in Figure 3 which may correspond to a robot navigation application, as discussed in Section 5.2. Suppose we obtain sensor readings at 8 bit, 4 bit and 2 bit for sensors $S1$, $S2$ and $S3$, respectively and we decide to turn off sensor $S4$. Here, we can estimate the class posterior probability (which can be a location, for example) with the following equation

$$Pr(C|S_{1,8b}, S_{2,4b}, S_{3,2b}) \sim Pr(S_{1,8b}|C) \cdot \sum_{S_{2,8b}} Pr(S_{2,4b}|S_{2,8b})Pr(S_{2,8b}|C) \cdot \sum_{S_{3,4b}} \sum_{S_{3,8b}} Pr(S_{3,2b}|S_{3,4b}) \cdot Pr(S_{3,4b}|S_{3,8b}) \cdot Pr(S_{3,8b}|C) \cdot Pr(C), \quad (6)$$

and predict the class $c \in C$ with the highest posterior.

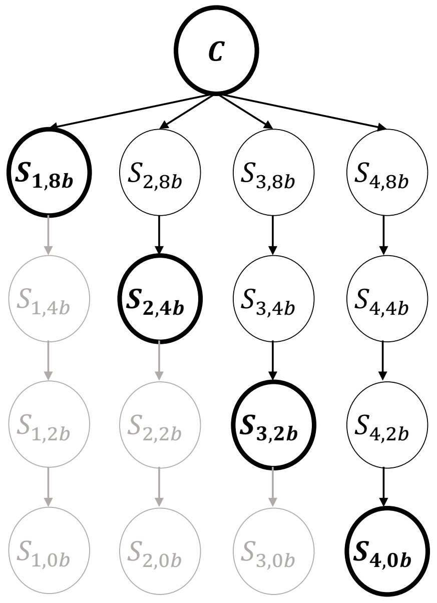


Figure 3. Example of a four feature application where each of them is observed at a different precision (circles with bold edges). Features with higher precision than the observed are marginalized (circles with black edges) and versions with lower precision are not observed (circles with gray edges). Note that feature 4 is observed with "0 bit" precision, which is equivalent to pruning it.

4 ACCURACY-RESOURCE TUNABILITY

The proposed model enables multiple resource and accuracy dependent operating points. In this paper we analyze the trade-off induced by the available feature combination choices and we propose a methodology to find the optimal operating points given the system's resource constraints.

We propose an accuracy-resource sensitive algorithm that selects the optimal feature precision across the accuracy-resource usage trade-off space. At each iteration, we select the feature set that optimizes a cost function CF, which is defined according to the desired application and the constraints thereto [13, 17]. In this paper we maximize the cost function given by

$$CF = \log\left(\frac{\Delta resource}{\max(resource)}\right) - \log(\Delta accuracy), \quad (7)$$

where the term Δ refers to the predicted state difference between time k and time k+1 as will be detailed in Algorithm 1. The greedy neighborhood search in our heuristic ensures resource reduction, and the cost function further motivates it by explicitly trading off the two terms.

Two of this algorithm's aspects distinguish it from conventional state-of-the-art feature selection techniques [7, 28, 31]: 1) We merge accuracy gain and resource usage in a joint cost optimization, hence taking hardware implementation aspects into account from the algorithmic level. 2) In contrast to cost-aware feature selection techniques which decide whether to use a feature or not, we enable the selection of a variety of feature precision combinations.

Algorithm 1 details the method. We initialize the selected feature set to the highest precision feature combination $\mathbf{U}_{selected} = \{F_{1,m}, ..., F_{n,m}\}$. At each iteration, we perform a greedy neighborhood search over n feature combination candidates. In each candidate i, the precision of feature F_i is dropped one level with respect to the current precision. We evaluate the classification accuracy and resource usage of each candidate and select the one that maximizes the cost function CF. The procedure is repeated until the feature combination with the lowest precision is selected ($\mathbf{U}_{selected} = \{F_{1,0}, ..., F_{n,0}\}$). Note that the algorithm is able to perform feature pruning if a "null precision" leaf is added to the Naive Bayes model (see Figure 3 for an example).

Classification accuracy is computed by estimating the posterior probability $Pr(C|k)$ of every instance k from a testing data-set $\mathbf{U}_{test}$ and comparing the prediction to to the instance's label (see Algorithm 2).

5 EXPERIMENTS

We evaluate the resource-accuracy trade-off achieved by our proposal with one synthetic dataset and three data corpora from two real applications relevant to the IoT paradigm.

5.1 Synthetic Data

This dataset consists of 2000 points sampled from 4 Gaussians $\mathcal{N}(\mathbf{m}_i, \boldsymbol{\sigma}_i)$, $i = \{1, 2, 3, 4\}$, where $\mathbf{m}_1 = \begin{pmatrix} -1.66 \\ -0.33 \\ -0.33 \\ -2.00 \end{pmatrix}$, $\mathbf{m}_2 = \begin{pmatrix} 1.00 \\ 0.5 \\ 1.00 \\ 1.00 \end{pmatrix}$, $\mathbf{m}_3 = \begin{pmatrix} 3.33 \\ 2.00 \\ 0.5 \\ 0.5 \end{pmatrix}$, $\mathbf{m}_4 = \begin{pmatrix} -1.66 \\ -1.43 \\ -0.66 \\ -3.33 \end{pmatrix}$, $\boldsymbol{\sigma}_1 = \begin{pmatrix} 0.80 \\ 1.00 \\ 1.00 \\ 1.00 \end{pmatrix}$, $\boldsymbol{\sigma}_2 = \begin{pmatrix} 0.70 \\ 1.00 \\ 1.00 \\ 1.00 \end{pmatrix}$, $\boldsymbol{\sigma}_3 = \begin{pmatrix} 1.00 \\ 1.00 \\ 1.00 \\ 1.00 \end{pmatrix}$ and $\boldsymbol{\sigma}_4 = \begin{pmatrix} 1.00 \\ 1.00 \\ 1.00 \\ 1.00 \end{pmatrix}$. The Gaussians are defined to have different degrees of overlap in every dimension and have therefore a varying miss-classification risk for different feature combinations. We quantize the data-set at 5, 3, 2 and 1 bits and randomly divide it into a training and a testing set (used for model training and accuracy estimation, respectively). We compute the resource usage with Equation 3, as included in Table 1 . To assign the variable α_i, we assume that features that are less likely to cause miss-classification would be more expensive to extract and compute in a real application. Thus giving a higher value to them. We add a "null precision" leaf, to enable feature pruning as shown in the example depicted by Figure 3.

Figure 4 shows the resource vs accuracy trade-off curve achieved by the proposed algorithm and achieved by a typical resource-aware heuristic [5] in red and blue, respectively. The gray point cloud represents all the possible accuracy-resource trade-off operational points to select from. The proposed heuristic has a richer feature combination space to select from, which prevents accuracy degradation for a

[5] The typical resource-aware heuristic considers only features at the highest precision $F_{i,m}$ and decides whether to prune them by maximizing CF.

Algorithm 1: Feature precision selection algorithm for accuracy-resource tradeoff

```
1  Feature precision selection ($\mathbf{U}_{test}, T_{i,m}$ , $\Theta$, $C$, $CF$);
   Input  : $\mathbf{U}_{test}, T_{i,m}$ , $\Theta$, $C$, $CF$
   Output: Selected feature set $\mathbf{U}_{selected_k} = \{F_{1,b_1}, ..., F_{n,b_n}\}$
2  k=0;
   /* Initialize with the highest precision feature set */
3  $\mathbf{U}_{selected_k} = \{F_{1,m}, ..., F_{n,m}\}$
4  $accuracy_k \leftarrow$ AccuracyEvaluation($\Theta$,$C$, $\mathbf{U}_{selected_k}$)
5  $resource_k \leftarrow T_{i,b_i}$ $\forall$ $F_{i,b_i} \in \mathbf{U}_{selected_k}$
   /* while the lowest feature precision has not been selected */
6  while $U_{selected_k} \neq \{F_{1,0}, ..., F_{n,0}\}$
7  do
8      for $i = 1$ to $n$ // For each candidate combination
9      do
           /* drop $F_i$'s precision one level */
10         $\mathbf{U}_{candidate_i} \leftarrow \mathbf{U}_{selected_k} \setminus F_{i,b_i} \vee \{F_{i,b_i-1}\}$
11         $accuracy_{candidate_i} \leftarrow$ AccuracyEvaluation($\Theta$, $C$, $\mathbf{U}_{candidate_i}$);
12         $resource_{candidate_i} \leftarrow T_{i,b_{canditate_i}}$ $\forall$ $F_{i,b_{candidate_i}} \in \mathbf{U}_{candidate_i}$;
13         $\Delta accuracy_{candidate_i} = accuracy_k - accuracy_{candidate_i}$;
14         $\Delta resource_{candidate_i} = resource_k - resource_{candidate_i}$;
15     end
16     update k=k+1;
17     $\mathbf{U}_{selected_k} \leftarrow \underset{\mathbf{U} \in \mathbf{U}_{candidate}}{\text{argmin}}$ CF($\Delta accuracy_{candidate}$,$\Delta resource_{candidate}$);
18     update $accuracy_k \leftarrow$ AccuracyEvaluation($\Theta$,$C$, $\mathbf{U}_{selected_k}$)
19     update $resource_k \leftarrow T_{i,b_i}$ $\forall$ $F_{i,b_i} \in \mathbf{U}_{selected_k}$
       Return: $\mathbf{U}_{selected_k}$
20 end
```

Algorithm 2: Classification accuracy evaluation algorithm

```
1  AccuracyEvaluation ($\Theta$,$C$,$\mathbf{U}$);
   Input  : $\Theta$,$C$,$\mathbf{U}$
   Output: accuracy
2  $correct = 0$;
3  for $k = 1$ to $N$ // With $N$ the number of instances in the testing set
4  do
       /* For each class we approximate the posterior probability given the
          instance currently analyzed */
5      $Pr(C|k) \leftarrow Pr(k|C) \cdot Pr(C)$
       /* We predict the class with the highest posterior probability */
6      $c_{max_k} = \underset{c \in C}{\text{argmax}} Pr(C|k)$
7      if $c_{max_k} == c_k$ then correct=correct+1;
8  end
9  update $accuracy \leftarrow correct \div N$;
   Return: accuracy
```

resource usage scale-down of up to 20 times. The non-tunable precision heuristic has comparatively very few feature combination options to select from, which leads, in contrast, to a maximum resource scaling of approximately 2 times without accuracy degradation.

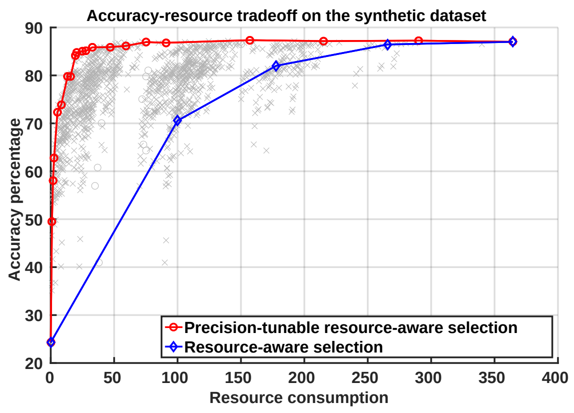


Figure 4. Algorithm performance comparison on the synthetic dataset.

5.2 Real Datasests

We analyze three sensor based applications that benefit from feature precision tuning. The first is a robot navigation task. The second and third dataset are activity recognition tasks.

Wall-Following Robot Navigation We analyze a public domain dataset that was collected as a mobile robot navigates through a room following the wall in a clockwise direction, for 4 rounds, using 4 ultrasound sensors positioned on the front, left, right and back of its body [11]. Four states can be identified from the sensor readings: Move-Forward, Slight-Right-Turn, Sharp-Right-Turn or Slight-Left-Turn. The data-set has a precision of 8 bits and we further quantize it at 5, 2 and 1 bits. We assume the four sensors have the same hardware properties, so we set the variable α_i equal to one for all of them and use Equation 3 to generate the resources for the experiments, as shown in Table 1. Furthermore, we add a random number between 0 and 5 to each cost to simulate non ideal performance conditions.

Figure 5 shows the cost-accuracy trade-off achieved by the proposed precision-tunable heuristic and the trade-off achieved by a cost aware method in red and blue, respectively. The gray crosses represent all the possible operation points to choose from. Both heuristics display negligible accuracy degradation when their resource consumption is scaled down a factor 2 (from 400 to 200). The slight accuracy gain achieved by the non-tunable heuristic can be due to the discretization related improvements discussed in [32] and [29] . For a factor 4 resource scaling (from 400 to 100) the non-tunable heuristic has already pruned 3 out of its four features which causes the accuracy to degrade from 90% to 75%. The precision-tunable heuristic keeps observing all features, yet reduces their precision which also produces a factor 4 resource consumption downscale but no accuracy degradation.

USC-HAD This dataset was designed as a benchmark for Human Activity Detection (HAD) algorithm comparisons [34]. It was collected by an Inertial Measurement Unit (IMU) placed in the subjects'

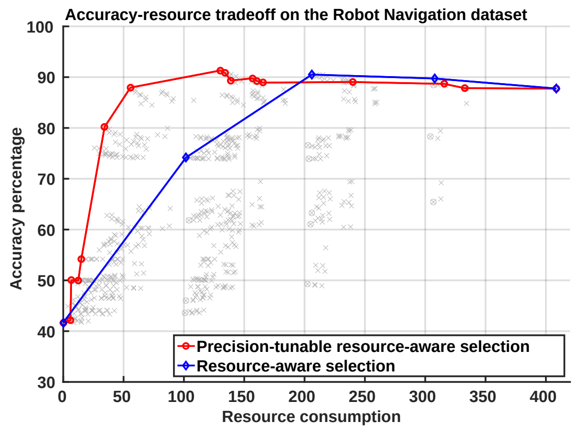


Figure 5. Performance comparison in the robot navigation application

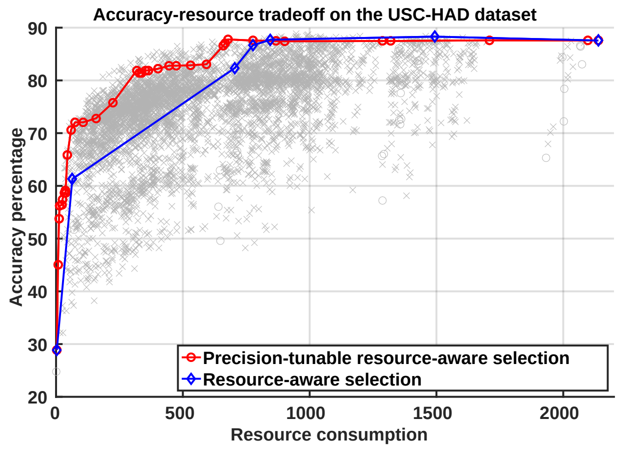


Figure 6. Trade-off comparison on the Human Activity Detection dataset with gyroscope and accelerometer.

hip consisting of a 3-axis accelerometer and a 3-axis gyroscope and it contains measurements for the identification of 12 different low-level daily activities. In accordance to previously performed Activity Recognition analyses [7, 26], the activities that can be best classified with naive Bayes and that are therefore used in this experiment are Walking-forward, Running-Forward, Sitting and Sleeping.

We tuned the dataset's precision from the original 8 bits to 5,4,3,2 and 1 bits. For resource assignment, we consider that the power consumption of a gyroscope can be up to 10 times that of an accelerometer [34] so we set the corresponding α_i variables to 1 and 10, respectively, and use Equation 3 to calculate the resource consumption. Like in the previous experiment, we add a random number between 0 and 5 to simulate non ideal behavior. The resource consumption assignments for this experiment are detailed in Table 1. Again, we enable feature pruning through the addition of the 0 bit leaf to he model.

Figure 6 shows the cost-accuracy trade-off curves achieved by the precision-tunable and the cost-aware only heuristics in red and blue, respectively. The possible operating points are represented by gray crosses. For a resource consumption downscale of 2.5 (from 2130 to 845), the non-precision tunable heuristic suffers from an accuracy degradation of 6% (88% to 82%), while there is no accuracy reduction with the precision-tunable method. Although the 6% accuracy loss/2x cost saving of the non-tunable strategy could be acceptable in some situations, it is worth noting the limitations imposed by the available number of operating points. In addition to the 2.5x resource downscale, only a 30x reduction (from 2130 to 65) is possible at the expense of accuracy degrading from 88% to 61%. The precision-tunable strategy has, in contrast, the possibility to choose from approximately 26 operation points, with up to 6x resource savings before accuracy is lower than 80%.

HAR-RIO In this dataset, 5 activities (Sitting-Down, Standing-Up, Standing, Walking, and Sitting) can be identified from 8 hours of recordings performed by 4 accelerometers positioned in the waist, left thigh, right ankle and right arm of 4 healthy subjects [30]. The accelerometers are tri-axial which results in a total number of 12 features; $\{x_i, y_i, z_i\}$, $i = \{1, 2, 3, 4\}$. For the experiments in this paper, 9 of those features were selected in accordance to previously performed classification algorithm benchmarking [30], namely $\{y_1, z_1, x_2, y_2, z_2, x_3, y_3, z_4, y_4, z_4\}$. The dataset's precision is 8 bits, we quantize it at 4,3,2,1 bits and we add the "null precision" leaf that enables feature pruning.The resource parameters used in this experiment are listed in 1.

Figure 7 shows the results from this experiment with the same color coding as previous. The precision-tunable approach's performance is superior, as it achieves up to 12x resource savings (from 620 to 50) for a maximum accuracy degradation of 4% (from 80% to 76%). The non-tunable strategy displays accuracy losses of less than 5% up to a resource consumption scaling of 3x (620 to 200). For any resource down scaling larger than that, the accuracy degrades more than 10%.

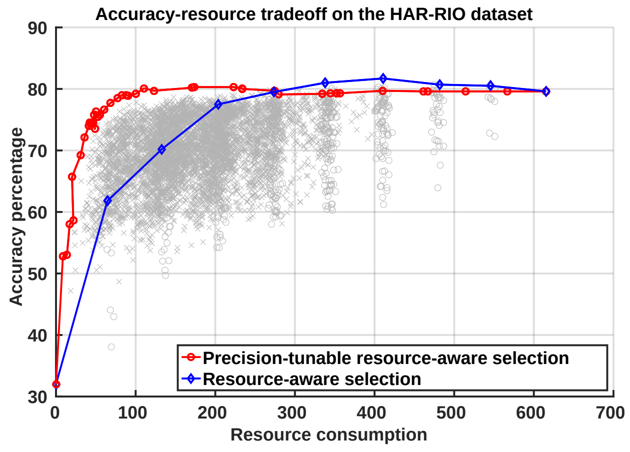


Figure 7. Trade-off comparison on the Human Activity Recognition dataset with accelerometers.

6 CONCLUSIONS AND DISCUSSION

Our main contribution in this paper was to enable efficient embedded sensor fusion through a resource-aware naive Bayes model, capable of exploiting variable precision features. By encapsulating various precision features within the model structure, we enable the possibility to dynamically tune resource consumption and inference accuracy according to the circumstances and available resources. We propose an algorithm that finds optimal operating points by reducing resource consumption and minimizing accuracy degradation.

Table 1. Feature resources used for experiments

		m bits feature precision						
Dataset	α	10	8	5	4	3	2	1
Synthetic								
Feat. 1	1	100	-	25	-	9	4	1
Feat. 2	0.7	70	-	17.5	-	6.3	2.8	0.7
Feat. 3	0.9	90	-	22.5	-	8.1	3.6	0.9
Feat. 4	0.8	80	-	20	-	7.2	3.2	0.8
Robot	1	100	-	25	-	-	4	1
USC-HAD								
Accel.	1	-	64	25	16	9	4	1
Gyro.	10	-	640	250	160	90	40	10
HAR-RIO	1	-	64	25	16	9	4	1

We have compared our scheme with a state-of-the-art resource-aware feature selection technique and we conclude that overall our scheme has better cost saving capabilities due to the rich variety of operational points it can choose from. We tested one artificial and three public domain data corpora with the proposed methodology. Accuracy degradation was prevented while achieving resource usage scalings of 20x for the synthetic dataset, 4x for the Robot Navigation application, 6x for the Human Activity Detection application with accelerometers and gyroscopes, and 12x for the Human Activity Recognition application with accelerometers. The non-tunable precision heuristic achieved, in comparison, a resource scaling of 2x for the synthetic dataset, 2x for the Robot Navigation application, 2.5x for the Human Activity Detection application with accelerometers and gyroscopes, and 3x for the Human Activity Recognition application with accelerometers.

There are many ways in which feature quality tuning can improve hardware resource efficiency. We proved this concept by tuning feature precision but the next step in our work will be to extend the proposed method to other quality tuning paradigms beyond precision tunability such as varying levels of noisy sensing. This will potentially require the modification of the proposed multiple-level Bayesian Network as the relationship between nodes of different qualities will not be deterministic anymore. Furthermore, we will explore more complex structures for applications that are not modeled with sufficient accuracy under the naive Bayes independence assumption.

The long term goal is to integrate the optimal feature precision selection scheme in an embedded sensory application, where the system dynamically and autonomously selects features and their precision given the current state of the hardware devices with limited computational overhead. This scheme could enable the seamless integration of sensory based algorithms into smart environments which is one of the elements envisioned for the IoT.

REFERENCES

[1] Rajesh Arumugam, Vikas Reddy Enti, Liu Bingbing, Wu Xiaojun, Krishnamoorthy Baskaran, Foong Foo Kong, A Senthil Kumar, Kang Dee Meng, and Goh Wai Kit, 'Davinci: A cloud computing framework for service robots', in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pp. 3084–3089. IEEE, (2010).

[2] Luigi Atzori, Antonio Iera, and Giacomo Morabito, 'The internet of things: A survey', *Computer networks*, **54**(15), 2787–2805, (2010).

[3] Komail MH Badami, Steven Lauwereins, Wannes Meert, and Marian Verhelst, 'A 90 nm CMOS, power-proportional acoustic sensing frontend for voice activity detection', *Solid-State Circuits, IEEE Journal of*, **51**(1), 291–302, (2016).

[4] F. H. Bijarbooneh, W. Du, E. C. H. Ngai, X. Fu, and J. Liu, 'Cloud-assisted data fusion and sensor selection for internet of things', *IEEE Internet of Things Journal*, **3**(3), 257–268, (June 2016).

[5] Azzedine Boukerche, Antonio AF Loureiro, Eduardo F Nakamura, Horacio ABF Oliveira, Heitor S Ramos, and Leandro A Villas, 'Cloud-assisted computing for event-driven mobile services', *Mobile Networks and Applications*, **19**(2), 161–170, (2014).

[6] Jie Cheng and Russell Greiner, 'Comparing Bayesian network classifiers', in *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pp. 101–108. Morgan Kaufmann Publishers Inc., (1999).

[7] Jian Cui and Bin Xu, 'Cost-effective activity recognition on mobile devices', in *Proceedings of the 8th International Conference on Body Area Networks*, BodyNets '13, pp. 90–96, ICST, Brussels, Belgium, Belgium, (2013). ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[8] Artem Dementyev, Steve Hodges, Stephen Taylor, and Johan Smith, 'Power consumption analysis of bluetooth low energy, zigbee and ant sensor nodes in a cyclic sleep scenario', in *Wireless Symposium (IWS), 2013 IEEE International*, pp. 1–4. IEEE, (2013).

[9] K. Frank, P. Robertson, S. Fortes Rodriguez, and R. Barco Moreno, 'Faster bayesian context inference by using dynamic value ranges', in *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2010 8th IEEE International Conference on*, pp. 50–55, (2010).

[10] K. Frank, M. Rckl, and P. Robertson, 'The bayeslet concept for modular context inference', in *Mobile Ubiquitous Computing, Systems, Services and Technologies, 2008. UBICOMM '08. The Second International Conference on*, pp. 96–101, (Sept 2008).

[11] A. L. Freire, G. A. Barreto, M. Veloso, and A. T. Varela, 'Short-term memory mechanisms in neural network learning of robot navigation tasks: A case study', in *Robotics Symposium (LARS), 2009 6th Latin American*, pp. 1–6, (Oct 2009).

[12] Nir Friedman, Dan Geiger, and Moises Goldszmidt, 'Bayesian network classifiers', *Journal of Machine Learning*, **29**(2), 131–163, (1997).

[13] Isabelle Guyon and André Elisseeff, 'An introduction to variable and feature selection', *J. Mach. Learn. Res.*, **3**, 1157–1182, (March 2003).

[14] Josué Iglesias, Ana M. Bernardos, Paula Tarrío, José R. Casar, and Henar Martín, 'Design and validation of a light inference system to support embedded context reasoning', *Personal and Ubiquitous Computing*, **16**(7), 781–797, (2012).

[15] Peter Kinget and Michiel Steyaert, 'Impact of transistor mismatch on the speed-accuracy-power trade-off of analog CMOS circuits.', in *Proceedings of the IEEE Custom Integrated Circuits Conference*, pp. 333–336, (1996).

[16] N. D. Lane, S. Bhattacharya, P. Georgiev, C. Forlivesi, L. Jiao, L. Qendro, and F. Kawsar, 'Deepx: A software accelerator for low-power deep learning inference on mobile devices', in *2016 15th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, pp. 1–12, (April 2016).

[17] Steven Lauwereins, Wannes Meert, Jort Gemmeke, and Marian Verhelst, 'Ultra-low-power voice-activity-detector through context- and resource-cost-aware feature selection in decision trees', in *2014 IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, pp. 1–6, (Sept 2014).

[18] Jonathan Lester, Tanzeem Choudhury, Nicky Kern, Gaetano Borriello, and Blake Hannaford, 'A hybrid discriminative/generative approach for modeling human activities.', in *IJCAI*, volume 5, pp. 766–772, (2005).

[19] Daniel Lowd and Pedro Domingos, 'Learning arithmetic circuits', *arXiv preprint arXiv:1206.3271*, (2012).

[20] Bert Moons, Bert De Brabandere, Luc Van Gool, and Marian Verhelst, 'Energy-efficient convnets through approximate computing', in *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 1–8. IEEE, (2016).

[21] Bert Moons and Marian Verhelst, 'Energy and accuracy in multi-stage stochastic computing', in *New Circuits and Systems Conference (NEWCAS), 2014 IEEE 12th International*, pp. 197–200. IEEE, (2014).

[22] Bert Moons and Marian Verhelst, 'Dvas: Dynamic voltage accuracy scaling for increased energy-efficiency in approximate computing', in *Low Power Electronics and Design (ISLPED), 2015 IEEE/ACM International Symposium on*, pp. 237–242. IEEE, (2015).

[23] Bert Moons and Marian Verhelst, 'A 40nm CMOS, 35mW to 270mW,

precision-scalable cnn processor for run-time scalable embedded vision', in *IEEE VLSI Symposium*. IEEE, (2016).

[24] Judea Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann, 1988.

[25] Nico Piatkowski, Sangkyun Lee, and Katharina Morik, 'Integer undirected graphical models for resource-constrained systems', *Neurocomputing*, **173**, 9–23, (2016).

[26] Olga Politi, Iosif Mporas, and Vasileios Megalooikonomou, 'Human motion detection in daily activity tasks using wearable sensors', in *Signal Processing Conference (EUSIPCO), 2014 Proceedings of the 22nd European*, pp. 2315–2319. IEEE, (2014).

[27] John G. Proakis and Dimitris K Manolakis, *Digital Signal Processing (4th Edition)*, Pearson, 2006.

[28] Yvan Saeys, Iaki Inza, and Pedro Larraaga, 'A review of feature selection techniques in bioinformatics', *Bioinformatics*, **23**(19), 2507–2517, (2007).

[29] Sebastian Tschiatschek and Franz Pernkopf, 'On Bayesian network classifiers with reduced precision parameters', *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **37**(4), 774–785, (2015).

[30] Wallace Ugulino, Débora Cardador, Katia Vega, Eduardo Velloso, Ruy Milidiú, and Hugo Fuks, 'Wearable computing: Accelerometers data classification of body postures and movements', in *Advances in Artificial Intelligence-SBIA 2012*, 52–61, Springer, (2012).

[31] Zhixiang Xu, Matt J Kusner, Kilian Q Weinberger, Minmin Chen, and Olivier Chapelle, 'Classifier cascades and trees for minimizing feature evaluation cost', *The Journal of Machine Learning Research*, **15**(1), 2113–2144, (2014).

[32] Ying Yang and Geoffrey I Webb, 'Proportional k-interval discretization for naive-Bayes classifiers', in *ECML*, pp. 564–575. Springer, (2001).

[33] Harry Zhang, 'The optimality of naive Bayes', in *Proceedings of FLAIRS*, (2004).

[34] Mi Zhang and Alexander A. Sawchuk, 'USC-HAD: A daily activity dataset for ubiquitous activity recognition using wearable sensors', in *ACM International Conference on Ubiquitous Computing (Ubicomp) Workshop on Situation, Activity and Goal Awareness (SAGAware)*, Pittsburgh, Pennsylvania, USA, (September 2012).

A Distributed Event Calculus for Event Recognition

Alexandros Mavrommatis[1 2], **Alexander Artikis**[3 2], **Anastasios Skarlatidis**[2] **and Georgios Paliouras**[2]

Abstract. Events provide a fundamental abstraction for representing time-evolving information. Complex event recognition focuses on tracking and analysing streams of events, in order to detect patterns of special significance. The event streams may originate from various types of sensor, such as cameras and GPS sensors. Furthermore, the stream velocity and volume pose significant challenges to event processing systems. We propose dRTEC, an event recognition system that employs the Event Calculus formalism and operates in multiple processing threads. We evaluate dRTEC using two real-world applications and show that it is capable of real-time and scalable event recognition.

1 Introduction

Today's organisations need to act upon Big Data streams in order to support their resource management, capitalise on opportunities and detect threats. Towards this, event recognition systems have been particularly helpful, as they support the detection of complex events (CE)s of special significance, given streams of 'simple, derived events' (SDE)s arriving from various types of sensor [9]. A CE is a collection of events (SDEs and/or CEs) that satisfy a (spatio-)temporal pattern. In the maritime domain, for example, event recognition systems have been used to make sense of position streams emitted from thousands of vessels, in order to detect, in real-time, suspicious and illegal activity that may have dire effects in the maritime ecosystem and passenger safety [2].

In previous work, we developed the 'Event Calculus for Run-Time reasoning' (RTEC), a formal computational framework for event recognition [3]. RTEC is an Event Calculus dialect [18] that includes optimisation techniques supporting efficient event recognition. A form of caching stores the results of sub-computations in the computer memory to avoid unnecessary re-computations. A simple indexing mechanism makes RTEC robust to events that are irrelevant to the computations we want to perform. A set of interval manipulation constructs simplify event patterns and improve reasoning efficiency. Furthermore, a 'windowing' mechanism makes event recognition history-independent.

RTEC is a logic programming implementation of the Event Calculus. This way, event patterns have a formal, declarative semantics [3]. On the other hand, RTEC does not have built-in support for distributed processing. This is a significant limitation, as Big Data applications, such as maritime monitoring, require the processing of high velocity SDE streams. Moreover, the integration of RTEC into non-Prolog systems is not always straightforward, requiring workarounds that hinder performance.

To deal with the increasing velocity and volume demands of today's applications, we developed 'dRTEC', a distributed implementation of RTEC. dRTEC employs Spark Streaming[4], an extension of the Apache Spark API that enables scalable, high-throughput and fault-tolerant stream processing. Reasoning in Spark Streaming may be performed exclusively in memory, where the input SDE stream is aggregated into a series of batch computations on small time intervals. dRTEC uses Spark Streaming's inherent support for distributed processing to take advantage of modern multi-core hardware for scalable event recognition.

The use of Spark Streaming additionally facilitates the integration of dRTEC, as an event recognition module, into existing (large-scale) stream processing systems. dRTEC has been evaluated in the context of two such systems. First, in the context of the SYNAISTHISI project[5], dRTEC is the human activity recognition module detecting 'long-term activities', such as fighting and leaving unattended objects, given 'short-term' activities detected on video frames by the underlying visual information processing components. In this application, we evaluated dRTEC using a benchmark activity recognition dataset. Second, in the datACRON project[6], dRTEC recognises suspicious and illegal vessel activities given a compressed vessel position stream produced by a trajectory processing module. To evaluate dRTEC on the maritime domain, we used a real position stream from over 6,000 vessels sailing through the Greek seas in the summer of 2009. The empirical analysis showed that dRTEC scales better than RTEC, both to increasing velocity SDE streams and larger numbers of CEs.

The remainder of the paper is organised as follows. In the following section we briefly review RTEC. Then, we introduce they key components of dRTEC. Section 4 presents our empirical analysis, while in Section 5 we summarise our approach, discuss related work and outline directions for further research.

2 Event Calculus for Run-Time Reasoning

dRTEC is a distributed implementation of RTEC[7], the 'Event Calculus for Run-Time reasoning' [3]. The time model of RTEC is linear including integer time-points. Where F is a *fluent*—a property that is allowed to have different values at different points in time—the term $F = V$ denotes that fluent F has value V. Table 1 presents the main RTEC predicates. Variables start with an upper-case letter, while predicates and constants start with a lower-case letter. The happensAt predicate defines the event instances, the initiatedAt and termi-

[1] School of Electronic & Computer Engineering, Technical University of Crete, Greece
[2] Institute of Informatics & Telecommunications, NCSR "Demokritos", Greece
[3] Department of Maritime Studies, University of Piraeus, Greece {blackeye,a.artikis,anskarl,paliourg}@iit.demokritos.gr

[4] http://spark.apache.org/streaming/
[5] http://iot.synaisthisi.iit.demokritos.gr/
[6] http://www.datacron-project.eu/
[7] https://github.com/aartikis/RTEC

natedAt predicates express the effects of events, while the holdsAt and holdsFor predicates express the values of the fluents. holdsAt and holdsFor are defined in such a way that, for any fluent F, holdsAt$(F = V,\ T)$ if and only if T belongs to one of the maximal intervals of I for which holdsFor$(F = V,\ I)$.

We represent instantaneous SDEs and CEs by means of happensAt, while durative SDEs and CEs are represented as fluents. The majority of CEs are durative and thus, in CE recognition the task is to compute the maximal intervals for which a fluent representing a CE has a particular value continuously.

Table 1. RTEC Predicates.

Predicate	Meaning
happensAt$(E,\ T)$	Event E occurs at time T
initiatedAt$(F = V,\ T)$	At time T a period of time for which $F = V$ is initiated
terminatedAt$(F = V,\ T)$	At time T a period of time for which $F = V$ is terminated
holdsAt$(F = V,\ T)$	The value of fluent F is V at time T
holdsFor$(F = V,\ I)$	I is the list of the maximal intervals for which $F = V$ holds continuously
union_all(L, I)	I is the list of maximal intervals produced by the union of the lists of maximal intervals of list L
intersect_all(L, I)	I is the list of maximal intervals produced by the intersection of the lists of maximal intervals of list L
relative_complement_all(I', L, I)	I is the list of maximal intervals produced by the relative complement of the list of maximal intervals I' with respect to every list of maximal intervals of list L

Fluents in RTEC are of two kinds: *simple* and *statically determined*. For a simple fluent F, $F = V$ holds at a particular time-point T if $F = V$ has been *initiated* by an event that has occurred at some time-point earlier than T, and has not been *terminated* in the meantime. This is an implementation of the *law of inertia*. To compute the *intervals* I for which $F = V$ holds continuously, i.e. holdsFor$(F = V,\ I)$, we compute all time-points T_s at which $F = V$ is initiated, and then, for each T_s, we find the first time-point T_f after T_s at which $F = V$ is terminated. Consider the following example from activity recognition:

$$\begin{array}{l}
\text{initiatedAt}(\mathit{leaving\_object}(P,\ \mathit{Obj}) = \text{true},\ T) \leftarrow \\
\quad \text{happensAt}(\mathit{appear}(\mathit{Obj}),\ T), \\
\quad \text{holdsAt}(\mathit{inactive}(\mathit{Obj}) = \text{true},\ T), \\
\quad \text{holdsAt}(\mathit{close}(P,\ \mathit{Obj}) = \text{true},\ T), \\
\quad \text{holdsAt}(\mathit{person}(P) = \text{true},\ T)
\end{array} \qquad (1)$$

$$\begin{array}{l}
\text{terminatedAt}(\mathit{leaving\_object}(P,\ \mathit{Obj}) = \text{true},\ T) \leftarrow \\
\quad \text{happensAt}(\mathit{disappear}(\mathit{Obj}),\ T)
\end{array} \qquad (2)$$

The above rules are intended to capture the activity of leaving an object unattended. appear and $\mathit{disappear}$ are instantaneous SDEs produced by the underlying computer vision algorithms. An entity 'appears' when it is first tracked. Similarly, an entity 'disappears' when it stops being tracked. An object carried by a person is not tracked—only the person that carries it is tracked. The object will be tracked, that is, it will 'appear', if and only if the person leaves it somewhere. $\mathit{inactive}$ is a durative SDE. Objects (as opposed to persons) can exhibit only inactive activity. $\mathit{close}(P,\ \mathit{Obj})$ is a statically determined fluent indicating whether the distance between two entities P and Obj, tracked in the surveillance videos, is less than some threshold of pixel positions. $\mathit{person}(P)$ is a simple fluent indicating whether there is sufficient information that entity P is a person as opposed to an object. According to rule (1), 'leaving object' is initiated when an inactive entity starts being tracked close to a person. Rule (2) dictates that 'leaving object' stops being recognised when the entity is no longer tracked. The maximal intervals during which $\mathit{leaving\_object}(P,\ \mathit{Obj}) = \text{true}$ holds continuously are computed using the built-in RTEC predicate holdsFor from rules (1) and (2).

In addition to the domain-independent definition of holdsFor, RTEC supports application-dependent holdsFor rules, used to define the values of a fluent F in terms of the values of other fluents. Such a fluent F is called *statically determined*. holdsFor rules of this type make use of interval manipulation constructs—see the last three items of Table 1. Consider the following example:

$$\begin{array}{l}
\text{holdsFor}(\mathit{greeting}(P_1, P_2) = \text{true},\ I) \leftarrow \\
\quad \text{holdsFor}(\mathit{close}(P_1, P_2) = \text{true},\ I_1), \\
\quad \text{holdsFor}(\mathit{active}(P_1) = \text{true},\ I_2), \\
\quad \text{holdsFor}(\mathit{inactive}(P_1) = \text{true},\ I_3), \\
\quad \text{holdsFor}(\mathit{person}(P_1) = \text{true},\ I_4), \\
\quad \text{intersect\_all}([I_3, I_4],\ I_5), \\
\quad \text{union\_all}([I_2, I_5],\ I_6), \\
\quad \text{holdsFor}(\mathit{person}(P_2) = \text{true},\ I_7), \\
\quad \text{holdsFor}(\mathit{running}(P_2) = \text{true},\ I_8), \\
\quad \text{holdsFor}(\mathit{abrupt}(P_2) = \text{true},\ I_9), \\
\quad \text{relative\_complement\_all}(I_7,\ [I_8, I_9],\ I_{10}), \\
\quad \text{intersect\_all}([I_1, I_6, I_{10}],\ I)
\end{array} \qquad (3)$$

In activity recognition, we are interested in detecting whether two people are greeting each other. A greeting distinguishes meetings from other, related types of interaction. Similar to $\mathit{inactive}$, active (mild body movement without changing location), $\mathit{running}$ and abrupt are durative SDEs produced by the vision algorithms. According to rule (3), two tracked entities P_1 and P_2 are said to be greeting, if they are close to each other, P_1 is active or an inactive person, and P_2 is a person that is neither running nor moving abruptly.

RTEC restricts attention to *hierarchical* formalisations, those where it is possible to define a function *level* that maps all fluents and all events to the non-negative integers as follows. Events and statically determined fluents of level 0 are those whose happensAt and holdsFor definitions do not depend on any other events or fluents. In CE recognition, they represent the input SDEs. There are no simple fluents in level 0. Events and simple fluents of level n (n>0) are defined in terms of at least one event or fluent of level $n-1$ and a possibly empty set of events and fluents from levels lower than $n-1$. Statically determined fluents of level n are defined in terms of at least one fluent of level $n-1$ and a possibly empty set of fluents from levels lower than $n-1$.

RTEC performs CE recognition by means of continuous query processing, and concerns the computation of the maximal intervals of fluents. At each query time Q_i, the input entities that fall within a specified sliding window ω are taken into consideration. All input entities that took place before or at $Q_i - \omega$ are discarded/'forgotten'. This constraint ensures that the cost of CE recognition depends only on the size of ω and not on the complete SDE history. The size of ω, and the temporal distance between two consecutive query times—the 'step' $Q_i - Q_{i-1}$—are tuning parameters that can be chosen by the user.

When ω is longer than the step $Q_i - Q_{i-1}$, it is possible that an SDE occurs in the interval $(Q_i - \omega, Q_{i-1}]$ but arrives at RTEC only after Q_{i-1}; its effects are taken into account at query time Q_i. This

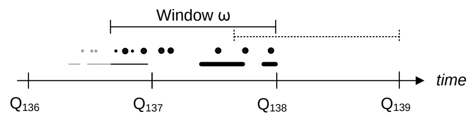


Figure 1. Windowing in RTEC.

is illustrated in Figure 1. The figure displays the occurrences of instantaneous SDEs as dots and durative ones as line segments. For CE recognition at Q_{138}, only the SDEs marked in black are considered, whereas the greyed out ones are neglected. Assume that all SDEs marked in bold arrived only after Q_{137}. Then, we observe that two SDEs were delayed i.e. they occurred before Q_{137}, but arrived only after Q_{137}. In this example, the window is larger than the step. Hence, these SDEs are not lost but considered as part of CE recognition at Q_{138}.

After 'forgetting' SDEs, RTEC computes and stores the intervals of CEs. At Q_i, the CE intervals computed by RTEC are those that can be derived from SDEs that occurred in the interval $(Q_i{-}\omega, Q_i]$, as recorded at time Q_i. RTEC adopts a caching technique where fluents are processed in a bottom-up manner; this way, the intervals of the fluents that are required for the processing of a fluent of level n will simply be fetched from the cache without the need for re-computation. More details about the reasoning engine of RTEC (including a complexity analysis), as well as its expressivity, may be found at [3].

3 Distributed Event Calculus

dRTEC is a distributed implementation of RTEC in Spark Streaming using the Scala programming language. Like RTEC, dRTEC performs CE recognition by means of continuous temporal projection, i.e. at each query time dRTEC computes the maximal intervals of fluents given an incoming SDE stream. Other tasks offered by other Event Calculus implementations, such as abduction, are not supported. In addition to the optimisation techniques of RTEC, such as windowing, dRTEC supports CE recognition using a structured set of operations for distributed reasoning. dRTEC follows a syntax-based, application-independent approach to translate query processing into distributed reasoning. Figure 2 illustrates the basic components of the engine using the activity recognition application. dRTEC accepts SDE streams through MQTT[8], a lightweight publish-subscribe messaging transport. Spark Streaming separates the incoming stream into individual sets, called 'micro-batches'. The window in dRTEC may contain one or more micro-batches. Each micro-batch may contain events, expressed by happensAt, and fluents, expressed by holdsFor. For example, according to the SDEs in the first micro-batch shown in Figure 2, the entity id0 started being tracked—'appeared'—at time/video frame 80. Moreover, the entity id1 was running continuously in the interval [90,100).

dRTEC performs various tasks on the incoming SDE streams. These are presented in the sections that follow. (We focus on the novel components of dRTEC, discussing only briefly the implementation of the RTEC reasoning techniques in Spark Streaming.) The CEs that are recognised using the incoming SDEs are streamed out through MQTT (see 'Output Stream' in Figure 2).

[8] http://mqtt.org/

3.1 Dynamic Grounding & Indexing

At each recognition time Q_i, RTEC grounds the CE patterns using a set of constants for the variables appearing in the patterns, except the variables related to time. Moreover, RTEC operates under the assumption that the set of constants is 'static', in the sense that it does not change over time, and known in advance. For instance, in the maritime surveillance domain, RTEC operates under the assumption that all vessel ids are known beforehand. Similarly, in activity recognition all ids of the tracked entities are assumed to be known. For many application domains, this assumption is unrealistic. More importantly, there are (many) query times in which RTEC attempts to recognise CEs for (many) constants, for which no information exists in the current window.

To address this issue, dRTEC supports 'dynamic' grounding. At each query time Q_i, dRTEC scans the SDEs of the current window ω to construct the list of entities for which CE recognition should be performed. Then, it appends to this list all entities that have CE intervals overlapping $Q_i{-}\omega$. Such intervals may be extended or (partially) retracted, given the information that is available in the current window. In this manner, dRTEC avoids unnecessary calculations by restricting attention to entities for which a CE may be recognised at the current query time.

Indexing is used to convert the input SDEs into a key-value pair format for data partitioning. The partitions are distributed among the available cores (processing threads) of the underlying hardware for parallel processing. Each SDE is indexed according to its entity. In activity recognition, for example, the index concerns the ids of the tracked entities (see 'Dynamic Grounding & Indexing' in Figure 2). For each window, the SDEs concerning the same entity are grouped together and subsequently sent to the same processing thread.

3.2 Non-Relational Processing

Indexing is followed by non-relational fluent processing performed at each thread in parallel (see the 'Non-Relational Processing' boxes of Figure 2). Non-relational processing refers to the computation of the maximal intervals of fluents involving a single entity. (In the absence of such fluents, dRTEC proceeds directly to 'pairing'.) In activity recognition, for example, we want to determine whether a tracked entity is a human or an object (see the rules presented in Section 2). An entity is said to be a person if it has exhibited one of the 'running', 'active', 'walking' or 'abrupt movement' short-term behaviours since it started being tracked. In other words, the classification of an entity as a person or an object depends only the short-term activities of that entity. The distinction between non-relational and relational processing allows us to trivially parallelise a significant part of the CE recognition process (non-relational CE patterns). The processing threads are independent from one another, avoiding data transfers among them that are very costly.

Non-relational, as well as relational processing, concerns both statically determined and simple fluent processing, and windowing. These tasks are discussed in Section 3.4.

3.3 Pairing & Relational Processing

Relational processing concerns CE patterns that involve two or more entities. In activity recognition, we want to recognise whether two people are moving together or fighting. Prior to relational CE recognition, dRTEC produces all possible relations that may arise from the list of entities computed by the dynamic grounding process—see 'Pairing' in Figure 2. Then, these relations are distributed to all

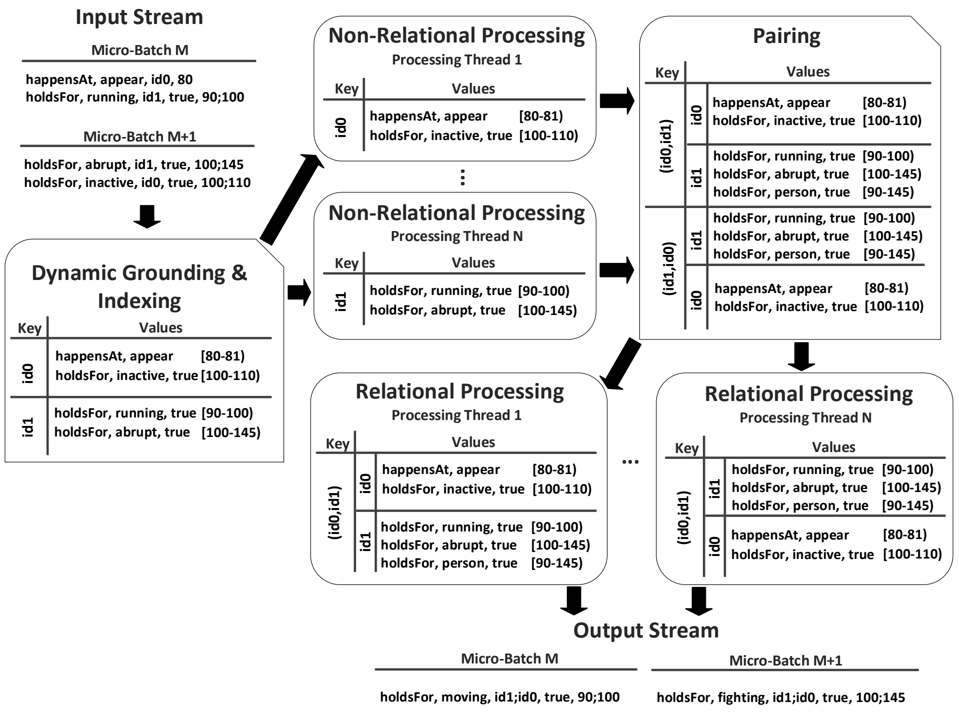

Input Stream
Micro-Batch M
happensAt, appear, id0, 80
holdsFor, running, id1, true, 90;100
Micro-Batch M+1
holdsFor, abrupt, id1, true, 100;145
holdsFor, inactive, id0, true, 100;110
Dynamic Grounding & Indexing
Key
Values
id0
happensAt, appear [80-81)
holdsFor, inactive, true [100-110)
id1
holdsFor, running, true [90-100)
holdsFor, abrupt, true [100-145)
Non-Relational Processing
Processing Thread 1
Processing Thread N
Pairing
(id0,id1)
(id1,id0)
holdsFor, person, true [90-145)
Relational Processing
Output Stream
holdsFor, moving, id1;id0, true, 90;100
holdsFor, fighting, id1;id0, true, 100;145

Figure 2. dRTEC processing.

available processing threads for parallel CE recognition. Note that, in contrast to non-relational processing, the information available to each processing thread is not disjoint. Assume, for example, that the pair (id0, id1) is processed by processing thread 1, while the pair (id1, id2) is processed by thread 2. Then both threads will have the output of non-relational processing concerning id1 (e.g. the list of maximal intervals during which id1 is said to be a person). However, there is no replication of computation, as the output of non-relational processing is cached, and the sets of relations of the processing threads are disjoint. Furthermore, similar to non-relational processing, each processing thread has all the necessary information, thus avoiding costly data transfers.

3.4 Fluent Processing

As mentioned earlier, both relational and non-relational processing concern the computation of the list of maximal intervals of fluents. For both types of fluent, simple and statically determined, dRTEC follows the reasoning algorithms of RTEC. For example, in the case of a simple fluent CE_s, dRTEC checks, at each query time Q_i, if there is a maximal interval of CE_s that overlaps $Q_i{-}\omega$. If there is such an interval then it will be discarded, while its initiating point will be kept. Then, dRTEC computes the initiating points of CE_s in $(Q_i{-}\omega, Q_i]$, and appends them to initiating point (if any) prior to $Q_i{-}\omega$. If the list of initiating points is empty then the empty list of intervals is returned. Otherwise, dRTEC computes the terminating points of CE_s in $(Q_i{-}\omega, Q_i]$, and pairs adjacent initiating and terminating points, as discussed in Section 2, to produce the maximal intervals.

Definitions 1 and 2 show, respectively, the initiating and terminating conditions of the 'leaving object' CE that were presented in Section 2 in the language of RTEC. Recall that 'leaving object' is a simple fluent. The GI function (GETINTERVAL, in full) retrieves the list of maximal intervals of a fluent. GI has three parameters: (a) the

Definition 1 Initiation of *leaving object* in dRTEC.

```
$I1 \leftarrow \text{GI}(occurrences,\ Obj,\ \texttt{Fluent}(happensAt,\ appear))$
$I2 \leftarrow \text{GI}(occurrences,\ Obj,\ \texttt{Fluent}(holdsFor,\ inactive,\ true))$
$I3 \leftarrow \text{GI}(occurrences,\ (P,\ Obj),\ \texttt{Fluent}(holdsFor,\ close,\ true))$
$I4 \leftarrow \text{GI}(occurrences,\ P,\ \texttt{Fluent}(holdsFor,\ person,\ true))$
$I \leftarrow I1.\text{INTERSECT\_ALL}(I2).\text{INTERSECT\_ALL}(I3).\text{INTERSECT\_ALL}(I4)$
```

'collection occurrences', i.e. a map pointing to the list of maximal intervals of a fluent, (b) the list of entities/arguments of the fluent, and (c) the fluent object. dRTEC uses exclusively intervals in its patterns. The occurrence of an event (e.g. 'appear') is represented by an instantaneous interval. This way, in addition to statically determined fluents, the interval manipulation constructs can be used for specifying simple fluents. In dRTEC these constructs are supported by 'interval instances' (see e.g. the last line of Definition 1).

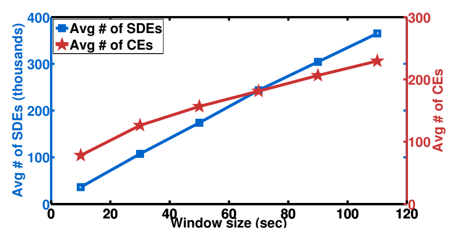
Definition 2 Termination of *leaving object* in dRTEC.

```
$I \leftarrow \text{GI}(occurrences,\ Obj,\ \texttt{Fluent}(happensAt,\ disappear))$
```

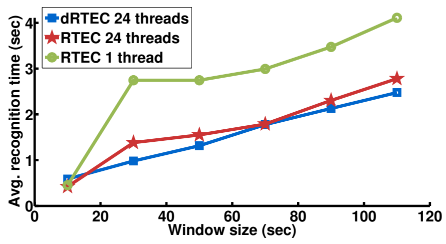
Statically determined fluents in dRTEC are specified in a similar manner. Definition 3, for example, shows the specification of 'greeting' (see rule (3) for the RTEC representation).

Definition 3 Statically determined fluent *greeting* in dRTEC.

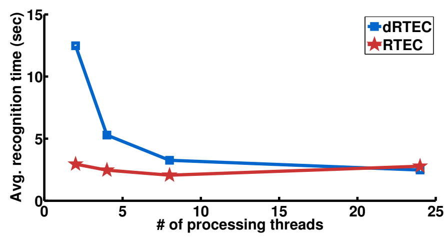
```
$I1 \leftarrow \text{GI}(occurrences,\ (P1,\ P2),\ \texttt{Fluent}(holdsFor,\ close,\ true))$
$I2 \leftarrow \text{GI}(occurrences,\ P1,\ \texttt{Fluent}(holdsFor,\ active,\ true))$
$I3 \leftarrow \text{GI}(occurrences,\ P1,\ \texttt{Fluent}(holdsFor,\ inactive,\ true))$
$I4 \leftarrow \text{GI}(occurrences,\ P1,\ \texttt{Fluent}(holdsFor,\ person,\ true))$
$I5 \leftarrow I3.\text{INTERSECT\_ALL}(I4)$
$I6 \leftarrow I2.\text{UNION\_ALL}(I5)$
$I7 \leftarrow \text{GI}(occurrences,\ P2,\ \texttt{Fluent}(holdsFor,\ person,\ true))$
$I8 \leftarrow \text{GI}(occurrences,\ P2,\ \texttt{Fluent}(holdsFor,\ running,\ true))$
$I9 \leftarrow \text{GI}(occurrences,\ P2,\ \texttt{Fluent}(holdsFor,\ abrupt,\ true))$
$I10 \leftarrow I7.\text{RELATIVE\_COMPLEMENT\_ALL}(I8.\text{UNION\_ALL}(I9))$
$I \leftarrow I1.\text{INTERSECT\_ALL}(I6).\text{INTERSECT\_ALL}(I10)$
```



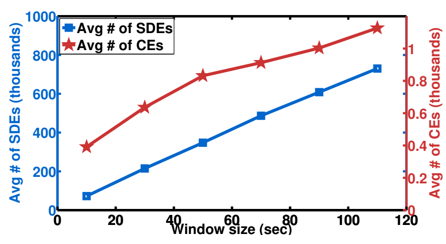
(a) Number of SDEs and CEs.



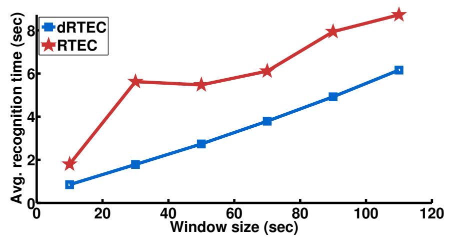
(b) dRTEC vs RTEC.



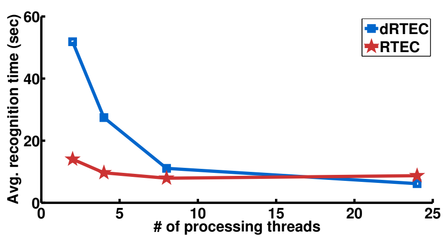
(c) dRTEC vs RTEC: 110 sec window.



(d) Number of SDEs and CEs.



(e) dRTEC vs RTEC: 24 processing threads.



(f) dRTEC vs RTEC: 110 sec window.

Figure 3. Activity recognition. Figures (a)–(c) (respectively (d)–(f)) concern the dataset with 10 (20) tracked entities.

4 Empirical Analysis

dRTEC has been evaluated in the context of two stream processing systems. In the system of the SYNAISTHISI project, dRTEC is the long-term activity recognition module operating on short-term activities detected on video frames. In the datACRON project, dRTEC recognises suspicious and illegal vessel activities given a compressed vessel position stream produced by a trajectory processing module. The empirical analysis presented below was performed on a computer with dual Intel Xeon E5-2630 processors, amounting to 24 processing threads, and 256GB RAM, running Ubuntu 14.04 LTS 64-Bit with Linux kernel 3.13 and Java OpenJDK 1.8. dRTEC is implemented in Apache Spark Streaming 1.5.2 using Scala 2.11.7. The source code, including the CE patterns for both applications, is publicly available[9]. dRTEC's warm up period is excluded from the presented results. In all cases, dRTEC recognises the same CEs as RTEC.

4.1 Activity Recognition

The SYNAISTHISI project aims at developing customisable, distributed, low-cost security and surveillance solutions. To evaluate dRTEC, we used the CAVIAR benchmark dataset[10] which consists of 28 surveillance videos of a public space. The CAVIAR videos show actors which are instructed to carry out several scenarios. Each video has been manually annotated by the CAVIAR team to provide the ground truth for activities which take place on individual video frames. These short-term activities are: entering and exiting the surveillance area, walking, running, moving abruptly, being active and being inactive. We view these activities as SDEs. The CAVIAR team has also annotated the videos with long-term activities: a person leaving an object unattended, people having a meeting, moving together, and fighting. These are the CEs that we want to recognise.

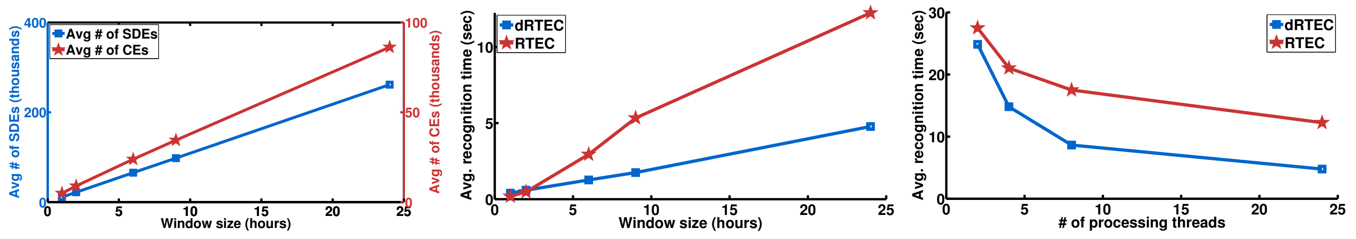
The CAVIAR dataset includes 10 tracked entities, i.e. 90 entity pairs (most CEs in this application concern a pair of entities), while the frame rate is 40 milliseconds (ms). On average, 179 SDEs are detected per second (sec). To stress test dRTEC, we constructed a larger dataset. Instead of reporting SDEs every 40 ms, the enlarged dataset provides data in every ms. The SDEs of video frame/time k of the original dataset are copied 39 times for each subsequent ms after time k. The resulting dataset has on average of 3,474 SDEs per sec. Figures 3(a)–3(c) show the experimental results on this dataset. We varied the window size from 10 sec to 110 sec. The slide step $Q_i - Q_{i-1}$ was set to be equal to the size of the window. Figure 3(a) shows the average number of SDEs per window size. The 10 sec window corresponds to approximately 36K SDEs while the 110 sec one corresponds to 365K SDEs. Figure 3(a) also shows the number of recognised CEs; these range from 80 to 230.

The average CE recognition times per window (in CPU seconds) for both dRTEC and RTEC are shown in Figure 3(b). dRTEC made use of all 24 processing threads. With the exception of the smallest window size, dRTEC outperforms RTEC. To allow for a fairer comparison, we invoked 24 instances of RTEC, each using in parallel one processing thread of the underlying hardware. Every RTEC instance was set to perform CE recognition for at most 4 entity pairs, and was provided only with the SDEs concerning the entities of these pairs (no load balancing was performed). In this setting, dRTEC outperforms RTEC for most window sizes, but only slightly.

Figure 3(c) shows the effect of increasing the number of available processing threads on the performance of dRTEC and RTEC. We varied the number of available threads from 2 to 24; the window size was set to 110 sec. RTEC achieves its best performance early—the increase of processing threads affects it only slightly. In contrast, dRTEC requires all 24 processing threads to match (slightly outperform) RTEC. The cost of data partitioning through dynamic grounding and indexing in dRTEC pays off only in the case of 24 threads.

To stress test further dRTEC, we constructed an even larger dataset by adding a copy of the previous dataset with new identifiers for the tracked entities. Thus, the resulting dataset contains a total of 20 tracked entities and 380 entity pairs, while approximately 7K SDEs take place per sec. Figures 3(d)–3(e) show the experimental results. We varied again the window size from 10 sec to 110 sec. In this case, however, the SDEs range from 72K to 730K (see Figure 3(d)). The number of recognised CEs is also much higher; it ranges from 390 to 1100.

[9] `https://github.com/blackeye42/dRTEC`
[10] http://groups.inf.ed.ac.uk/vision/CAVIAR/CAVIARDATA1



(a) Number of SDEs and CEs. (b) dRTEC vs RTEC: 24 processing threads. (c) dRTEC vs RTEC: 24 hour window.

Figure 4. Event recognition for maritime surveillance.

Figure 3(e) shows the average CE recognition times per window when all 24 processing threads were available both to dRTEC and RTEC. Each RTEC instance was set to perform CE recognition for at most 16 entity pairs, having available only the SDEs concerning the entities of these pairs. Both dRTEC and RTEC remain real-time, even in the presence of 730K SDE windows. In this set of experiments, dRTEC outperforms RTEC in all window sizes, and the difference is more significant. This is an indication that dRTEC scales better to larger datasets. Figure 3(f) shows the effect of increasing the number of processing threads. We observe a similar pattern to that of the previous experiments (see Figure 3(c)).

4.2 Maritime Surveillance

The datACRON project aims to develop novel methods for detecting threats and abnormal activity in very large numbers of moving entities operating in large geographic areas. In the stream processing system of datACRON, dRTEC serves as the component recognising various types of suspicious and illegal vessel activity. We conducted experiments against a real position stream from the Automated Identification System[11], spanning from 1 June 2009 to 31 August 2009, for 6,425 vessels sailing through the Aegean, the Ionian, and part of the Mediterranean Sea[12]. The trajectory detection module of datACRON compresses the vessel position stream to a stream of critical movement events of the following types: 'low speed', 'speed change', 'gap', indicating communication gaps, 'turn', and 'stopped', indicating that a vessel has stopped in the open sea. Each such event includes the coordinates, speed and heading of the vessel at the time of critical event detection. This way, the SDE stream includes 15,884,253 events. Given this SDE stream, we recognise the following CEs: illegal shipping, suspicious vessel delay and vessel pursuit.

We varied the window size from 1 hour, including approximately 26K SDEs, to 24 hours, including 285K SDEs (see Figure 4(a)). The slide step $Q_i - Q_{i-1}$ is always equal to the window size. The number of recognised CEs ranges from 5K to 86K. In other words, the recognised CEs are almost two orders of magnitude more than the CEs in the activity recognition application.

Figure 4(b) shows the average CE recognition times per window when all processing threads were used by both implementations. Similar to the previous experiments, each RTEC instance was given only the SDEs of the vessels for which it performs CE recognition. Although RTEC matches the performance of dRTEC for small window sizes (1 hour and 2 hour windows), dRTEC scales much better to larger window sizes. In other words, dRTEC seems to perform much better in the presence of a large number of CEs. Figure 4(c) shows the effect of increasing the processing threads. Unlike the activity recognition application, dRTEC outperforms RTEC even when just a few processing threads are available. Similar to the activity recognition domain, dRTEC makes better use of the increasing number of threads.

[11] http://www.imo.org/OurWork/Safety/Navigation/Pages/AIS.aspx

[12] This anonymised dataset (for privacy, each vessel id has been replaced by a sequence number) is publicly available at http://chorochronos.datastories.org/?q=content/imis-3months

5 Discussion

Several techniques have been proposed in the literature for complex event processing in Big Data applications, including pattern rewriting [26], rule distribution [25], data distribution [13, 4] and parallel publish-subscribe content matching [21]. See [15, 16] for two recent surveys. Moreover, Spark Streaming has been recently used for complex event processing[13]. The key difference between our work and these approaches is the use of the Event Calculus. dRTEC inherits from RTEC the ability to represent complex temporal phenomena, explicitly represent CE intervals and thus avoid the related logical problems [23], and perform reasoning over background knowledge. This is in contrast to other complex event recognition systems, such as [8, 19], the well-known SASE engine[14] [27], and the Chronicle Recognition System [12].

Concerning the Event Calculus literature, dRTEC includes a windowing technique. On the contrary, no Event Calculus system 'forgets' or represents concisely the SDE history. Moreover, dRTEC employs a data partitioning technique using dynamic grounding and indexing. This way, dRTEC can take advantage of modern multi-core hardware. This is in contrast to Event Calculus approaches [7, 5, 24, 6, 22], where the implementations have no built-in support for distributed processing. For instance, our empirical evaluation verified that dRTEC scales better than RTEC, both to SDE streams of increasing velocity and larger numbers of CEs. Note that RTEC has proven efficient enough for a variety of real-world applications [3, 2], and already outperforms the well-known Esper engine[15] in a wide range of complex event recognition tasks [1].

Several event processing systems, such as [14, 11, 8, 10, 20], operate only under the assumption that SDEs are temporally sorted. On the contrary, dRTEC supports out-of-order SDE streams and may dynamically update the intervals of recognised CEs, or recognise new CEs, as a result of delayed SDE arrival.

The use of Spark Streaming in dRTEC facilitates the integration with other modules not implemented in Prolog, such as the computer vision modules of the SYNAISTHISI project and the trajectory compression module of datACRON. The integration of RTEC with such modules was often problematic, due to issues of the libraries integrating Prolog with other programming languages. Moreover, dRTEC

[13] `https://github.com/Stratio/Decision`

[14] `http://sase.cs.umass.edu/`

[15] `http://www.espertech.com/esper/`

avoids the memory management issues of Prolog systems that arise from continuous query computations.

For further work, we are investigating the use of a streaming infrastructure that does not rely on micro-batching (e.g. Flink[16]). Furthermore, we aim to integrate (supervised) structure learning techniques for the automated construction of Event Calculus patterns [17].

ACKNOWLEDGEMENTS

This work was funded partly by the SYNAISTHISI project, which was co-financed by the European Fund for Regional Development and from Greek National funds, and partly by the EU-funded H2020 datACRON project. We would also like to thank Elias Alevizos for his help in the empirical analysis of dRTEC.

REFERENCES

[1] E. Alevizos and A. Artikis, 'Being logical or going with the flow? A comparison of complex event processing systems', in *Proccedings of SETN*, pp. 460–474, (2014).

[2] E. Alevizos, A. Artikis, K. Patroumpas, M. Vodas, Y. Theodoridis, and N. Pelekis, 'How not to drown in a sea of information: An event recognition approach', in *IEEE International Conference on Big Data*, pp. 984–990, (2015).

[3] A. Artikis, M. Sergot, and G. Paliouras, 'An event calculus for event recognition', *Knowledge and Data Engineering, IEEE Transactions on*, **27**(4), 895–908, (2015).

[4] C. Balkesen, N. Dindar, M. Wetter, and N. Tatbul, 'Rip: Run-based intra-query parallelism for scalable complex event processing', in *Proceedings of DEBS*, (2013).

[5] I. Cervesato and A. Montanari, 'A calculus of macro-events: Progress report', in *Proceedings of TIME*, pp. 47–58, (2000).

[6] F. Chesani, P. Mello, M. Montali, and P. Torroni, 'A logic-based, reactive calculus of events', *Fundamenta Informaticae*, **105**(1-2), 135–161, (2010).

[7] L. Chittaro and A. Montanari, 'Efficient temporal reasoning in the cached event calculus', *Computational Intelligence*, **12**(3), 359–382, (1996).

[8] G. Cugola and A. Margara, 'TESLA: a formally defined event specification language', in *Proceedings of DEBS*, pp. 50–61, (2010).

[9] G. Cugola and A. Margara, 'Processing flows of information: From data stream to complex event processing', *ACM Comput. Surv.*, **44**(3), 15:1–15:62, (June 2012).

[10] N. Dindar, P. M. Fischer, M. Soner, and N. Tatbul, 'Efficiently correlating complex events over live and archived data streams', in *Proceedings of DEBS*, pp. 243–254, (2011).

[11] L. Ding, S. Chen, E. A. Rundensteiner, J. Tatemura, W.-P. Hsiung, and K. Candan, 'Runtime semantic query optimization for event stream processing', in *Proceedings of ICDE*, pp. 676–685, (2008).

[12] C. Dousson and P. Le Maigat, 'Chronicle recognition improvement using temporal focusing and hierarchisation', in *Proceedings of IJCAI*, pp. 324–329, (2007).

[13] B. Gedik, S. Schneider, M. Hirzel, and K. Wu, 'Elastic scaling for data stream processing', *IEEE Trans. Parallel Distrib. Syst.*, **25**(6), 1447–1463, (2014).

[14] D. Gyllstrom, E. Wu, H.-J. Chae, Y. Diao, P. Stahlberg, and G. Anderson, 'SASE: Complex event processing over streams', in *Proceedings of the International Conference on Innovative Data Systems Research (CIDR)*, (2007).

[15] M. Hirzel, R. Soulé, S. Schneider, B. Gedik, and R. Grimm, 'A catalog of stream processing optimizations', *ACM Comput. Surv.*, **46**(4), 46:1–46:34, (2013).

[16] Martin Hirzel, 'Partition and compose: parallel complex event processing', in *Proceedings of ACM DEBS*, pp. 191–200, (2012).

[17] N. Katzouris, A. Artikis, and G. Paliouras, 'Incremental learning of event definitions with inductive logic programming', *Machine Learning*, **100**(2-3), 555–585, (2015).

[18] R. Kowalski and M. Sergot, 'A Logic-based Calculus of Events', *New Generation Computing*, **4**(1), 67–95, (1986).

[19] J. Krämer and B. Seeger, 'Semantics and implementation of continuous sliding window queries over data streams', *ACM Transactions on Database Systems*, **34**(1), 1–49, (2009).

[20] M. Li, M. Mani, E. A. Rundensteiner, and T. Lin, 'Complex event pattern detection over streams with interval-based temporal semantics', in *Proceedings of DEBS*, pp. 291–302, (2011).

[21] A. Margara and G. Cugola, 'High-performance publish-subscribe matching using parallel hardware', *IEEE Trans. Parallel Distrib. Syst.*, **25**(1), 126–135, (2014).

[22] M. Montali, F. M. Maggi, F. Chesani, P. Mello, and W. M. P. van der Aalst, 'Monitoring business constraints with the Event Calculus', *ACM TIST*, **5**(1), (2014).

[23] A. Paschke, 'ECA-RuleML: An approach combining ECA rules with temporal interval-based KR event/action logics and transactional update logics', Technical Report 11, Technische Universität München, (2005).

[24] A. Paschke and M. Bichler, 'Knowledge representation concepts for automated SLA management', *Decision Support Systems*, **46**(1), 187–205, (2008).

[25] B. Schilling, B. Koldehofe, and K. Rothermel, 'Efficient and distributed rule placement in heavy constraint-driven event systems', in *Proceedings of IEEE HPCC*, pp. 355–364, (2011).

[26] N. P. Schultz-Møller, M. Migliavacca, and P. Pietzuch, 'Distributed complex event processing with query rewriting', in *Proceedings of DEBS*, pp. 4:1–4:12, (2009).

[27] H. Zhang, Y. Diao, and N. Immerman, 'On complexity and optimization of expensive queries in complex event processing', in *Proccedings of SIGMOD*, pp. 217–228, (2014).

[16] `https://flink.apache.org/`

Author Index