A decision making algorithm for user preferences and norms in context-aware systems

Oskar Suikki
Abstract

As context-aware systems are becoming popular in modern society, artificial intelligence will have to consider the norms of society. However, norms and personal goals can conflict with each other complicating how an artificial intelligence should act.

In this thesis I will present a decision making algorithm for evaluating contexts where the user has preferences and are affected by norms with deadlines. The norms and user preferences can have varying levels of importance. The algorithm is demonstrated with a operational example of the algorithm on a use case of planning a user’s visit to a smart city.
Acknowledgements

I would like to thank my supervisor Juan Carlos Nieves for the great feedback and help during the work of this thesis. And for the very interesting discussions regarding artificial intelligence and norms.
## Contents

1 Introduction  
1.1 Research questions  
1.2 Outline  

2 Background  
2.1 Context-awareness  
2.2 Norms  
2.3 User Preferences  
2.4 Rational Agent  
2.5 Forward-chaining planning  
2.6 Utility  

3 Related work  

4 Method  
4.1 Use case - City visit planning application  
4.2 Environment implementation  
4.2.1 User  
4.2.2 Activity  
4.2.3 Norm  
4.2.4 User Preference  
4.2.5 Environment Summary  
4.3 The rational agent  
4.3.1 BDI Model  
4.3.2 Plan  
4.3.3 Plan generation  
4.4 Filtering  

5 Context details  

5.1 City 17
5.2 User 18

6 Results 19
6.1 Norm weight: 1.0 - User preference weight: 1.0 19
6.2 Norm weight: 0.4 - User preference weight: 1.0 19
6.3 Norm weight: 1.0 - User preference weight: 0.4 20

7 Discussion 21
7.1 Plan generation complexity 21
7.2 Minor plan variations 21
7.3 Norms and user preferences value dilemma 21
7.4 Frame problem 22

8 Conclusion 23

References 23
1 Introduction

Context-aware computing is a class of information systems that use information from their physical environment to adapt their behavior, e.g., information such as location, nearby people, nearby devices etc[11]. Nowadays, different examples of context-aware systems can be found as mobile applications. Smartphone sales are consistently increasing, in 2016 nearly 1.5 billion units were sold, a 5 percent increase in total sales over 2015 [1]. Due to the mobile nature of these devices, interest in context-aware applications increase.

Context-aware systems work with the environment and often with the location of the user, it is of interest to take the norms of the location into consideration. Norms are generally defined as a way to act that it is socially acceptable; for example laws of a country, or rules that are specific for a certain location. User preferences can also conflict with norms which complicates how the system should act.

It is of interest for systems to always pick actions that satisfy norms and user preferences as much as possible. In this thesis I will present an algorithm for finding the optimal decision a system can make based on user preferences and norms. This will include implementing a context containing user preferences and norms and a rational agent that will evaluate the context.

1.1 Research questions

This thesis will cover the following two research questions:

- How can a context-aware system evaluate a context when considering user preferences, norms and norm deadlines?
- For a set of norms and user preferences of the environment with varying levels of importance, how can a context-aware system evaluate these?

By the end of this thesis a proof-of-concept of a context with norms and user preferences will be implemented, and a rational agent that acts in this context and evaluates the norms, user preferences and the deadlines. The results of this thesis can be used as a basis for decision-making in context-aware systems where multiple parameters with fuzzy values has to be evaluated. [?, ?, 6]
1.2 Outline

2. Background
   This chapter covers definitions important to this thesis. Context-awareness, norms, user preference and rational agents.

3. Related work
   This chapter describes work related to this thesis and how this thesis differs.

4. Method
   This chapter explains the method used to answer the research questions. This includes description of the use case, implementation of the context and rational agent.

5. Context details
   This chapter describes details about the context that is used in the results chapter.

6. Results
   This chapter shows the results from running the agent on the context from the previous chapter.

7. Discussion
   In this chapter, I will discuss limitations of the system and observations made during the work of this thesis.

8. Conclusion In this chapter, I sum up the work and suggest future work.
2 Background

In this chapter I will describe concepts that are important to this thesis.

2.1 Context-awareness

Context-aware systems sense and react based on their environment. For example, a mobile device could sense that the user is currently in the meeting room and is sitting down. Based on this information, a system can reject any unimportant calls to prevent disturbance at a meeting. The three important aspects of a context are: Where you are; Who you are with; What resources are nearby. However, context aware systems face possibly hard problems to solve since contexts can change over time and the acquisition of a context can be a difficult[11].

2.2 Norms

The Longman Dictionary of Contemporary English defines norms as ”the usual or normal way of doing something”[5]. Norms can be seen as rules of the environment in how someone is expected to behave. Interactions like when meeting a new person shaking hands with the person might be a norm in some environments. Country laws can also be interpreted as norms, such as following speed limits when driving. Norms are completely dependent on the context and norms have different values as in how important they are. For example, breaking the law has a bigger effect on you and the society than not shaking hands with someone you meet.

Norms can be broken up into two categories. Norms where someone is obliged to do something, and norms that permit/forbid something[8].

A formal description of norms consists of three parts[6].

- Activation condition: When the norm starts being active
- Deactivation condition: When the norm stops being active
- Maintenance condition: When the norm has been violated
2.3 User Preferences

User preferences are an important part of context-aware systems. Preference is defined as "the fact you like something or someone more than another thing or person"[13]. The user preference defines how the user wants the system to behave.

User preferences can be also formally described similarly as norms[6].

- Activation condition: When the user preference starts being active
- Deactivation condition: When the user preference stops being active
- Maintenance condition: When the user preference has been violated

2.4 Rational Agent

Acting rational means acting to achieve one’s goals, given one’s beliefs[12]. For artificial intelligence, this means using information about the world in order to achieve its goals. Preferably the agent should find a way to achieve its goals in the optimal and/or correct way. However, there are situations where there is not provably correct thing to do, yet something must be done[12].

When considering norms and user preferences this becomes complicated because who is to decide what is the correct decision? With time added to the equation the agent might have to decide between a correct or optimal solution, or something in between.

There are various different approaches for implementing rational agents. In [14], Gerhard Weiss groups rational agents into four different classes:

- logic-based agents - uses logical deduction for decision making
- reactive agents - direct mapping of situation to action
- belief-desire-intention (BDI) agents - decision making depends on manipulation of data structures representing beliefs, desires and intentions of the agent.
- layered architectures - decision making is realized via various software layers

2.5 Forward-chaining planning

Forward-chaining planning is a planning algorithm used in state space search. The algorithm starts at a initial problem state and moves forward to new states while saving the path the algorithm has moved. When the algorithm reaches a goal the saved path is a possible plan[9]. Forward-chaining planning is often optimized with the use of heuristics to guide the search[10].
2.6 Utility

In artificial intelligence a way of implementing happiness or satisfaction for the agent is by using a utility value. By mapping the outcome of actions to a real number the agent can make rational decisions for which actions have a better outcome. Utility values are calculated with a utility function, how the utility function is defined is often problem specific [12].
3 Related work

Norms have become one of the relevant areas for agent research as it helps to improve predictability of software agents the society. In [4], the principles of a normative agent is explained in how a normative agent needs to understand norms of the society and be able to deliberately follow or violate these norms.

There are various different approaches to representing norms and normative agents in practice. In [8] an implementation based on Situation Calculus is presented; and, in [3] Drools is used to implement context and norms. These papers are focused on detailed representations of norms. This thesis will use a simplified representation of norms since I am exploring the conflict between user preferences and norms.

In [2] and [6], different approaches to handling conflicts between norms and personal goals are presented. The authors of [2] expand the BDI architecture with sacrifice and diligence functions. This paper also deals with norms with time-constraints. However, there is a focus on accomplishing a norm within a timelimit by sacrificing or increasing effort on other conflicting tasks. In this thesis, I will present a method that will try to optimize time usage for both personal goals and norms with the use of values and a utility function.

The authors of [6] presents a model for norms and user preferences as well as a basis for a system that plans and filters. I will expand upon [6] in this thesis with an operational implementation using the model for norms and user preferences presented.
4 Method

To answer the research questions I will implement a use case that the system will evaluate. In this chapter I will explain the use case and important parts of the implementation.

4.1 Use case - City visit planning application

The use case for this thesis is a system that creates a plan for the user when visiting a city. The city consists of different activities that the user can participate in, for example going to a restaurant or watch a movie at the cinema. The user has some desires that he/she wants to carry out in the city and the application will find the best schedule to do these activities. The system is provided with all the activities in the city as well as the possible times to do these activities.

The user of the application has a profile which contains preferences of different activities. The system will try to prioritize these preferences if possible. However the user can also be affected by norms that needs to be satisfied within a timelimit. This can be appointments such as a visit to the hospital.

This use case is based on the concept of smart cities. A vision to integrate information and communication technology and internet of things to manage city assets.[7]

4.2 Environment implementation

This section will describe the data analysis of the environment the rational agent will act upon.

4.2.1 User

The user class represents the profile of the user. This includes user’s preferences about different city activities, norms the user is affected by as well as the activities the user wants to do. The user desires are satisfied by doing activities that satisfy a certain desire.

- name - The name of the user
- preferences - A list of the user’s preferences
- activeNorms - A list of norms currently affecting the user
- desires - A list of things the user wants to do in the city
4.2.2 Activity

The activity class represents something that can be done in the city. For example, eating at a certain restaurant. A activity has a name, a desire it satisfies, the time it takes to carry out the activity and the time the activity can be done. An activity takes a minimum of one hour. Multiple activities can satisfy the same desire, for example all kinds of restaurants can satisfy hunger.

- name - The name of the activity
- satisfies - The desire the activity satisfies
- timeConsumation - The amount of time the activity takes to carry out
- availableTimes - The available times to do this activity

4.2.3 Norm

Norms sets an obligation on a user to perform a specific activity within a certain time. Every norm has an importance value assigned. The value is a decimal value between 0 and 1. The norm consists of the following variables:

- id - An identifier for the norm
- condition - The condition that needs to be met in order to fulfill the norm. This will be the name of the activity that has to be done.
- activateTime - The time where the norm activates
- deactivateTime - The time where the norm deactivates
- value - The importance value of the norm
- fulfilled - If the norm has been fulfilled.

4.2.4 User Preference

The user preference represents what the user thinks of a certain activity. Each preference is valued with a decimal number between 0 and 1.

- id - User preference id
- condition - The condition for the user preference. This will be the name of the activity that triggers this user preference.
- fulfilled - If the user preference is fulfilled
- value - The value of the preference
4.2.5 Environment Summary

The environment consists primarily of the user and city. The user has desires and preferences while being affected by norms. The city consists of all the possible activities to carry out. These activities will satisfy desires and fulfill norms and user preferences. The relation between each of these classes is shown in figure 1.

Figure 1: The relations between different classes in the environment
4.3 The rational agent

This section will cover the details of the rational agent implementation. The agent will evaluate the context and present the optimal plan for the user.

4.3.1 BDI Model

This agent uses a characterization of the Belief Desire Intentions (BDI) model as a practical reasoning algorithm. The BDI model is chosen because for a context-aware system it makes sense in the way it gathers and processes information about the world. The BDI model consists of three main parts.

**Beliefs** - The information state of the agent. This expresses what the agent knows about the world.

**Desires** - The goals that the agent wants to carry out.

**Intentions** - What the agent has chosen to do.

The following is a overview of the steps the agent will act in:

```plaintext
function BDI():
    user = get_user() //get desires/beliefs
    city = get_city() //get beliefs
    plans = generate_plans(user, city)
    optimal_plan = filter_plans(plans)
```

The goal of this agent is to deliver optimal plans for the user. As seen in the BDI function, after getting the desires and beliefs the agent will generate plans that will achieve these desires. However in order to find the optimal plans, the agent has to evaluate every generated plan. The process of generation and filtering is explained more in detail in the sections 4.3.3 and 4.3.4.

4.3.2 Plan

A plan consists an array of activities to carry out at each time. As seen in Figure 2, each slot in the array refers to a certain time where either the name of an activity can be put or "FREE" to represent an open slot. The plan spans a maximum of 6 hours where each activity takes a minimum of 1 hour.

The plan implementation contains several other variables necessary for the planning and filtering process:

- **activities** - The list of activities in a chronological order.
- **norms** - The norms that will be evaluated for this plan.
- **userPreferences** - The user preferences that will be evaluated for this plan.
- **unsatisfiedDesires** - The desires the user want to satisfy. The planning process will try to empty this list.
- **utility** - The utility value for the plan.
4.3.3 Plan generation

To find the optimal plan it is necessary to generate every possible plan and evaluate them. To generate plans a type of forward-chaining is used. Forward-chaining was chosen as it was the easiest way to solve the plan generation problem within the time span of this thesis.

For each desire, I generate plans for all possible combinations of activities relevant to the user. This means every activity that satisfies either user desire or a norm that the user is affected by, this stack of desires and norms will be called goals.

The plan generation algorithm is divided into two parts. The first part where a plan is created for every possible time a goal can be achieved. The goal is removed from the goal stack and saved in the plan. This part of the algorithm is as follows:

```
function generatePlans(user) :
    goals = all user desires and norms
    numberOfGoals = goals.size()
    for every goal:
        goalActivities = find all activities that satisfy goal
        for every activity in goalActivity:
            for every available time of the activity:
                create a new plan
                add activity to plan at the corresponding time in activity
                set the remaining goals of the plan to goals
                remove the current goal from the plan's goal stack.
                add the plan to the list of plans
```
numberOfGoals = numberOfGoals - 1

After the initial plans are made the algorithm will iterate through every goal creating a new plans if possible. This is done with the following function:

```python
function generatePlansByGoals(currentPlans) :
    for every plan in currentPlans
        for every goal in the plan
            goalActivities = find all activities that satisfy goal
            for every activity in goalActivity:
                for every available time of the activity:
                    add activity to plan at the corresponding time in activity
                    add plan to newPlans if no duplicates exist
    return newPlans
```

The complete plan generation code is as follows:

```python
function generatePlans(user) :
    goals = all user desires and norms
    numberOfGoals = goals.size()
    for every goal:
        goalActivities = find all activities that satisfy goal
        for every activity in goalActivity:
            for every available time of the activity:
                create a new plan
                add activity to plan at the corresponding time in activity
                set the remaining goals of the plan to goals
                remove the current goal from the plan’s goal stack.
                add the plan to the list of plans
    numberOfGoals = numberOfGoals - 1

    while numberOfGoals is greater than 0:
        plans = generatePlansByGoals(plans)
        numberOfGoals = numberOfGoals - 1
    return plans
```

### 4.4 Filtering

Finding the optimal plans is accomplished by calculating a utility value for every generated plan. The utility function used is Each plan contains a set of norms and user preferences. Let $P = ([N_1, N_2, \ldots, N_n], [S_1, S_2, \ldots, S_n])$ represent the set of norms and user preferences in a plan. The utility values for the norms and user preferences are calculated first, this is accomplished with the following two formulas:

$$U(P_N) = \sum_{N \in P} value(N) \quad (4.1)$$
$$U(P_S) = \sum_{S \in P} value(S) \quad (4.2)$$
The final utility value is calculated by adding the values from the two previous formulas 4.1 and 4.2. Each value will have a weight of a positive value assigned. These weights parameters will determine how much the agent should value user preferences or norms. Formula 4.3 shows the final where $w_1$ and $w_2$ represents the weights

$$U(P) = w_1 U(P_N) + w_2 U(P_S)$$

(4.3)
5 Context details

This chapter explains the details of the context the rational agent evaluates.

5.1 City

The city is constructed as a list of activities. Table 1 shows the activities in this context. Some activities are only available at certain times, this is shown in Table 2.

Table 1 All activities in a city, the desires they satisfy and how much time carrying out an activity takes

<table>
<thead>
<tr>
<th>Activity Name</th>
<th>Satisfies</th>
<th>Time consumption</th>
</tr>
</thead>
<tbody>
<tr>
<td>Museum</td>
<td>Museum</td>
<td>2</td>
</tr>
<tr>
<td>Sushi</td>
<td>Food</td>
<td>1</td>
</tr>
<tr>
<td>Grill</td>
<td>Food</td>
<td>1</td>
</tr>
<tr>
<td>Cinema_Act</td>
<td>Movie</td>
<td>3</td>
</tr>
<tr>
<td>Cinema_Rom</td>
<td>Movie</td>
<td>3</td>
</tr>
<tr>
<td>Walk_Dog</td>
<td>Dog</td>
<td>2</td>
</tr>
<tr>
<td>Postalservice</td>
<td>Postal</td>
<td>1</td>
</tr>
<tr>
<td>Friends</td>
<td>Social</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 2 The available times for time specific activities. All activities not included in this table can be carried out at any time.

<table>
<thead>
<tr>
<th>Time</th>
<th>Postalservice</th>
<th>Friends</th>
<th>Cinema_Act</th>
<th>Cinema_Rom</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Closed</td>
<td>Open</td>
<td>Open</td>
<td>Open</td>
</tr>
<tr>
<td>2</td>
<td>Open</td>
<td>Closed</td>
<td>Closed</td>
<td>Closed</td>
</tr>
<tr>
<td>3</td>
<td>Open</td>
<td>Open</td>
<td>Open</td>
<td>Open</td>
</tr>
<tr>
<td>4</td>
<td>Open</td>
<td>Closed</td>
<td>Closed</td>
<td>Closed</td>
</tr>
<tr>
<td>5</td>
<td>Closed</td>
<td>Open</td>
<td>Closed</td>
<td>Closed</td>
</tr>
<tr>
<td>6</td>
<td>Closed</td>
<td>Closed</td>
<td>Closed</td>
<td>Closed</td>
</tr>
</tbody>
</table>
5.2 User

The user has a list of desires as seen in table 3. This is a list of things the user wants to do in the city. The user preferences as seen in table 4 specify how the user value different activities in the city. Table 5 shows the norms the user is affected by.

Table 3 The user’s desires. The user wants to eat something, go to the museum, go watch a movie and hang out with friends

<table>
<thead>
<tr>
<th>Desires</th>
</tr>
</thead>
<tbody>
<tr>
<td>Food</td>
</tr>
<tr>
<td>Museum</td>
</tr>
<tr>
<td>Movie</td>
</tr>
<tr>
<td>Social</td>
</tr>
</tbody>
</table>

Table 4 The user preference for the user

<table>
<thead>
<tr>
<th>ID</th>
<th>Condition</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Food_Meat</td>
<td>Grill</td>
<td>0.3</td>
</tr>
<tr>
<td>Food_Sushi</td>
<td>Sushi</td>
<td>0.6</td>
</tr>
<tr>
<td>Museum</td>
<td>Museum</td>
<td>0.4</td>
</tr>
<tr>
<td>Movie_Action</td>
<td>Cinema_Act</td>
<td>0.4</td>
</tr>
<tr>
<td>Movie_Romance</td>
<td>Cinema_Rom</td>
<td>3</td>
</tr>
<tr>
<td>Hangout</td>
<td>Friends</td>
<td>0.6</td>
</tr>
</tbody>
</table>

Table 5 The norms for the user. The user has a booked hairdresser appointment, a dog the user needs to walk and a package that needs to be sent

<table>
<thead>
<tr>
<th>ID</th>
<th>Condition</th>
<th>Value</th>
<th>Activation time</th>
<th>Deactivation time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Booked_Hairdresser</td>
<td>Hairdresser</td>
<td>0.6</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>Walk_Dog</td>
<td>Walk_Dog</td>
<td>0.75</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Send_Package</td>
<td>Postalservice</td>
<td>0.2</td>
<td>1</td>
<td>6</td>
</tr>
</tbody>
</table>
6 Results

In this chapter I will present results of the agent evaluating a context. The agent will evaluate the context described in the previous chapter three times with different values assigned to the utility function weights each time.

6.1 Norm weight: 1.0 - User preference weight: 1.0

With both weights set to 1 the agent finds whatever plan that maximizes the utility. Figure 3 shows the plan, norms and unsatisfied desires. The Send Package norm ends up unfulfilled because of its low value. And the two highest user preferences Sushi and Friends is picked.

![Generating plans...]

Optimal plan: 2.45
Desires/ Norms:
1: MUSEUM
2: MOVIE
3: POSTAL SERVICE

Norms:
BOOKED HAIRDRESS : fulfilled
WALK DOG : fulfilled
SEND PACKAGE : unfilled

Plan
1: SUSHI
2: WALK DOG
3: WALK DOG
4: HAIRDRESSER
5: FRIENDS
6: FRIENDS

Figure 3: The optimal plan when favoring neither norms or user preferences

6.2 Norm weight: 0.4 - User preference weight: 1.0

In this case user preferences are favored over norms. Comparing with the results from the previous section, the agent decides to skip the Walk Dog norm for going to the museum. The hairdresser still ends up in the plan, this is because there is no other activities that match the desires at that time. The results are shown in figure 4.
6.3 Norm weight: 1.0 - User preference weight: 0.4

In the final case the agent favors norms over user preferences. As seen in figure 5 the agent fulfills all norms by suggesting going to the postalservice instead of socializing with friends. Because of this, timeslot 6 in the plan ends up free because nothing can be done at that time.

Generating plans...
Done generating
Generated: 2206 plans
Norm weight: 1.0 | User Pref weight: 0.4

Optimal plan: 1.79
Unsatisfied Desires/Norms:
1: MUSEUM
2: MOVIE
3: SOCIAL

Norms:
BOOKED_HAIRDRESS : fulfilled
WALK DOG : fulfilled
SEND PACKAGE : fulfilled

Plan
1: SUSHI
2: WALK DOG
3: WALK DOG
4: HAIRDRESSER
5: POSTAL SERVICE
6: FREE

Figure 5: The optimal plan when favoring norms
7 Discussion

In this chapter I will discuss the results from the previous chapter

7.1 Plan generation complexity

To generate plans a type of forward chaining was used. This will generate all possible plans in the context and combined with the utility function it is certain the optimal plan will be found.

However the big downside to this method is its time complexity. There are multiple factors in the use case that will cause the amount of possible plans to increase by a large amount. When testing the system I found the plan generation time to increase by an exponential amount when adding more desires or norms to the user.

A important detail in the plan generation algorithm is that plans should only be added if no duplicates exist. However depending on the complexity of a plan it can also be time consuming to compare plans. But in our implementation this was a fast operation because of the limited version of the use case.

A faster variant of plan generation could be by using the values assigned to norms and user preferences in a heuristic search, however this method might not find the optimal plan because of the issue of knowing if you should keep searching or not.

7.2 Minor plan variations

Because of how the plan generation creates all possible combinations of activities and their times the system will most likely end up with multiple plans with the same activities and utility value but with the activities in different order in the plan. How to handle this is up to the system developer since it might be of interest to have every possible variation of a plan. But if the scope of a plan is large there will be a large amount of variations that the system will have to handle.

7.3 Norms and user preferences value dilemma

There is a dilemma for norms and user preferences that appears when assigning values to them. Do you pick one norm with high value or multiple norms that add up to the same value? The reasoning algorithm presented in this thesis does not handle this issue. If this
situation were to occur both plans would tie at the top. The system developer would have to take the decision if the system should select plans with few satisfied norms (the high value norm plan) or the plan with many satisfied norms.

### 7.4 Frame problem

The frame problem expresses the difficulty of expressing facts about the world to an artificial intelligence[12]. While the term originated from using first-order logic to express facts, it has become used in general conversations about artificial intelligence and the difficulties of knowledge representation.

When implementing norms and contexts this becomes an issue of how detailed you need to describe facts. For this thesis I had to make a lot of assumptions about the different activities, in a real world implementation of such a system every activity could have unique features that would be necessary to take into account in order for the system to fully evaluate the context.
8 Conclusion

While the reasoning algorithm presented presents a optimal plan there is a lot of limitations to the algorithm as discussed in chapter 7. Further improvements to the plan generation can be done, in particular using heuristics could be a good approach because of the use of norm and user preference values. Because of inefficiency of the plan generation it is difficult to test the system on contexts with large number of norms and user preferences.

The authors of *Resolving conflicts between personal and normative goals in normative agent systems* [2] present a different method of handling user preferences and norms with a deadline. The deadline used in the study is different from this thesis which makes it difficult to compare the results of both approaches. Due to lack of time I was not able to implement the same context, but it would be interesting to see how both approaches compare.

In this thesis I presented a simple and general implementation of norms and user preferences. These can be used for other contexts or used as a basis for contexts that require more specific conditions. While the rational agent is not general, the only context specific parts in the BDI model is the data representation and plan generation. Because of the use of utility values that are calculated with norms and user preferences, the filtering process is easy to adapt on other types of plans as long as they are compatible with norms and user preferences.
References


