Improving the Contextual Selection of BDI Plans by Incorporating Situated Experiments

Ahmed-Chawki Chaouche[1,2], Amal El Fallah Seghrouchni[1], Jean-Michel Ilié[1], and Djamel Eddine Saïdouni[2]

[1] LIP6, UMR 7606 UPMC - CNRS, 4 place jussieu 75005 Paris, France
{ahmed.chaouche,amal.elfallah,jean-michel.ilie}@lip6.fr
[2] MISC, University Constantine 2, Ali Mendjeli Campus, 25000 Constantine, Algeria
saidouni@misc-umc.org

Abstract. This paper is dedicated to intentional BDI agents evolving in ambient environment. The planning management framework we propose, looks for efficient guidance to improve the satisfaction of the agent's intentions with respect to the possible concurrent plans and the current context of the agent. Adopting the idea that "location" and "time" are key stones information in the activity of the agent, we show how to enforce guidance by ordering the different possible plans. As a major contribution, we demonstrate two original utility functions that are designed from the past-experiences of the action executions, and that can be combined accordingly to the current balance policy of the agent.

Keywords: Ambient agent, BDI, plan selection, contextual guidance, past-experiences.

1 Introduction

Designing and developing applications for Ambient Intelligence (AmI) systems involves different challenges in several computing areas as well as intelligent systems research, sensor networks, mobile technologies and interaction within user centered design. Multi-Agent Systems (MAS) are a powerful approach to design AmI systems to function potentially within dynamic environments. They offer interesting frameworks and support for autonomy, proactivity, societal cooperation and context-awareness needed for such systems.

To deal with MAS, we need to capture the desired properties in the right way as well as modeling agent's abilities and functionalities. The Belief-Desire-Intention (BDI) model can flexibly handle the entire modeling and developing of rational agents in AmI systems, based on well-structured and goal-driven aspects, e.g. [1,2].

However, some AmI applications are really complex [3]. The environment can be open so that the agents may enter and leave. Moreover, agents can move between locations. In fact, the agents must adapt their decisions and behavior to cope with dynamic and unexpected changes of the context. It is necessary to

R. Chbeir et al. (Eds.): AIAI 2015, IFIP AICT 458, pp. 266–281, 2015.
DOI: 10.1007/978-3-319-23868-5_19

increase the agent reactiveness and consistency, contributing to better flexibility and resilience properties.

In [1], new possibilities of modeling AmI systems are presented, not only by referring to MAS and BDI agents, but also by embedding a formal and automated planning management mechanism. Plans are consistent w.r.t. the intentions of the agent and can be achieved concurrently. Based on an underlying model of actions and processes, a contextual planning system (CPS) was demonstrated, that can be exploited to maximize the satisfaction of intentions, taking the spatial information into account.

However, we argue that the former mechanism, which looks for guidance, can be improved by a learning from the past-experiences of actions. For instance, taking into account the failure of actions can be used to evaluate a pertinence of some plan wherein this action occurs, e.g. [4]. Closer to our aim, the work of [5] evaluates the contribution of each plan in terms of utility and preference, seen as parameters for the optimization.

The utility functions we introduce evaluate plans on safety and speed notions. Both can be based on concrete information issued from the executions of actions. In particular, safety criteria is based on the outcomes of the past-experiences whereas speed criteria is learned from the durations of these experiences. We need to define a reinforcement technique adapted to AmI agents and the unexpected changes of their contexts. In particular, our approach to predict the outcome and the duration of some action in a plan does not rely on recurrent learning principles but rather on a systematic analysis of some pertinent situated past-experiences.

In this paper, we present a method to organize the filtering of past-experiences according to spatio-temporal strategies. In fact, location is a very fundamental context information to evaluate an action but time is another important data. For example, it is not really interesting to take the highway, at the time it is crowded. This can be captured by a study of the duration of previous transportations via the highway, made at some time during the business day.

The remaining of the paper is organized as follows: Section 2 describes the functional architecture of our planning process, together with its guidance mechanism based on the capability of an agent to reason on the future actions, on the basis of its knowledge about past experiences. Section 3 shows how the possible plans to satisfy the intentions of the agent can be expressed compactly with the AgLOTOS specification language. From such an expression, the derived CPS structure is built automatically to process the agent guidance. Section 4 details the acquisition of past-experiences and their exploitation. In Section 5, the spatio-temporal guidance mechanism is presented, based on the CPS information enriched by the past-experiences of actions. Section 6 presents related work, then Section 7 concludes our paper.

2 Contextual Planning Architecture

The overall agent architecture, we deal with, is standardly based on several processes. Triggered by some *events* expressing the context changes, the *mental*

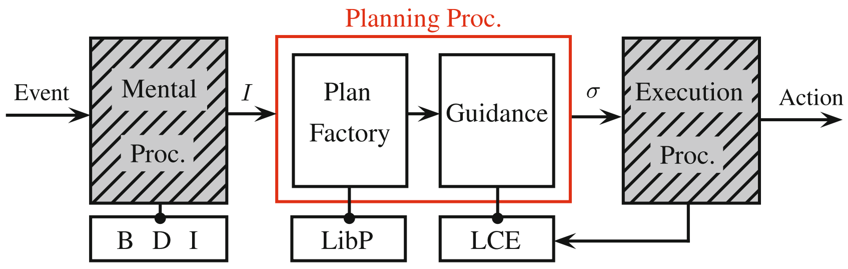

Fig. 1. Extended BDI model with contextual planning architecture

process is assumed to be highly deductive and rational, based on BDI structures (Beliefs, Desires and Intentions).

Figure 1 brings out a functional view of our *planning process*, in charge to offer some plan to execute (σ) from the current set of intentions (I). It is composed of two modules:

– The *Plan Factory* builds the so-called *agent plan* which globally corresponds to the current set of intentions. The *LibP library* is used to exploit the different alternative of plans that must be specified for each intention. With respect to the current spatial context of the agent, a CPS is built, made of the different allowed interleaving of the actions in plans. By its traces, the CPS represents all the possible plans that can be realized.
– The *Guidance module* is able to enrich the CPS information taking profit from the past-experiences of actions, that are stored in the so-called *Learned Contextual Experiences (LCE) module*. This results in a structure called CPS with learning (CPS-L for short), which can be used to searching the optimal trace w.r.t. the spatio-temporal context of the agent.

Our approach to specify the agent behavior is strongly related to the fact that the mental process can proactively decide to revise its set of intentions, at real time. Let us illustrate two nominal cases [1]:

– the agent can decide to investigate the first action of one of the best possible plans offered by the planning process, but can decide to revise some of the intentions, in case there is no plan to select, in respect to the current context.
– face to unexpected changes of context, the agent can update some of its intentions. This causes the revising of a part of the agent plan automatically.

3 From Concurrent Planning to Contextual Guidance

3.1 Agent Plan Structure

Figure 2 highlights the tree plan structure of the *agent plan* ($\overline{P}$). At the second level, each *intention plan* corresponds to the achievement of one intention. At the

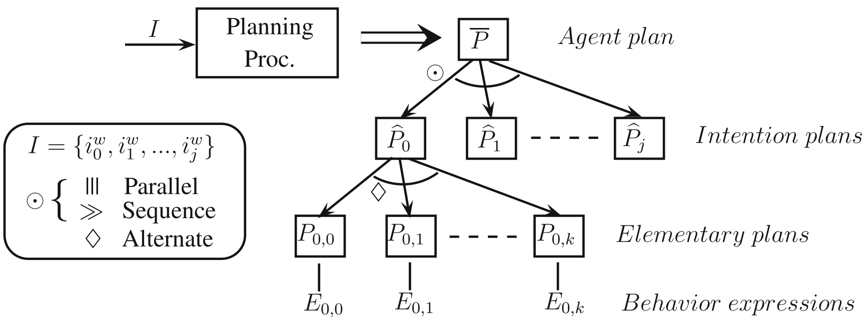

Fig. 2. Agent plan structure

last level, elementary plans are those and only those obtained from the *library of plans* (*LibP*).

The plan structure is formally captured by using a formal description language, namely AgLOTOS. It introduces modularity and concurrency aspects to compose and schedule different sub-plans, viewed as processes [1]. Let us now briefly recall the AgLOTOS-based specifications for plans.

Agent Plan Level. The set of intention plans can be globally handled by using the concurrent $|||$ or sequential $\gg$ operators between intention plans, leading to the specification of an agent plan. Let $\overline{\mathcal{P}}$ be the set of names qualifying the possible agent plans with $\overline{P} \in \overline{\mathcal{P}}$ and $\widehat{\mathcal{P}}$ be the set of names used to identify the possible intention plans with $\widehat{P} \in \widehat{\mathcal{P}}$, such that $\overline{P}$ is any agent plan defined by:

$$\overline{P} \ ::= \ \widehat{P} \ \mid \ \overline{P} \, ||| \, \overline{P} \ \mid \ \overline{P} \gg \overline{P}$$

Intention Plan Level. An intention plan can be expressed by alternative elementary plans (P), such that each one can be executed to achieve the corresponding intention. This is captured in AgLOTOS by using the composition operator $\Diamond$. In particular, an intention is satisfied if and only if at least one of the associated elementary plans is successfully executed. Formally, let $\mathcal{P}$ be the set of names qualifying the possible elementary plans with $P \in \mathcal{P}$. We define an intention plan $\widehat{P}$ as:

$$\widehat{P} \ ::= \ P \ \mid \ \widehat{P} \Diamond \widehat{P}$$

Elementary Plan Level. Elementary plans are described by *AgLOTOS expressions*, referring to a finite set of observable actions. Any AgLOTOS expression is associated with contextual information related to the (current) BDI state of an agent. For that, let Θ be a finite set of space locations where an agent can move and Λ be the set of agents with which he can communicate. Let $\mathcal{O}$ be the (finite) set of observable actions which are viewed as instantiated predicates, ranging over $a, b, ...$

Let $Act = \mathcal{O} \cup \{\tau\}$, be the set of actions, where τ is the internal action. In the context of this paper, the internal actions mainly correspond to the information related to the termination of a plan. The AgLOTOS language specifies pairs for each elementary plan composed of an identifier and an AgLOTOS expression to specify its behavior. The syntax of an elementary plan P is inductively defined as follows:

$$\begin{aligned} P &::= E \\ E &::= exit \mid a; E \qquad (a \in \mathcal{O}) \end{aligned}$$

In this syntax, $P ::= E$ represents an elementary plan identified by P, such that its behavior expression is E. The expression $a; E$ denotes an action a prefixing E, and the elementary expression *exit* expresses the successful termination of some plan. For sake of concision, the other operators defined in AgLOTOS are not detailed in this paper. Please refer to [1] for more details, however, let us note that elementary plans do not generate new intentions, as it is the case for some other BDI approaches.

Building of Agent Plan from Intentions. In order to account for any BDI state of the agent, we propose that the agent can label the different elements of the set I of intentions[1] by using a weight function $weight : I \longrightarrow \mathbb{N}$. The ones having the same weight are composed by using the concurrent parallel operator $|||$. In contrast, the intention plans corresponding to distinct weights are ordered by using the sequential operator $\gg$.

For instance, let $I = \{i_g^2, i_e^1, i_m^2\}$ be the considered set of intentions, such that the superscript information denotes a weight value and $\widehat{P_g}, \widehat{P_e}, \widehat{P_m}$ respectively correspond to their intention plans. The constructed agent plan could be viewed as: $\overline{P} ::= (\widehat{P_g}|||\widehat{P_m}) \gg \widehat{P_e}$. In terms of elementary plans, by considering that $\widehat{P_g} ::= P_g$, $\widehat{P_m} ::= P_m$ and $\widehat{P_e} ::= P_{e1} \Diamond P_{e2}$, then the agent plan is refined in $\overline{P} ::= (P_g|||P_m) \gg (P_{e1} \Diamond P_{e2})$.

3.2 Contextual Planning System (CPS)

With respect to some agent plan $\overline{P}$, we introduce a notion of configuration, denoted $[\overline{P}]$, corresponding to the planning state of the agent. This helps specifying the evolution of the agent plan. Some examples are given further. Let usl now define the contextual planning state of the agent, taking into account the location of the agent and the outcomes of different intention plans defined for this agent.

Definition 1. *A contextual planning state is a tuple (ps, ℓ, T), where ps is any planning state described by an agent plan configuration $[\overline{P}]$, ℓ corresponds to the possible location for the agent in this state, and T is the subset of intention plans that terminate successfully in this state.*

[1] We assume that the BDI agent itself can solve conflicting situations that could arise between intentions for some context, by means of a scheduling process applied to the set of intentions.

The AgLOTOS operational semantics is used to specify the contextual possible planning state changes for the agent. In this paper, it is applied to produce a Contextual Planning transition System, called *CPS*, from an initial contextual planning state, e.g. $(ps, \ell, \emptyset)$, meaning that the agent is initially at location ℓ and ps is its planning state.

Definition 2. *The Contextual Planning System (CPS) is a labeled Kripke structure* $\langle S, s_0, Tr, \mathcal{L}, \mathcal{T} \rangle$ *where:*

- S *is the set of contextual planning (CPS) states,*
- $s_0 = (ps, \ell, \emptyset) \in S$ *is the initial CPS state of the agent,*
- $Tr \subseteq S \times Act \times S$ *is the set of transitions. The transitions are denoted* $s \xrightarrow{a} s'$ *s.t.* $s, s' \in S$ *and* $a \in Act$,
- $\mathcal{L} : S \to \Theta$ *is the location labeling function,*
- $\mathcal{T} : S \to 2^{\widehat{\mathcal{P}}}$ *is the intention termination labeling function which captures the intention plans that have been completed.*

3.3 Illustrative Example

Let us consider two agents, Alice and Bob. The scenarios of Alice and Bob are specified separately. It is assumed that Bob and Alice may coordinate in order to achieve their intentions, at their mental process levels. The actions in plans are simply expressed using instantiated predicates, like $getc(\ell_2)$ for the 'get copies' action. Intention plans are composed of elementary plans which are viewed as concurrent processes, terminated by *exit*, a la LOTOS. For instance, the set $I_B = \{meeting(Alice, \ell_1), getting\_copies(\ell_2)\}$ is defined for Bob, containing two concurrent intentions of the same weight.

The associated agent plan is: $\overline{P_B} ::= (\widehat{P_g}|||\widehat{P_m})$. The intention plan $\widehat{P_m}$ is used to satisfy the intention $meeting(Alice, \ell_1)$ and has only one elementary plan $P_m ::= move(\ell_1); meet(Alice); exit_m$. The second intention plan $\widehat{P_m}$ corresponds to the intention $getting\_copies(\ell_2)$ and also has one elementary plan $P_g ::= getc(\ell_2); confirm; exit_g$.

The initial agent plan configuration is directly given from the above here plan description, $[\overline{P_B}] = (move(\ell_1); meet(Alice); exit_m)|||getc(\ell_2); confirm; exit_g$. One of the possible derivations from this configuration consists in performing the *getc* action. So, the agent plan configuration of Bob changes to $move(\ell_1);\ meet(Alice); exit_m\ |||\ confirm; exit_g$.

The Contextual Planning System of Bob, denoted CPS_B, is illustrated in Figure 3. It is built from the initial CPS state, $s_0 = ([\overline{P_B}], \ell_2, \emptyset)$, taking into account the current location ℓ_2 of Bob. It is worth noting that the CPS could grow exponentially depending on the current intentions of the agent, hence on the related actions in plans. Nevertheless, we consider that the number of intentions to be dealt with dynamically is reasonably small, moreover the CPS structure is only partial since actions are only allowed w.r.t. the current spatio-temporal context of the agent[2].

[2] Intuitively speaking, the achievements of intentions are more or less costly depending on the plans they involve.

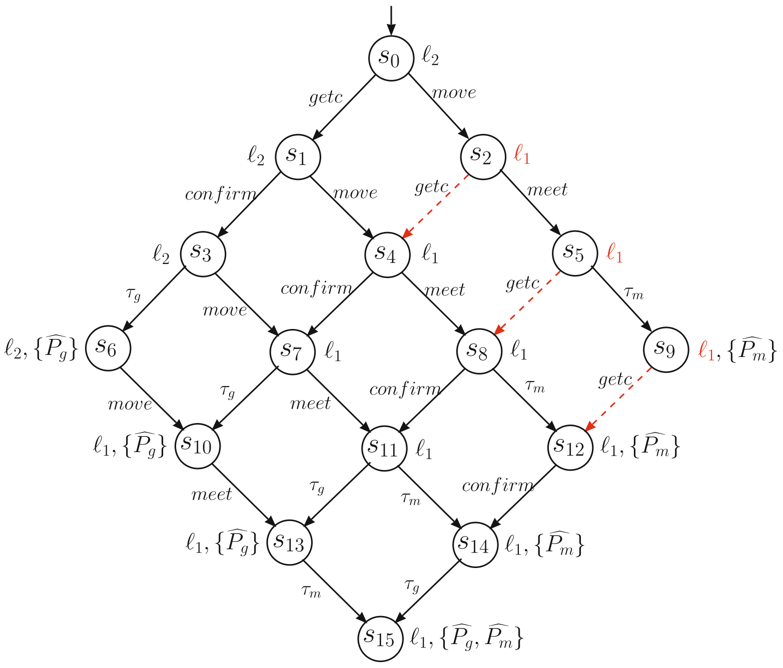

Fig. 3. The CPS_B corresponding to the plan $\overline{P_B}$

3.4 Guidance for Intention Satisfaction

In a CPS, any transition $s \xrightarrow{a} s$ represent actions to be performed. Like in the STRIPS description language [6], actions are modeled by instantiated predicates defined with preconditions and effects. In this paper, the preconditions concern only the contextual information attached to the source state. Let $pre(a)$ be the precondition of any action a, e.g. $pre(a(\ell)) = \ell = \mathcal{L}(s)$. For instance in Figure 3, the dashed edges represent the disabled transitions from the states $s \in \{s_2, s_5, s_8\}$, because $pre(getc(\ell_2)) = \ell_2 \neq \mathcal{L}(s)$.

In order to guide the agent, the planning process can select an execution trace which maximizes the number of intentions that can be satisfied. This can be captured over the set $\Sigma \subseteq 2^{Tr}$ of all possible traces of the CPS. We introduce the notion of *maximum trace* based on the mapping $end : \Sigma \longrightarrow 2^{\widehat{\mathcal{P}}}$, used to specify the set $end(\sigma)$ of the different terminated intention plans that occur in a trace $\sigma \in \Sigma$. Let Σ_{MAX} represent the set of maximum traces of the CPS.

As a specific tree structure, Figure 6 represents the 10 maximum traces of CPS_B. In this unfolded version of Figure 3, the trace carried out by $s_0 \to s_2 \to s_5 \to s_9$, is not represented because it is not a maximum trace.

4 Learning Actions from Past-Experiences

In order to improve the agent runtime performance, the planning process, in relation to the execution process, can capture the action execution in a given context (current location and current time). The use of the learned data in a pertinent way allows us to build an enriched CPS structure, called *CPS with Learning* (*CPS-L* for short). As a result, the best possible strategic decisions are taken, driven by the agent and its preferences.

4.1 Data Acquisition

The performance of an action a is evaluated w.r.t. a given context, however we focus on location. Each concrete performance of a in some location is considered as an *(action) experience.*

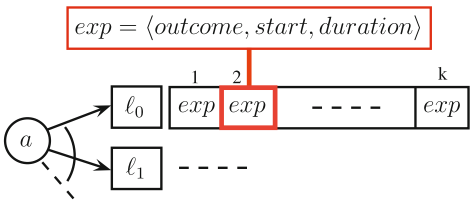

Fig. 4. Learned Contextual Experiences (LCE) of an action a

Definition 3. *An experience exp of an action a in a location ℓ is a tuple $\langle outcome, start, duration \rangle$, where*

- *$outcome \in \{-1, 1\}$, is the result of the run performing a, respectively a failure or a success,*
- *$start \in \mathbb{R}^+$, is the start date for the run of a,*
- *$duration \in \mathbb{R}^+$, is the action duration which provides that a was successfully terminated, undefined otherwise.*

The structure represented in Figure 4 is generically called the *Learned Contextual Experiences* (*LCE*). Regarding to the action a, LCE_a shows different FIFO queues of experiences of a, distinguishing the different locations where the action was performed. The queue $LCE_a(\ell)$ is ordered by the *start* date of the runs, so that the last recorded experience is at the top of the queue.

More precisely, if an action a is performed in some location ℓ with a certain experience exp, the agent may push it s.t. $LCE_a(\ell) = LCE_a(\ell) \cup \{exp\}$. Moreover, k represents the effective size of $LCE_a(\ell)$. Regarding to any experience exp of a queue $LCE_a(\ell)$, we denote by $index(exp)$ the position of exp in the queue. Further, the three components of a past-experience exp are respectively denoted $exp.outcome$, $exp.start$, $exp.duration$.

4.2 Data Relevance Strategies

The strategy information specified by the agent is given by the following definition and described below.

Definition 4. *The queues of LCE are parameterized by the agent strategy* $\mathcal{S} = \langle K, forget, M, \mathcal{C} \rangle$ *where,*

- K *is the maximum size of the queue,*
- $forget : 1..K \rightarrow \mathbb{R}^+$ *is the forgetting function, yielding a relevance weight for each experience,*
- $M \leq K$ *is the maximum number of filtered experiences,*
- $\mathcal{C}$ *is a periodic classification, e.g. daily, weekly, monthly or annual, applied in a modulo operation over the start dates of the queue,* $filter$*, defined over any queue, is a time filtering function yielding a sub-queue according to* M *and* $\mathcal{C}$*,*

Table 1. LCE of the action *getc* in the location ℓ_2

index	1	5	6		10	11	12		18	*k*
$LCE_{getc}(\ell_2)$	...	A	B	...	C	D	E	...	F	...

$filter_{6,daily}(10)$

Experiences	*outcome*	$mod_{daily}(start)$	*duration*	*index*	$forget(index)$
A	-1	8.29	-	5	0.20
B	1	9.50	3.00	6	0.16
C	1	9.05	2.10	10	0.10
D	1	10.78	3.40	11	0.09
E	-1	10.05	-	12	0.08
F	1	11.05	5.12	18	0.05

The Forgetting Strategy. For all the queues in LCE_a, the forgetting function associates a *relevance weight* for each experience stored in the queue. As an interesting case illustrated in Table 1, the forgetting function $forget(index) = \frac{1}{index}$ is used, yielding much more relevance for any experience in the queue than another one of greater index.

For instance, the experience C stored at the index 10 in the queue $LCE_{getc}(\ell_2)$ has a forget value of $forget(10) = 0.10$, whereas the experience F stored at the index 18 has a forget value of $forget(18) = 0.05$.

Observe that every queue $LCE_a(\ell)$ is bounded by K elements. Beyond the forgetting of the extra (older) data, this allows one to tackle the data explosion problem implied by the consideration of many experiences over LCE. Indeed, in case the queue is full, the adding of a new experience causes the removing of the oldest one.

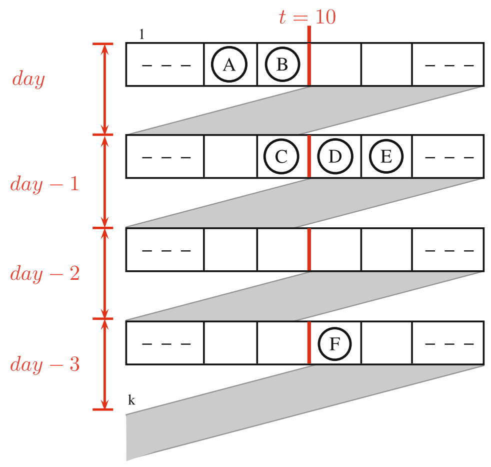

Fig. 5. The example of the queue $LCE_{getc}(\ell_2)$

The Time Filtering Strategy. Regarding to any queue $LCE_a(\ell)$, in order to operate the selection, both the start dates of experiences and the *current date value* '*date*' are evaluated through some classification period $\mathcal{C}$. For this purpose, we introduce the function $mod : \mathcal{C} \times \mathbb{R}^+ \longrightarrow \mathbb{R}^+$, s.t. $mod_{\mathcal{C}}(date)$ corresponds to the start date modulo the classification period $\mathcal{C}$ (we consider standardly that date can be viewed as its textual form or like real timestamp value). For instance, if $date =$ `'Monday 10 February 2015, 10:00'`, then $t = mod_{daily}(date) = 10$ whereas $mod_{weekly}(date) =$`'Monday 10'`. We filter the queue experiences in order to only consider the ones which have the smallest time interval $(|t - mod_{\mathcal{C}}(exp.start)|)$. The mapping $filter_{M,\mathcal{C}}(t)$ of $LCE_a(\ell)$ specifies that M experiences must be selected. In case M is greater than the size k of the queue, all the past-experiences of the queue are considered.

In Table 1, the applied filtering is $filter_{6,daily}(10) = \{A, B, C, D, E, F\}$ which means that $LCE_{getc}(\ell_2)$ is filtered on the 6 closest experiences, w.r.t. $t = mod_{daily}(date) = 10$. As illustrated in Figure 5 for $mod_{daily}(t)$ over the start dates of $LCE_{getc}(\ell_2)$, the modulo operation applied to the queue graphically yields a spiral ribbon, the rings of which correspond to the successive periods, e.g. days in our example. Daily speaking, the start dates of the experiences A, B and C occur before t, and the ones of D, E and F occur after t.

In this example, we have taken the following instantiated strategy: $\mathcal{S}_1 = (20, \frac{1}{index}, 6, daily)$.

4.3 Computing the Expected Performance and Expected Duration for an Action

For each non empty queue $LCE_a(\ell)$, the expected performance $EP_a(\ell) \in [-1, 1]$ represents the performance of a in some location ℓ, based on M experiences filtered from $LCE_a(\ell)$ s.t.:

$$EP_a(\ell) = \frac{\sum_{exp}^{filter_{M,C}(t)} exp.outcome * forget(index(exp))}{\sum_{exp}^{filter_{M,C}(t)} forget(index(exp))} \tag{1}$$

When performing a in ℓ, the closest $EP_a(\ell)$ is to '1', the greater the chance for success, whereas the closest $EP_a(\ell)$ from '-1', the greater the risk of failure. In the case $LCE_a(\ell) = \emptyset$ meaning that the running of a in ℓ has not been already explored, we choose $EP_a(\ell) = 0$ in order to privilege the exploration against every (bad) case s.t. $EP_a(\ell) < 0$.

Again for each non empty queue $LCE_a(\ell)$, the expected duration value $ED_a(\ell) \in \mathbb{R}^+$ corresponds to the effective durations of the M filtered experiences, according to:

$$ED_a(\ell) = \frac{\sum_{exp}^{filter_{M,C}(t)} exp.duration * forget(index(exp))}{\sum_{exp}^{filter_{M,C}(t)} forget(index(exp))} \tag{2}$$

Coming back to the frame example, from $t = 10$, we obtain $EP_{getc}(\ell_2) = 0.19$. Moreover, $ED_{getc}(\ell_2) = 3.16$, which in real time, this stands for `3h10mn`. Observe that the past-experience F has a weak impact on $ED_{getc}(\ell_2)$ despite its important duration (5.12). In fact, the forgetting function applied to its (important) 18 index, makes it negligible compared to the other filtered past-experiences of lower indexes.

5 Spatio-Temporal Guidance from Past-Experiences

5.1 Contextual Planning System with Learning (CPS-L)

The structure *CPS-L* inherits from the maximum traces Σ_{MAX} of the CPS, augmented by the different values $EP_a(\ell)$ and $ED_a(\ell)$ whatever the action a to perform in ℓ.

Definition 5. *Let Σ_{MAX} be the set of the maximum traces of a CPS built from the set of intentions of the agent. The* Contextual Planning System with Learning CPS-L *is a tuple $\langle CPS, \mathcal{EP}, \mathcal{ED} \rangle$ where:*

– *$CPS = \langle S, s_0, Tr, \mathcal{L}, \mathcal{T} \rangle$ s.t. $\Sigma_{MAX} \subseteq 2^{Tr}$ is the set of the maximum traces of the CPS.*

– *$\mathcal{EP}$ is a mapping from Tr to $[-1, 1]$, s.t. from each transition $tr = (s, a, s) \in \Sigma_{MAX}$, $\mathcal{EP}(tr) = EP_a(\mathcal{L}(s))$,*
– *$\mathcal{ED}$ is a mapping from Tr to $\mathbb{R}^+$, s.t. from each transition $tr = (s, a, s) \in \Sigma_{MAX}$, $\mathcal{ED}(tr) = ED_a(\mathcal{L}(s))$.*

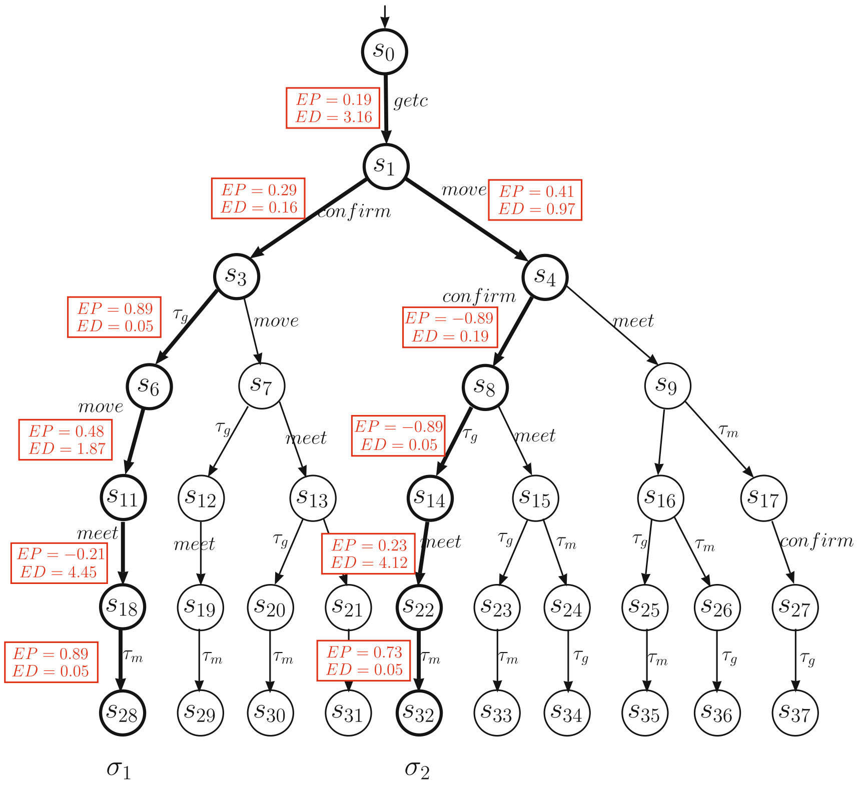

Fig. 6. Maximum traces and $CPS\text{-}L_B$ for the plan $\overline{P_B}$

Figure 6 is a representation of the CPS-L structure derived from the CPS of Figure 3. It distinguishes the maximum traces (with possibly common prefix), and 2 of them explicitly exhibit the labels attached to the transitions. For instance, the values $EP_{getc} = 0.19$ and $ED_{getc} = 3.16$ are attached to the transition $(s_0, getc, s_1)$. These values are computed from the experiments selected in $LCE_a(\ell)$, specified in Table 1.

From any CPS-L, it is straightforward to extend the *expected quality* to every maximum trace $\sigma \in \Sigma_{MAX}$, in order to compare them, as follows:

$$QP(\sigma) = \frac{\sum\limits_{tr \in Tr}^{\sigma} \mathcal{EP}(tr)}{|\sigma|} \tag{3}$$

$$QD(\sigma) = \sum_{tr \in Tr}^{\sigma} \mathcal{ED}(tr) \tag{4}$$

In order to compare the traces in-between them, we normalize all the expected quality values to be in $[-1, 1]$, s.t. '-1' represents the worst case and '1' the best one. This is already done for QP, moreover, to normalize QD, we consider the extreme values $QD_{min} = min(QD(\sigma) \mid \sigma \in \Sigma_{MAX})$ and $QD_{max} = max(QD(\sigma) \mid \sigma \in \Sigma_{MAX})$, hence, the *normalized quality of duration*, is featured by the $NQD(\sigma)$ expression, for any represented σ.

$$NQD(\sigma) = 1 - \left(\frac{QD(\sigma) - QD_{min}}{QD_{max} - QD_{min}} * 2\right) \tag{5}$$

5.2 Optimal Trace of the CPS-L

The algorithm 1 synthetically highlights the planning process approach for selecting an (optimal) maximum trace, σ_{opt}. As this process acts as a service provider, it must also notify the case where there is no possible maximum trace. Actually, this is useful to let the mental process revise its planning process request in accordance. To select an optimal trace, the set Σ_{MAX} of maximum traces is ordered regarding the assigned values from QP and NQD, considered separately.

Algorithm 1. Spatio-Temporal Guidance process

```
1: Require:
   $I$: set of weighted intentions;
   $\mathcal{S}$ : relevance strategy;
   $\mathcal{B}$ : balance proportions;
   $LCE$: Learned Contextual Experiences;

2: Build $\overline{P}$ from $I$;
3: Construct the $CPS$ from $\overline{P}$;
4: Extract $\Sigma_{MAX}$ from the $CPS$;
5: if $\Sigma_{MAX} \neq \emptyset$ then
6:    Enrich the $CPS\text{-}L$ from the $CPS$ and $LCE$;
7:    Order $\Sigma_{MAX}$ from the $CPS\text{-}L$;
8:    Offer $\sigma_{opt}$ among the ones of $\Sigma_{MAX}$;
9: end if
```

This allows the planning process to select an optimal sequence, either with the lowest duration or the maximum performance (i.e. with the minimum risk of failure). To make a compromise, we also assume that the mental process can specify a preference *balance* denoted $\mathcal{B}$, to the planning process. It is expressed as a proportion to be applied to the QP and NQD values, in order to obtain a comparable *global quality* value $Q_{\mathcal{B}}(\sigma)$ belonging to $[-1, 1]$ for all the maximum traces ($\sigma \in \Sigma_{MAX}$). Of course, the optimal trace σ_{opt} corresponds to the maximum trace having the best global quality value.

$$Q_{\mathcal{B}}(\sigma) = \mathcal{B}_P * QP(\sigma) + \mathcal{B}_D * NQD(\sigma) \tag{6}$$

Table 2. Computed quality values of σ_1 and σ_2 maximum traces

Σ_{MAX}	QP	QD	NQD	$Q_{\mathcal{B}1}$	$Q_{\mathcal{B}2}$
σ_1	0.42	9.74	−0.63	−0.31	0.10
σ_2	−0.04	8.54	0.27	0.17	0.05

Table 2 highlights the global qualities of two maximum traces σ_1 and σ_2 of Figure 6, knowing that over all the maximum traces, we obtain $QD_{min} = 7.58$ and $QD_{max} = 10.23$. Assuming the preference balance $\mathcal{B}1$ s.t. $\mathcal{B}1_P = 0.30$ and $\mathcal{B}1_D = 0.70$, i t appears that $Q_{\mathcal{B}1}(\sigma_2)$ is greater than $Q_{\mathcal{B}1}(\sigma_1)$, so the maximum trace σ_2 can be offered to the execution process as the optimal maximum trace. In contrast for $\mathcal{B}2$ ($\mathcal{B}2_P = 0.70$, $\mathcal{B}2_D = 0.30$), σ_2 is the one that can be offered ($Q_{\mathcal{B}2}(\sigma_1) > Q_{\mathcal{B}2}(\sigma_0)$).

A last remark is that the exploitation of the CPS-L allows efficient exploitation/exploration guidance by focusing on some transition labels. In particular, the selection of a trace having a '0' labeled transition corresponds to exploration of untested action, whereas a trace having a '-1' labeled transition should fail, but that could be selected in a degraded mode, when there is no better choice. Exploration can be investigated since our approach to select good traces from the past-experiences is only heuristics. In contrast, a conservative agent (rejecting exploration) should abstract the above exploration cases.

6 Discussion and Related Works

Learning aims to improve the behavior of intelligent agents thanks to the experimental-based information. It has been involved at various levels of the agents. Learning was first investigated at the mental level of BDI agents either to improve the BDI deliberation from some learned knowledge e.g. [7] or to produce a new plan with respect to some objective e.g. [8].

Learning was also used to reinforce the selection of plans among different possible alternatives. In particular, a decision tree was introduced by [9] to represent the different contexts in which the agent behaves. Indeed, the behavior of the agent is learned from the successes and failures of the executions of the agent's plans. The idea to take advantage from the past-experiences was adapted in [4] bringing out an on-line technique, based on a hierarchical goal-plan structure, in order to make the selection of some alternative according to the failures of the previous ones.

The work proposed in [5] addresses plan selections. Closer to our approach, it proposes to evaluate the contribution of each plan in terms of utility and preferences seen as parameters for optimization. In our approach the utilities are given by the agent and are similar to the concept of soft-goal and the preferences discussed in [5]. They correspond, in our approach, to a multi-objective strategy based on the proportions of performances and temporal filtering information, together used for the selection of some maximum traces. Let us notice that our traces depend of real experiments instead of pre-established probabilities.

For the two last approaches [5,10], the validation of the proposed techniques requires a huge number of experiences since they are based on probabilities. In this paper, unlike the former proposals, our approach is driven by the maximization of intentions' satisfaction, in order to guide the agent through the set of traces implied by the concurrency of actions.

The learned CPS can be viewed as reinforcement learning for the selection of elementary plans but based on the successes and failures of the executed actions. Actually, the selection of a trace implies the selection of some alternative for each intention plan. The queues recording the past-experiences of actions are bounded in order to tackle the combinatorial complexity introduced by their management. This approach is compatible with the idea of forgetting useless history.

Regarding the relevance strategies, our technique is aligned with earlier propositions like the Q-learning algorithm. The last one learns a quality value for each action taking into account that the known values become deprecated as the time progresses, under a function like $\epsilon = 1/t$. In our context, the applied forgetting function $(1/x)$ allows us to privilege the most recent past-experiences according to a logical recording time.

Finally, our plans are built dynamically by reasoning on situated actions and their context (duration, start, location, etc.) which is valuable for dynamic environment and contexts.

7 Conclusion

The proposed CPS-L planning tree structure reveals various interests in the agent design since it demonstrates a formal representation of the concurrency of the plans to achieve the intentions of the agent. In fact, each maximum trace in this structure represents a possible plan to execute at real time, in respect to any current context of the agent. As a major contribution of this paper, guidance is reinforced by exploiting the past-experiences of the execution of situated actions. Face to the change of contexts, the forgetting strategy is of a great help since it allows to privilege the recent past experiences. We provide two utility functions that is used to label the CPS-L, respectively based on the outcomes (the success or failure information of the executions) and time concepts (the starts and durations of the executions). Exploration is also considered in case there is no past-experiment for some situated action. Anyway, these CPS-L labels allow us by extension to compare the possible plans. A normalization approach is demonstrated in order to combine both functions, so that optimizations is balanced in-between speed and safety plans.

The CPS-L of the frame scenario, is built out from our prototype tool associated with an automatic generation of the past-experiences. It yields from the study that the current time of the agent can effectively be used as a reference to filter a number of experiences, even in the case the past-experiences were performed under a time periodical strategy. With respect to every situated action, the used filtering function strongly improves the chances to only take relevant past-experiences into account, in respect to the current time of the agent, however, this reduces the number of experiences used to estimate the duration of the action. In fact, different parameters could be used to adapt the influences of the past-experiences, therefore, we are now investigating the different strategies to better guide the agent.

References

1. Chaouche, A.-C., Seghrouchni, A.E.F., Ilié, J.-M., Saïdouni, D.E.: A Higher-Order Agent Model with Contextual Planning Management for Ambient Systems. In: Kowalczyk, R., Nguyen, N.T. (eds.) TCCI XVI. LNCS, vol. 8780, pp. 146–169. Springer, Heidelberg (2014)
2. Calvaresi, D., Claudi, A., Dragoni, A.F., Yu, E., Accattoli, D., Sernani, P.: A goal-oriented requirements engineering approach for the ambient assisted living domain. In: PETRA 2014, pp. 20:1–20:4. ACM, New York (2014)
3. Preuveneers, D., Novais, P.: A survey of software engineering best practices for the development of smart applications in ambient intelligence. JAISE 4(3), 149–162 (2012)
4. Airiau, S., Padgham, L., Sardina, S., Sen, S.: Incorporating learning in BDI agents. In: Proceedings of ALAMAS - ALAg (2008)
5. Nunes, I., Luck, M.: Softgoal-based plan selection in model-driven bdi agents. In: AAMAS 2014, pp. 749–756 (2014)
6. Meneguzzi, F., Zorzo, A.F., da Costa Móra, M., Luck, M.: Incorporating planning into BDI agents. Scalable Computing: Practice and Experience 8, 15–28 (2007)
7. Merke, A., Riedmiller, M.: Karlsruhe brainstormers - a reinforcement learning approach to robotic soccer. In: Birk, A., Coradeschi, S., Tadokoro, S. (eds.) RoboCup 2001. LNCS (LNAI), vol. 2377, pp. 435–440. Springer, Heidelberg (2002)
8. Karim, S., Subagdja, B., Sonenberg, L.: Plans as products of learning. In: IAT 2006, pp. 139–145 (2006)
9. Guerra-Hernández, A., El Fallah-Seghrouchni, A., Soldano, H.: Learning in BDI multi-agent systems. In: Dix, J., Leite, J. (eds.) CLIMA 2004. LNCS (LNAI), vol. 3259, pp. 218–233. Springer, Heidelberg (2004)
10. Waters, M., Padgham, L., Sardina, S.: Evaluating coverage based intention selection. In: AAMAS 2014, pp. 957–964 (2014)