# Spatio-Temporal Guidance for Ambient Agents

Ahmed Chawki Chaouche, Amal El Fallah-Seghrouchni, Jean-Michel Ilié, Djamel Eddine Saïdouni

# Spatio-Temporal Guidance for Ambient Agents

Ahmed-Chawki Chaouche*§, Amal El Fallah Seghrouchni*,
Jean-Michel Ilié* and Djamel Eddine Saïdouni§
* *LIP6, UMR 7606 UPMC - CNRS, 4 place jussieu 75005 Paris, France*
§ *MISC Lab., University Abdelhamid Mehri, Campus Ali Mendjeli, 25000 Constantine, Algeria*
*Emails: ahmed.chaouche@lip6.fr, amal.elfallah@lip6.fr*
*jean-michel.ilie@lip6.fr and saidouni@misc-umc.org*

***Abstract*— This paper proposes a planning management framework dedicated to dynamic systems. It deals with intentional BDI agents in aim to maximize the satisfaction of their intentions taking into account the spatio-temporal context.**

**The search for the optimal solution is based on an original and formal structure called Contextual Planning System (CPS). The CPS handles the satisfaction of concurrent plans associated with intentions. It looks for efficient guidance to improve the satisfaction of the agent's intentions with respect to its current context.**

**Our paper goes on to show how to take advantage of context-aware information representing spatio-temporal data related to the past-experience of action plan executions. Our objective is to improve the CPS mechanism by the use of reinforcement learning strategy. A prototype tool associated with automatic generation of experiences validates our approach on a significant number of experiences.**

***Keywords*-Ambient agent; BDI; contextual guidance; past-experiences.**

## I. Introduction

For the design of complex systems, Multi-Agent System (MAS) approaches offer interesting frameworks, since their agents are designed to be intelligent, proactive and autonomous [5]. In particular, BDI models offer a good alternative for reactive systems whenever planning is required.

Indeed, the work of [4] has shown how autonomous BDI agents [13] can evolve and move within a dynamic environment, based on an agent centric approach that combines planning and context-awareness. In that approach, the environment is open and dynamic. Agents may enter and leave, and as they move, their location may change at runtime [11], [12].

This paper introduces a planning management framework. Our approach relies on the following assumptions:

- The BDI-agent evolves in a dynamic environment. It is sensitive to the context and her changes; i.e. it is aware of the changes of the context.
- The BDI-agent is rational: it is goal-driven, i.e. the agent will pursue its goals; and it behaves according its strategy, i.e. the agent will try to satisfy the maximum of her concurrent intentions according to its strategy.
- The BDI-agent weighs the intentions with respect to its strategy.

Under the previous assumptions,it is necessary to increase the agent reactiveness to the continuous changes of its context. In order to endow the BDI-agent with a such reactive behavior we developed an approach in 4 phases detailed in the following sections.

Section II develops the planning process including a brief description of *AgLOTOS specification language*. It consists of building the *Plan of an agent* ($\overline{P}$) from the current intentions it aims to satisfy (in a concurrent way). A plan called *Intention Plan* ($\widehat{P}_i$) is associated with each intention. The intention plan is composed of *Elementary Plans* ($P$) corresponding to different ways (alternative plans) to achieve an intention.

Section III discusses the construction of the *Contextual Planning Structure* (*CPS*), given a contextual state of the agent. The agent may interleave the execution of its actions when they belong to intention plans ($\widehat{P}_i$) with the same weight. The CPS is then the projection of all possible executions corresponding to the actions of $\overline{P}$. The CPS takes into account the possible interleavings (i.e. interleavings that respect the weights of intentions and the current context). Intuitively speaking, each path of the CPS is a trace of a possible execution of $\overline{P}$ .

Section IV improves the CPS with learning: unlike most existing approaches [14], [2], we will not learn plans. Indeed, we are interested with a finer granularity, i.e. we will learn actions in order to qualify them (with success or failure) and to combine these actions in order to reactively build robust plans in a dynamic context. To improve the CPS we will reason on situated actions as a pair $(a, \ell)$ where '$\ell$' is the location where the action '$a$' occurs. Various contextual indicators could be associated with $(a, \ell)$ such as: the outcome of the action (failure or success), the starting time (when the action begins) and the duration (the time spent for the action achievement). In addition, the agent, according to its rationality, may have various strategies for the optimization of its intentions' achievement. For instance, the agent may consider the new learning information instead the older ones, or may concentrate on the data around a

specific time value. The agent strategy $S$ is given as an input to the optimization algorithm.

In Section V, the optimization algorithm is conceived to guide the agent. Two coefficients have been first considered to evaluate a situated action $(a, \ell)$. The first coefficient is the *expected performance* '$EP$' which computes an evaluation of the achievement of $(a, \ell)$. It is an average of the obtained outcomes through some selected past-experiences. The second coefficient is the *expected duration* '$ED$' of the situated action $(a, \ell)$ computed from several past-experiences. The CPS is enriched with these two coefficients $EP$ and $ED$ associated with its edges. The advantage of the edge qualification is that the agent can choose a robust trace, or can avoid risky traces, etc. A trace qualification is deduced from the propagation of $ED$ and $EP$ through the CPS, based on a normalized accumulation. The guidance algorithm provides the agent with a set of maximum traces that respect the agent strategy. Moreover, the preferences of the agent between "race" and "secure" performances are considered. For instance, taking a speedy car could be interesting but can be risked w.r.t. the wish of the agent to safely arrive to some destination. To be progressive, these preferences are expressed by the agent as a *balance* proportion $\mathcal{B}$, e.g. $30\%$ for race and $70\%$ for secure.

Section VI discusses our approach with regard to the state of the art and Section VII concludes this paper.

## II. Agent Plan Specification

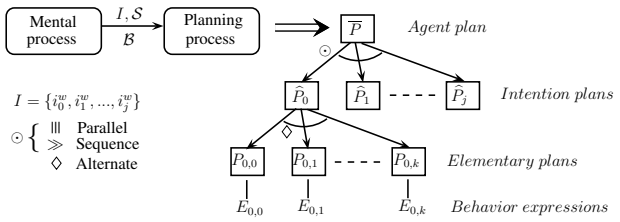


Figure 1. Agent planning structure

As usual in BDI models, the *mental process* represents the reasoning mechanism, based on the beliefs (B), desires (D), and intentions (I), the instances of which define the BDI states of the agent. Triggered by the perceived events, the mental process manages/updates the B, D and I structures.

As illustrated in Figure 1, the mental process can call the *planning process* to produce a plan, namely the *agent plan* ($\overline{P}$). Unlike other approaches, in our framework the plans corresponding to each intentions, namely *intention plans* ($\widehat{P}_i$) are achieved concurrently, according to a *weighting* in order to privilege some intentions over others. This allows the mental process to solve conflicting situations, by ordering the intentions plans. However, when intentions are not conflicting, the same weight enables concurrent achievement of the intention plans in an interleaved way. Moreover, each intention plan has alternative plans to be tried, called *Elementary plans* ($P$), obtained from a *library of plans* (*LibP*). This allows us to consider different ways to achieve the associated intention. Further, we assume that the *LibP* library is indexed by the set of all the possible intentions for the agent, yielding the elementary plans that achieve some intention.

Let us now detail the highlights of the AgLOTOS-based specifications for plans, previously presented in [4]. AgLOTOS inherits from the LOTOS language [3] so offers different ways to express the concurrency of actions in plans. It takes into account the AmI aspects i.e. mobility and communication.

*Agent plan level:* The set of intention plans can be globally handled by using the concurrent $|||$ or sequential $\gg$ AgLOTOS composition 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, such that each one can be launched to solve the corresponding intention. This is captured in AgLOTOS by using the composition operator $\Diamond$ to express the possibility to have alternative plans for executing intention. In particular, an intention is satisfied if and only if at least one of the associated elementary plans is successfully achieved. Formally, 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 relating to the (current) BDI state of an agent. For that, let $\Theta$ be a finite set of space locations where an agent can move and $\Lambda$ 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 $a, b, ...$ and let $L$ be any subset of $\mathcal{O}$. $H \subset \mathcal{O}$ is the set of the so-called mobility and communication primitives. The agent mobility is expressed by the primitive $move(\ell)$ which is used to handle the agent move to some location $\ell$ ($\ell \in \Theta$). The syntax of the communication primitives is defined by the expression $x!(\nu)$ which specifies the transmission/sending to the agent $x$ ($x \in \Lambda$) of some message $\nu$ and the expression $x?(\nu)$ means that the message $\nu$ is received from some agent $x$.

Let $Act = \mathcal{O} \cup \{\tau\}$, be the set of actions, where $\tau$ is the internal action. The AgLOTOS language specifies pairs for each elementary plan composed of an identifier and an AgLOTOS expression to feature its plan behavior.

Consider that the set of elementary plan's names is ranged over by $P_1, P_2, ...$ and that the set of all the possible behavior expressions is denoted $\mathcal{E}$, ranged over by $E_1, E_2, ....$ The expressions are written by composing actions through LOTOS operators. The syntax of an elementary plan $P$ is defined inductively as follows:

$$
\begin{array}{lcll}
P & ::= & E & \\
E & ::= & exit \mid stop & \\
 & & \mid a;E \mid E \odot E & (a \in \mathcal{O}) \\
 & & \mid hide\ L\ in\ E & \\
H & ::= & move(\ell) & (H \subset \mathcal{O}, \ell \in \Theta) \\
 & & \mid x!(\nu) \mid x?(\nu) & (x \in \Lambda) \\
\odot & = & \{\ [\ ], \gg, [>, |[L]|, ||, ||| \ \} &
\end{array}
$$

The elementary expression $stop$ specifies a plan behavior without possible evolution and $exit$ represents the successful termination of some plan. In the syntax, the expression $hide\ L\ in\ E$ denotes a hiding of the actions of $L$ that appear in $E$. The set $\odot$ represents the standard LOTOS operators: $E\,[\,]\,E$ specifies a non-deterministic choice, $E \gg E$ a sequential composition and $E\ [> E$ the interruption. The LOTOS parallel composition, denoted $E\,|[L]|\,E$, can model both synchronous composition, $E\,||\,E$ if $L = \mathcal{O}$, and asynchronous composition, $E\,|||\,E$ if $L = \emptyset$.

*Building of Agent Plan from Intentions:* In order to account for any BDI state of the agent, we propose that the agent can order 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 let $\widehat{P_g}, \widehat{P_e}, \widehat{P_m}$ be their corresponding intention plans, the constructed agent plan could be viewed as: $\overline{P} = (\widehat{P_g}|||\widehat{P_m}) \gg \widehat{P_e}$.

## III. Contextual Planning System

With respect to some agent plan $\overline{P}$, we introduce a notion of configuration, denoted $[\overline{P}]$, referring to the planning state of the agent in order to specify its plan evolution. We will now define the contextual planning state of the agent, taking into account the agent location and the outcomes of different intention plans defined for the agent.

*Definition 3.1:* A contextual planning state is a tuple $(ps, \ell, T)$, where $ps$ is any planning state described by an agent plan configuration $[\overline{P}]$, $\ell$ corresponds to the possible location for the agent in this state, and $T$ is the subset of intention plans who terminate successfully in this state.

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 $\ell$ and $ps$ is its planning state.

*Definition 3.2:* 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 $a \in Act$,
- $\mathcal{L} : S \rightarrow \Theta$ is the location labeling function,
- $\mathcal{T} : S \rightarrow 2^{\widehat{\mathcal{P}}}$ is the intention termination labeling function which captures the intention plans that have been completed.

### A. 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: $\overline{P_B} = (\widehat{P_g}|||\widehat{P_m})$, where $\widehat{P_m}$ and $\widehat{P_g}$ are two concurrent intention plans to be achieved. The first one corresponds to the intention $meeting(Alice, \ell_1)$ and the second to $getting\_copies(\ell_2)$. For sake of simplicity, both intention plans only contain one elementary plan each, without other alternatives. Here, $\widehat{P_m} = move(\ell_1); meet(Alice); exit_m$ and $\widehat{P_g} = getc(\ell_2); confirm; exit_g$. The resulting agent plan expression for Bob is thus to act two concurrent processes: *(1)* get the copies locally in $\ell_2$ and confirm this to Alice ($Alice!(confirm)$) and *(2)* move to $\ell_1$ to meet Alice there.

An example of derivation from the initial planning state is to perform the $getc$ action. So, the planning state change from $[\overline{P_B}] \xrightarrow{getc} [\overline{P'_B}]$, yields $[\overline{P'_B}] = move(\ell_1); meet(Alice); exit_m \ ||| \ confirm; exit_g$.

The Contextual Planning System of Bob, denoted $CPS_B$, is illustrated in Figure 2. It is built from the initial CPS state, $s_0 = ([\overline{P_B}], \ell_2, \emptyset)$, taking into account the current location $\ell_2$ of Bob.

### B. Guidance for Intention Satisfaction

In a CPS, the transitions from any (source) state $s \xrightarrow{a} s'$ represent actions to be performed. Like in the STRIPS description language [8], actions are modeled by instantiated predicates defined with preconditions and effects. In this paper, the preconditions concern only the contextual

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

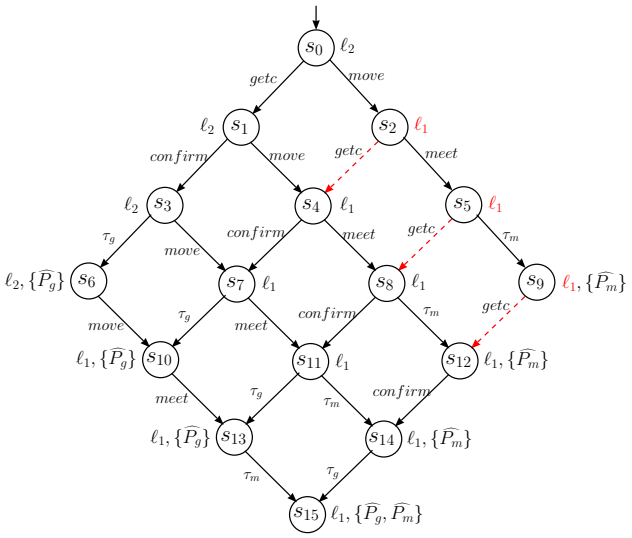


Figure 2. The $CPS_B$ corresponding to the plan $\overline{P_B}$

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 2, 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 $\Sigma_{MAX}$ represent the set of maximum traces of the CPS.

As a specific tree structure, Figure 5 represents the 10 maximum traces of $CPS_B$. In this unfolded version of Figure 2, the trace carried out by $s_0 \rightarrow s_2 \rightarrow s_5 \rightarrow s_9$, is not represented because it is not a maximum trace.

## IV. 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 running of actions 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.

### A. 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*.

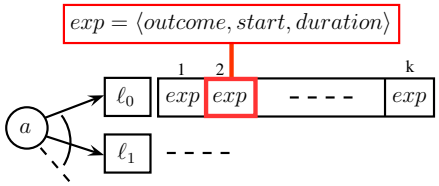


Figure 3. Learned Contextual Experiences (LCE) of an action $a$

*Definition 4.1:* (Action experience) An experience of an action $a$ in a location $\ell$ 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 3 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 $\ell$ 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 $exp.outcome$, $exp.start$, $exp.duration$.

### B. Data Relevance Strategies

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

*Definition 4.2:* The queues of $LCE$ are parameterized by the agent strategy $\mathcal{S} = \langle K, forget, M, \mathcal{C}, filter \rangle$ 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 $\mathcal{C}$,

**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 I, the forgetting function $forget(index) =$

Table I
LCE OF THE ACTION *getc* IN THE LOCATION $\ell_2$

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

$filter_{6,daily}(10)$

| Exp. | $outcome$ | $mod_{daily(start)}$ | $duration$ | $index$ | $forget$ |
|---|---|---|---|---|---|
| A | -1 | 8.29 | - | 5 | 0.20 |
| B | 1 | 9.50 | 3 | 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 |

$\frac{1}{index}$ 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.

**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 $mod : \mathcal{C} \times \mathbb{R}^+ \rightarrow \mathbb{R}^+$, 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 ($|t - mod_C(exp.start)|$). 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 I, 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 4 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: $\mathcal{S}_1 = (20, \frac{1}{index}, 6, daily, filter)$.

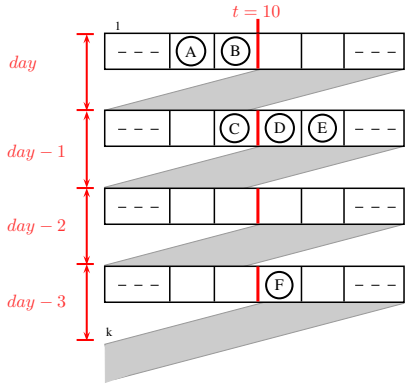


Figure 4. The example of the queue $LCE_a(\ell)$

### C. 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 $\ell$, based on $M$ experiences filtered from $LCE_a(\ell)$ s.t.:

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

When performing $a$ in $\ell$, 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 $\ell$ 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:

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

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.

## V. Spatio-Temporal Guidance from Past-Experiences

### A. Contextual Planning System with Learning (CPS-L)

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

*Definition 5.1:* Let $\Sigma_{MAX}$ 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 $tr = (s, a, s) \in \Sigma_{MAX}$, $\mathcal{ED}(tr) = ED_a(\mathcal{L}(s))$.

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

$$QP(\sigma) = \frac{\sum_{tr \in Tr}^{\sigma} \mathcal{EP}(tr)}{|\sigma|}$$

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

To compare the traces 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 $QD_{max} = max(QD(\sigma) \mid \sigma \in \Sigma_{MAX})$. We obtain the *normalized quality of duration*, as follows:

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

In Figure 5, the edges of two maximum traces are augmented by the two values $EP$ and $ED$, as in the CPS-L structure. For instance, the values $EP_{getc} = 0.19$ and $ED_{getc} = 3.16$ are attached to the transition $(s_0, getc, s_1)$ computed from the values of $LCE_a(\ell)$, specified in Table I, are attached to the transition $(s_0, getc, s_1)$.

As shown in Figure 6, the traces the quality of which enters the right-upper part of the figure are the best ones, and the ones entering the left-lower part are the worse ones. The other traces having a single high value either for $QP$ or $NQD$ enter the two other parts of the figure.

### B. Optimal Traces of CPS-L

We now consider that in addition to intentions $I$ and strategy $\mathcal{S}$, the mental process can specify a preferred *balance* denoted $\mathcal{B}$ between the two qualities for traces, performance and normalized duration. It helps characterizing

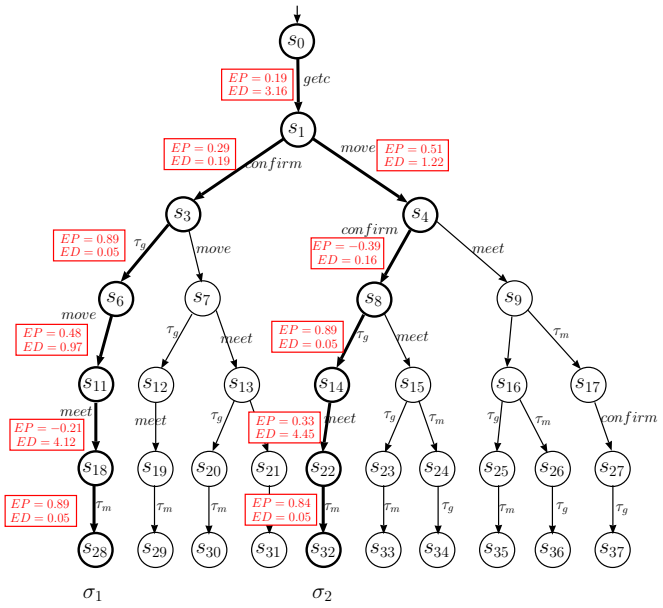


Figure 5. Maximum traces and $CPS\text{-}L_B$ for the plan $\overline{P_B}$

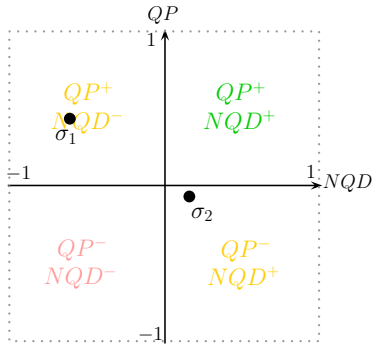


Figure 6. Balance between the $QP(\sigma)$ and the $NQD(\sigma)$ of a trace $\sigma$

the optimal (maximum) trace to be selected, namely $\sigma_{opt}$. The balance 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 maximum traces ($\sigma \in \Sigma_{MAX}$). Of course, the trace $\sigma_{opt}$ of the maximum quality is returned by the planning process.

$$Q_{\mathcal{B}}(\sigma) = \mathcal{B}_{\mathcal{P}} * QP(\sigma) + \mathcal{B}_{\mathcal{D}} * NQD(\sigma)$$

Table II
Computed quality values of $\sigma_1$ and $\sigma_2$ maximum traces

| $\Sigma_{MAX}$ | $QP$ | $QD$ | $NQD$ | $Q_{\mathcal{B}1}$ | $Q_{\mathcal{B}2}$ |
|---|---|---|---|---|---|
| $\sigma_1$ | 0.42 | 9.74 | −0.63 | −0.31 | 0.10 |
| $\sigma_2$ | -0.04 | 8.54 | 0.27 | 0.17 | 0.05 |

Table II highlights the qualities of two maximum traces $\sigma_1$ and $\sigma_2$ of Figure 5, knowing that over all maximum

traces, we obtain $QD_{min} = 7.58$ and $QD_{max} = 10.23$. Moreover, the balance $\mathcal{B}1$ specified by the mental process, is the pair ($\mathcal{B}1_P = 0.30$, $\mathcal{B}1_D = 0.70$). Since $Q_{\mathcal{B}1}(\sigma_2)$ is greater than $Q_{\mathcal{B}1}(\sigma_1)$, the maximum trace $\sigma_2$ can be offered to the execution process as the optimal maximum trace. In contrast for $\mathcal{B}2$ ($\mathcal{B}2_P = 0.70$, $\mathcal{B}2_D = 0.30$), $\sigma_2$ is the one that can be offered ($Q_{\mathcal{B}2}(\sigma_1) > Q_{\mathcal{B}2}(\sigma_0)$). The algorithm 1 is a synthesis of all our approach.

**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: Enrich the *CPS-L* from the $CPS$ and $LCE$;
6: Order $\Sigma_{MAX}$ from the *CPS-L*;
7: Offer $\sigma_{opt}$ among the ones of $\Sigma_{MAX}$;

## VI. Discussion and Related Works

*Learning and planning*

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. [9] or to produce a new plan with respect to some objective e.g. [7].

Learning was also used to reinforce the selection of plans among different possible alternatives. In particular, a decision tree was introduced by [6] 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 execution experiences was adapted in [1] 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 [10] 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 softgoal and the preferences discussed in [10]. 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 [10], [15], 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 paths 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 path 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, locality, etc.) which is valuable for dynamic environment and contexts.

## VII. Conclusion

This paper proposes an original approach to deal with BDI agent planning in a dynamic context. The framework is based on AgLOTOS algebraic language which is suitable to specify an agent plan as a set of concurrent processes, helped by a plan library describing elementary plans. The semantics of AgLOTOS allows to build, as structure, a Contextual Planning System (CPS), for any BDI state of the agent. From the current set of intentions, all the possible evolutions of a plan can be evaluated through the CPS.

The CPS structure has been enriched with contextual and spatio-temporal information related to past-experiences of the agent. Improved with learning, our CPS allows to learn situated actions in order to qualify them with relevant attributes (success or failure, durations, starting time, etc.) and to combine these actions in order to reactively build robust plans in a dynamic context.

Consequently, our approach provides a built-in look-ahead mechanism to contextually guide the agent in his future executions. This guidance, thanks to learning from past experiences, optimizes the number of satisfied intentions

while respecting the agent strategies based on his utilities and preferences.

This work has been applied in the context of ambient intelligence where the concepts of "location" and "context" are the key stone of such systems. As future work, we plan to tackle the problem of the verification of formal properties of this planning system which based on formal specification language and techniques. We plan also to extend our experiment to the context of a major project called "SMART" in which we develop the " SMART CAMPUS " Framework.

## REFERENCES


[1] S. Airiau, L. Padgham, S. Sardina, and S. Sen. Incorporating learning in BDI agents. In *Proceedings of the Adaptive Learning Agents and Multi-Agent Systems Workshop (ALAMAS+ALAg-08)*, Esteroil, Portugal, May 2008.

[2] G. Bourgne, H. Soldano, and A. El Fallah Seghrouchni. Learning better together. In *Proceedings of the 2010 Conference on ECAI 2010: 19th European Conference on Artificial Intelligence*, pages 85–90, Amsterdam, The Netherlands, The Netherlands, 2010. IOS Press.

[3] E. Brinksma, editor. *ISO 8807, LOTOS - A Formal Description Technique Based on the Temporal Ordering of Observational Behaviour*, 1988.

[4] A.-C. Chaouche, A. El Fallah Seghrouchni, J.-M. Ilié, and D. E. Saïdouni. A Higher-order Agent Model with Contextual Management for Ambient Systems. In R. Kowalczyk and N. T. Nguyen, editors, *Transactions on Computational Collective Intelligence XVI*, volume 8780 of *Lecture Notes in Computer Science*, pages 146–169. Springer Berlin Heidelberg, 2014.

[5] J. Doyle. Rationality and its roles in reasoning. *Computational Intelligence*, 8:376–40, 1992.

[6] A. Guerra-Hernandez, A. El Fallah Seghrouchni, and H. Soldano. Learning in bdi multi-agent systems. In J. Dix and J. Leite, editors, *Computational Logic in Multi-Agent Systems*, volume 3259 of *Lecture Notes in Computer Science*, pages 218–233. Springer Berlin Heidelberg, 2005.

[7] S. Karim, B. Subagdja, and L. Sonenberg. Plans as products of learning. In *Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, IAT '06, pages 139–145, Washington, DC, USA, 2006. IEEE Computer Society.

[8] F. Meneguzzi, A. F. Zorzo, M. da Costa Móra, and M. Luck. Incorporating planning into BDI agents. *Scalable Computing: Practice and Experience*, 8:15–28, 2007.

[9] A. Merke and M. Riedmiller. Karlsruhe brainstormers - a reinforcement learning approach to robotic soccer. In A. Birk, S. Coradeschi, and S. Tadokoro, editors, *RoboCup 2001: Robot Soccer World Cup V*, volume 2377 of *Lecture Notes in Computer Science*, pages 435–440. Springer Berlin Heidelberg, 2002.

[10] I. Nunes and M. Luck. Softgoal-based plan selection in model-driven bdi agents. In *Proceedings of the 2014 International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '14, pages 749–756, Richland, SC, 2014. International Foundation for Autonomous Agents and Multiagent Systems.

[11] A. Olaru, A. M. Florea, and A. El Fallah Seghrouchni. A context-aware multi-agent system as a middleware for ambient intelligence. *Mobile Networks and Applications(MONET)*, 18(3):429–443, 2013.

[12] D. Preuveneers and P. Novais. A survey of software engineering best practices for the development of smart applications in ambient intelligence. *Journal of Ambient Intelligence and Smart Environments (JAISE)*, 4(3):149–162, 2012.

[13] A. S. Rao and M. P. Georgeff. An abstract architecture for rational agents. In B. Nebel, C. Rich, and W. R. Swartout, editors, *KR*, pages 439–449. Morgan Kaufmann, 1992.

[14] D. Singh, S. Sardina, L. Padgham, and G. James. Integrating learning into a BDI agent for environments with changing dynamics. In C. K. Toby Walsh and C. Sierra, editors, *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI'11)*, pages 2525–2530, 2011.

[15] M. Waters, L. Padgham, and S. Sardina. Evaluating coverage based intention selection. In *Proceedings of the 2014 International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '14, pages 957–964, Richland, SC, 2014. International Foundation for Autonomous Agents and Multiagent Systems.