Emotional-based Planning

Luís Macedo[1,2]
[1]Department of Informatics and Systems Engineering, Engineering Institute, Coimbra Polytechnic Institute
R. Pedro Nunes, Quinta da Nora, 3030-199 Coimbra - Portugal
lmacedo@isec.pt

Amílcar Cardoso[2]
[2]Centre for Informatics and Systems of the University of Coimbra
Pinhal de Marrocos, 3030 Coimbra - Portugal
{macedo,amilcar}@dei.uc.pt

Abstract

This paper describes an emotional-based planner that combines the technique of decision-theoretic planning with the methology of HTN planning in order to deal with uncertain, dynamic large-scale real-world domains. We explain how plans are represented, generated and executed. Unlike in regular HTN planning, this planner can generate plans in domains where there is no complete domain theory by using cases instead of methods for task decomposition. The planner generates a variant of a HTN - a kind of AND/OR tree of probabilistic conditional tasks - that expresses all the possible ways to decompose an initial task network. The expected utility of alternative plans is computed beforehand at the time of building the HTN. Two approaches are proposed for this computation: based on motivational information collected from past executions of tasks (a kind of somatic-markers) or given by mathematical functions. The planner is used by agents inhabiting unknown, dynamic environments.

1 Introduction

Hierarchical Task Network (HTN) planning is a planning methology that is more expressive than STRIPS-style planning (Erol, Hendler, & Nau, 1994). Given a set of tasks that need to be performed (the planning problem), the planning process decomposes them into simpler subtasks until *primitive tasks* or actions that can be directly executed are reached. *Methods* provided by the domain theory indicate how tasks are decomposed into subtasks. However, for many real-world domains, sometimes it is hard to collect methods to completely model the generation of plans. For this reason an alternative approach that is based on cases of methods has been taken in combination with methods (Muñoz-Avila et al., 2001).

Real-world domains are usually dynamic and uncertain. In these domains actions may have several outcomes, some of which may be more valuable than others. Planning in these domains require special techniques for dealing with uncertainty. Actually, this has been one of the main concerns of the planning research in the last years, and several decision-theoretic planning approaches have been proposed and used successfully, some based on the extension of classical planning and others on Markov-Decision Processes (see (Blythe, 1999; Littman & Majercik, 1997) for a survey). In these decision-theoretic planning frameworks actions are usually probabilistic conditional actions, preferences over the outcomes of the actions is expressed in terms of an utility function, and plans are evaluated in terms of their Expected Utility (EU) (Russel & Norvig, 1995). The main goal is to find the plan or set of plans that maximizes an EU function, i.e, to find the optimal plan. However, this might be a computationally complex task.

Considered by many authors as the principal motivational system, emotion is one of the sub-systems that compose personality (Izard, 1991). Another important sub-system is the drive system (also an important kind of the motivational system). Psychological and neuroscience research over the past decades suggests that emotions play a critical role in decision-making, action and performance, by influencing a variety of cognitive processes (e.g., attention (Izard, 1991; Meyer, Reisenzein, & Schützwohl, 1997; Ortony & Partridge, 1987; Reisenzein, 2000), planning (Gratch, 1999), etc.). Actually, on the one hand, recent research in neuroscience (Damásio, 1994; LeDoux, 1996) supports the importance of emotions on reasoning and decision-making. For instance, results from recent studies of patients with lesions of the prefrontal cortex suggest an important role of emotions in decision-making. On the other hand, there are a few theories in psychology relating motivations (including drives and emotions) to action (Izard, 1991). For instance, in the specific case of emotions, as outlined by (Reisenzein, 1996), within the context of the belief-desire theories of action (the dominant class of theories in today's motivation psychology) there have been proposals such as that emotions are action goals, that emotions are or include action tendencies, that emotions are or include goal-desires, and that emotions are mental states that generate goal-desires.

In this paper we propose an emotional-based approach for decision-theoretic planning, HTN planning. In this approach, actions have several outcomes, each one eliciting different emotions, drives and other moti-

vations (elicited by the objects perceived). This motivational information is collected from past executions of tasks (a kind of somatic-markers) or given by mathematical functions. The selection of actions is based on their EU, which is measured in terms of this motivational information, i.e., based on the intensity of the emotions, drives and other motivations it may elicit. The planner combines the technique of decision-theoretic planning with the methology of HTN planning in order to deal with uncertain, dynamic large-scale real-world domains. Unlike in regular HTN planning, we don't use methods for task decomposition, but instead cases of plans. The planner generates a variant of a HTN - a kind of AND/OR tree of probabilistic conditional tasks - that expresses all the possible ways to decompose an initial task network. The EU of tasks and consequently of the alternative plans is computed beforehand at the time of building the HTN.

The next section describes the features of the planner related with plan representation. Section 3 presents the plan generation process and section 4 the plan execution and replanning process. Finally, we present the related work, and present conclusions and future work.

2 Plan Representation

Within our approach we may distinguish two main kinds of plans: concrete plans, i.e., cases of plans (Kolodner, 1993), and abstract plans. Concrete plans and abstract plans are interrelated since concrete plans are instances of abstract plans and these are built from concrete plans. Since the concept of abstract plan subsumes the concept of concrete plan, let us first describe the representation issues related with abstract plans and then present the main differences between concrete plans and abstract plans.

We represent abstract plans as a hierarchy of tasks (a variant of HTNs (e.g., (Erol et al., 1994; Nau, Muñoz-Avila, Cao, Lotem, & Mitchell, 2001)) (see Figure 1). Formally, an abstract plan is a tuple $AP = <T, L>$, where T is the set of tasks and L is the set of links. More precisely, we represent an abstract plan by a hierarchical graph-structured representation comprising tasks (represented by the nodes) and links (represented by the edges). We adopted the adjacency matrix approach to represent these graphs (Macedo & Cardoso, 1998). The links may be of hierarchical (abstraction or decomposition), temporal, utility-ranking or adaptation kind. This structure has the form of a planning tree (Lotem & Nau, 2000), i.e., it is a kind of AND/OR tree that expresses all the possible ways to decompose an initial task network. Like in regular HTNs, this hierarchical structure of a plan comprises primitive tasks or actions (non-decomposable tasks) and non-primitive tasks (decomposable or compound tasks). Primitive tasks correspond to the leaves of the tree and are directly executed by the agent, while compound tasks denote desired changes that involve several subtasks to accomplish it. For instance, the leaf node *driveTruck* of Figure 1 is a primitive task, while *inCityDel* is a compound task. The decomposition of a compound task into a sequence of subtasks is represented by linking the compound task to each subtask by a hierarchical link of type decomposition (denoted by *dcmp*). This corresponds to an AND structure. In addition, a hierarchical plan may also include special tasks in order to express situations when a decomposable task has at least two alternative decompositions. Thus, these special tasks are tasks whose subtasks are heads of those alternative decompositions. We called abstract tasks to those special decomposable tasks because they may be instantiated by one of their alternative subtasks. Thus, they are a kind of abstractions of their alternative instances. Notice that the subtasks of an abstract task may themselves be abstract tasks. This decomposition of abstract tasks into several alternative instances is expressed by linking the abstract task to each subtask by a hierarchical link of type abstract (denoted by *abst*). This corresponds to an OR structure. As we said, in addition to hierarchical links that express AND or OR decomposition (*dcmp* and *abst*), there are also temporal, utility-ranking and adaptation links between tasks. Temporal links are just like in regular HTNs. We followed the temporal model introduced by (Allen, 1983). Thus, links such as *after*, *before*, *during*, *overlap*, etc., may be found between tasks of an abstract plan. Utility-ranking links (denoted by *more_useful*) are used between subtasks of abstract tasks in order to express a relation of order with respect to their EU, i.e., the head tasks of the alternative decompositions of a given abstract task are ranked according to the EU of their decompositions. Adaptation links (Kolodner, 1993) are useful to generate an abstract plan from several plan cases. They explain how tasks and their components are related in a plan and therefore they explain how to adapt portions of cases of plans when they are reused to construct an abstract plan.

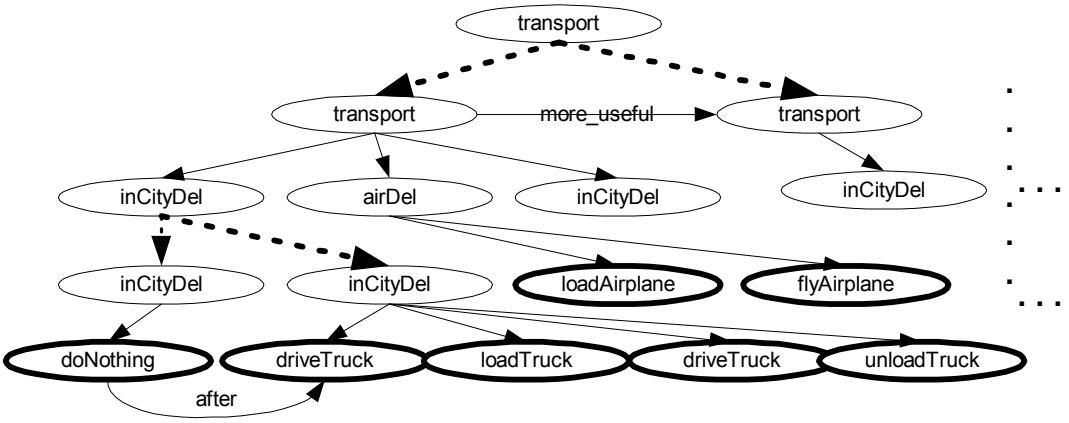

Figure 1 - Example of an abstract plan. Primitive tasks are represented by thick ellipses while non-primitive tasks are represented by thin ellipses. Dashed, thick arrows represent *abst* links, while thin arrows represent *dcmp* links.

A task T is both conditional and probabilistic (e.g.: (Blythe, 1999; Haddawy & Doan, 1994; Younes, 2003)). This means each primitive task has a set of conditions C={ c_1, c_2, ..., c_m} and for each one of these mutually exclusive and exhaustive conditions, c_i, there is a set of alternative effects $\mathcal{E}^i$={$< p_1^i , E_1^i >$, $< p_2^i , E_2^i >$, ..., $< p_{n_i}^i , E_{n_i}^i >$}, where E_j^i is the j^{th} effect triggered with probability $p_j^i \in [0,1]$ by condition c_i (i.e., $P(E_j^i \mid c_i) = p_j^i$), and such that $\sum_{j=1}^{n_i} p_j^i = 1$. Figure 2 presents the structure of a task. The probabilities of conditions are represented in that structure although we assume that conditions are independent of tasks. Thus, $P(c_i|T)=P(c_i)$. The main reason for this is to emphasize that the EU of a task, in addition to the probability of effects, depends on the probability of conditions too. In addition to conditions and effects, a task has other information components. Formally, a task (primitive or not) may be defined as follows.

Definition. A **task** is a tuple <*PS*, *ID*, *TT*, *AID*, *DO*, *IO*, *ST*, *ET*, *SL*, *EL*, *PR*, *A*, *EP*, *EU*, *P*>, where: *PS* is the set of preconditions that should be satisfied so that the task can be executed; *ID* is the task's identifier, i.e., an integer that uniquely identifies the task in a plan; *TT* is the task category (e.g.: *driveTruck*, *transport*); *AID* is the identifier of the agent that is responsible for the execution of the task; *DO* is the direct object of the task, i.e., the identifier of the entity that was subjected to the task directly (e.g.: for a task of type *driveTruck*, the direct object is the object - its id - to be driven; for a task of type *transport*, the direct object is the entity that is transported – for instance, a package); *IO* is the indirect object of the task, i.e., the answer to the question "To whom?" (e.g.: for a task of type *give*, the indirect object is the entity that receives the entity (the direct object) that is given – for instance, the person who receives money); *ST* is the scheduled start time of the task; *ET* is the scheduled end time of the task; *SL* is the start location of the agent that is responsible for executing the task; *EL* is the end location of the agent that is responsible for the execution of the task; *PR* is a boolean value that is true when the task is primitive; *A* is a boolean value that is true when the task is abstract (for primitive tasks it is always false); *EP* is the set of alternative probabilistic conditional effects of the task, i.e., *EP* = {$<c_i,\mathcal{E}^i>$: 1=< i <=m}; *EU* is the Expected Utility of the task; *P* is the probability of the task (this is always 1.0 for every task except the heads of alternative decompositions of an abstract task as we'll explain below).

Although non-primitive tasks are not directly executable by an agent, they are represented like primitive tasks. Therefore, some of the components are meaningful only for primitive tasks. However, others such as the set of alternative probabilistic conditional effects are essential for the ranking of the alternative decompositions of the abstract tasks in terms of the EU. That is why the set of conditional probabilistic effects and other meaningful properties are propagated upward through the hierarchy from the primitive tasks to the non-primitive tasks (this propagation will be explained in detail below).

Each effect (see Figure 2) comprises itself a few components of several kinds such as temporal, emotional, etc. These components may be of two kinds: non-procedural and procedural. The non-procedural (factual) component refers to the data collected from previous occurrences of the effect (contains the duration of the task, the emotions and respective intensities felt by the agent, the fuel consumed, etc., in previous executions of the task as stored in cases of plans). The procedural component refers to the process through which the temporal, emotional and other kinds of data may be computed (contains descriptions or rules of how to compute the components). Since the non-

procedural component of an effect may differ in different occurrences of a task (the duration of the task may be different, the emotions may be different, etc.), effects of tasks belonging to abstract plans may store the probability distributions for each variable (see Figure 2).

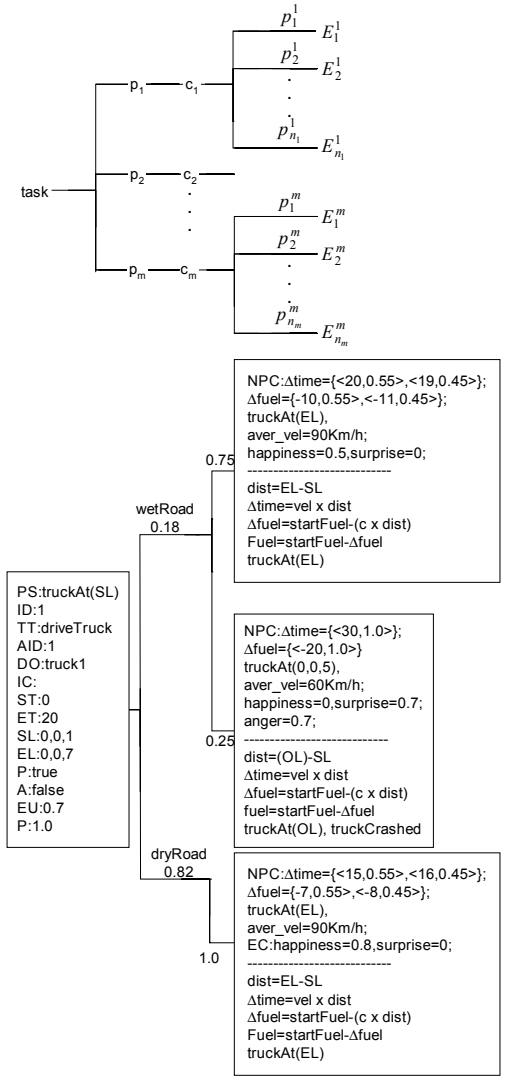

Figure 2 - Schematic representation of a task in abstract plan: general form and example.

Formally, an effect may be defined as follows.

Definition. An **effect** is a tuple <*ID*, *EC*, *EU*, *P*, *NPC*, *PC*>, where: *ID* is the identifier of the effect, i.e., an integer value that uniquely identifies the effect in the list of effects of the task; *EC* is the effect category to which it belongs (like tasks, effects are classified into categories); *EU* is the utility value (expected utility value for the case of tasks in abstract plans) of the effect; *P* is the probability value of the effect, i.e., the relative frequency of the effect (this gives us the number of times the effect occurred given that the task and the condition that triggers it occurred); *NPC* is the non-procedural component; *PC* is the procedural component.

Cases of plans share most of the features of abstract plans being also of hierarchical nature. The major differences are: unlike abstract plans, cases of plans don't have OR structures and consequently don't have abstract tasks; the primitive tasks have a probability of 1.0 (otherwise they won't belong to the case) and can only have a conditional effect since the conditions are mutually exclusive and exhaustive. Notice that, although a non-primitive task of a case of a plan may exhibit an effect, this is not relevant, since in real world only the primitive tasks are executable. However, the way a non-primitive task was decomposed is of primary importance for the generation of abstract plans as we will explain in the following section. Figure 3 shows an example of two cases of plans which are instances of the abstract plan presented in Figure 1, while Figure 4 presents an example of a primitive task which is an instance of the primitive task of an abstract plan presented in Figure 2.

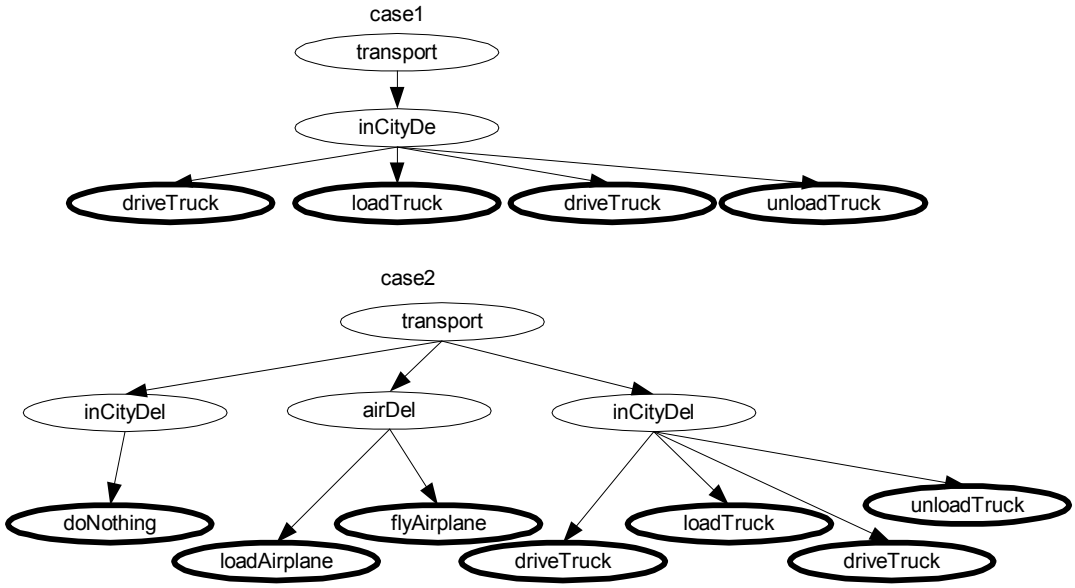

Figure 3 - Example of a case-base with two concrete plans (instances of the abstract plan of Figure 1).

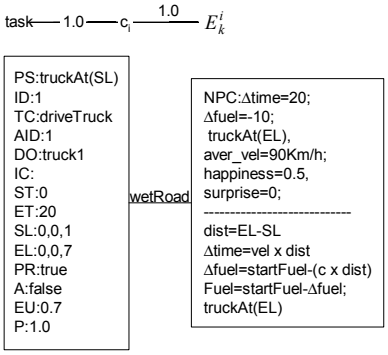

Figure 4 - Schematic representation of a task in an instance plan: general form and example.

3 Plan Generation

Since the planner is used by an agent that is part of a multi-agent environment, in order to solve a planning problem, the agent should have in memory the information of the initial state of the environment. This comprises a three-dimensional metric map of the environment (Thrun, 2002) in which inanimate and other animate agents are spatially represented. Figure 5 presents an example of a metric map that represents an initial state of world.

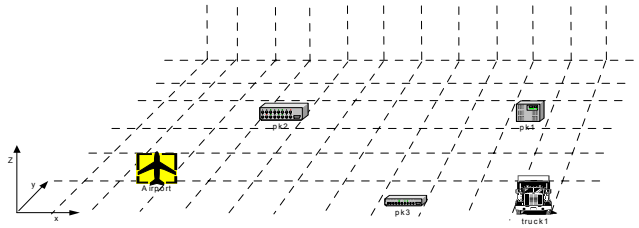

Figure 5 – Example of the metric map of an initial state of the environment in the logistics domain. It comprises: one truck (*truck1*) located at coordinates (11,0,0); three packages, *pk1*, *pk2* and *pk3*, located at, respectively, (8,0,0), (10,3,0) and (4,3,0); and, one plane located at the airport with coordinates (2,1,0).

A problem is an initial and incomplete HTN, i.e., a set of goal tasks. Planning is a process by which that initial HTN is completed resulting an abstract plan ready to be executed and incorporating alternative courses of action, i.e., it includes replanning procedures. Roughly speaking, this involves the following steps: first, the structure of the abstract plan (HTN) is built based on cases of past plans (this is closely related to the regular HTN planning procedure); then the conditional effects, probabilities are computed based on the primitive tasks of cases of past plans; the EU is computed for the primitive tasks of this abstract plan based on the procedural or non-procedural components of their effects; finally, these properties (conditional effects and respective probabilities, and EU) are propagated upward in the HTN, from the primitive tasks to the main task of the HTN. Figure 6 presents this algorithm.

Algorithm CONSTRUCT-ABSTRACT-PLAN(*abstPlan*)
abstPlan ← BUILD-STRUCTURE(*abstPlan*)
primTasks ← getPrimTasks(*abstPlan*)
primTasksAllPlanCases← getPrimTasksAllPlanCases()
COMPUT-PRIMTASKS-
PROPS(*primTasks*,*primTasksAllPlanCases*)
abstPlan←PROPAGAT-PROPS-UPWARD(*primTasks*,*abstPlan*)
return *abstPlan*
end

Figure 6 - Algorithm for the construction of an abstract plan.

3.1 Building the Structure of the Abstract Plan

Much like regular HTN planning, building the abstract plan is a process by which the initial HTN is completed through the recursively decomposition of its compound tasks. Unlike regular HTN planning, within our approach the domain theory (methods and operators in regular HTN planning) is confined to a finite set of actions/operators. Thus there are no explicit methods to describe how to decompose a task into a set of subtasks. Actually, methods are implicitly present in cases of past plans (see (Muñoz-Avila et al., 2001) for a similar approach). This is particularly useful in domains where there is no theory available. Therefore, the process of decomposing a task into subtasks is case-based and is performed as follows. Given a task, the possible alternative decompositions (task and its subtasks, as well as the links between them) are retrieved from cases of past plans. Two situations might happen. If there are more than one alternative decomposition, the given task is set as abstract and the set of decompositions are added to the HTN, linking each head task to the abstract task through a hierarchical link of type *abst*. Thus these head tasks are now the subtasks of the abstract task (see Figure 7 for an illustration of this process). On the other hand, if only one decomposition is retrieved, its subtasks are added as subtasks of the given task, linked by a hierarchical link of type *dcmp* (see Figure 8 for an illustration of this process). Whether a single decomposition or multiple decompositions are retrieved, the addition of it/them comprises an adaptation process (Kolodner, 1993), i.e., the retrieved decomposition(s) is/are changed if necessary so that it/they is/are consistent with the rest of the HTN.

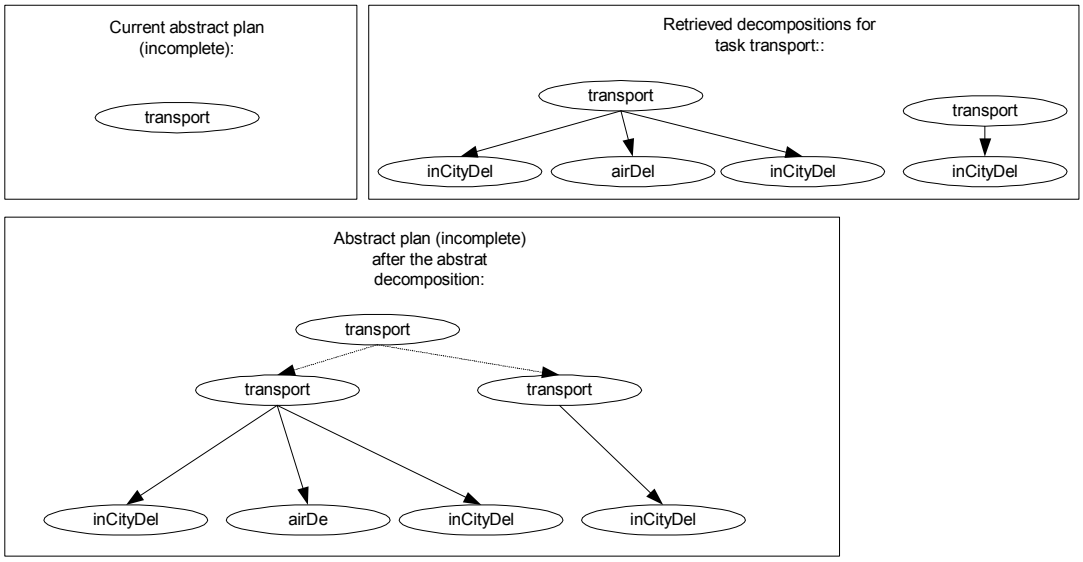

Figure 7 - Illustrative example of an OR-decomposition of an abstract task.

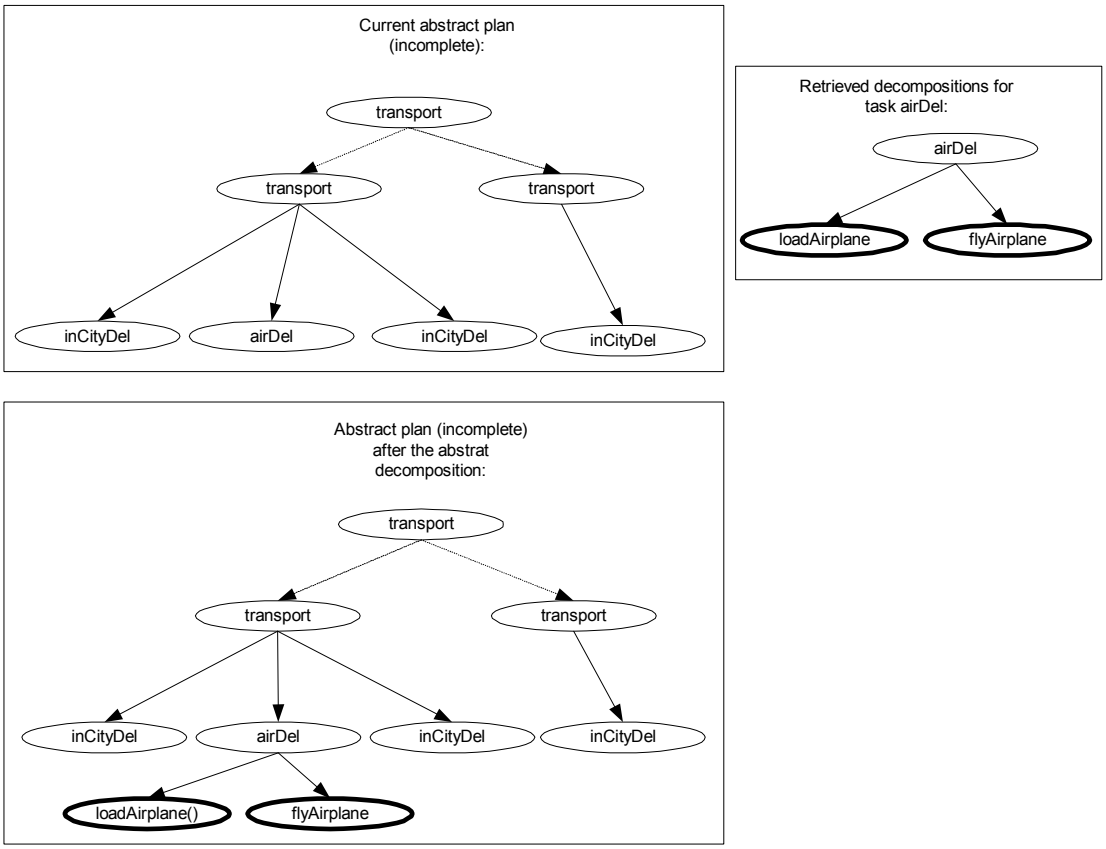

Figure 8 - Illustrative example of an AND-decomposition of a regular compound task.

The process of building the HTN ends when there is no more compound tasks to decompose, i.e., when the leaves of the tree are primitive tasks, or when there is no available decompositions in the case-base for at least one compound task.

Within our approach, a task belonging to an HTN has a probability value associated to it. This value expresses the probability of being executed given that its ancestor is executed. Thus, this probability is actually a conditional probability. Obviously, the probability of a task belonging to a case of a past plan is always 1.0 because it was executed (otherwise it won't belong to the case). The probability of the tasks belonging to an abstract plan is computed during the process of building the HTN as follows. Given the i^{th} subtask, ST_i, of a task T both belonging to an abstract plan, the probability of ST_i be executed given that T is executed is given by the conditional probability formula $P(ST_i/T)=\frac{P(ST_i \cap T)}{P(T)}$. Since within our approach there is no probabilistic model available, these probabilities have to be computed from data, i.e., from past occurrences of the tasks in cases of past plans, in the following manner. According to the frequency interpretation of probability, in r repetitions of an experiment, the value $P(X)$ is given by the number of times X occurred in the possible r times. This value is given by $S_r(X)/r$, where $S_r(X)$ denotes the absolute frequency of X (i.e., the number of times X occurred in the r repetitions of the experiment). As r increases, $S_r(X)/r$ converges to $P(X)$. In the context of HTN planning, the experiment should be understood as the decomposition of a task into subtasks. According to this frequentist approach of probability it can be shown that, $P(ST_i/T)=\frac{P(ST_i \cap T)}{P(T)}=\frac{S_r(ST_i \cap T)}{S_r(T)}$, when r is big. Thus, this expresses the number of times ST_i and T occurred together in the total amount of times T occurred, or in the context of HTN planning, this expresses the number of times ST_i was subtask of T in the total amount of times T was the task decomposed in past HTN plans. When ST_i is not a head of an alternative decomposition in the new plan (i.e., when T is not an abstract task), it means that T was always decomposed in the same way in past plans, i.e., into the same subtasks, which means ST_i occurred always when T occurred, otherwise ST_i won't be subtask of T. Thus, in this situation, the numerator and denominator of the above equation are equal and therefore $P(ST_i/T)$=1.0. However, when ST_i is a head of an alternative decomposition, it means

there were more than one way to decompose T in past plans, one of them being the decomposition headed by ST_i. Thus, counting the number of times the decomposition headed by ST_i was taken to decompose T, i.e., the number of times ST_i instantiated T, $S_r(ST_i \cap T)$, in all past plans and dividing this number by the number of times T was decomposed, i.e., $S_r(T)$, yields the value for $P(ST_i/T)$ for this situation.

After the abstract HTN is built, the conditional effects (and respective probabilities) and the EU are computed for the primitive tasks.

3.2 Motivation and Emotion-based Computation of the EU

As said above, a task T is both conditional and probabilistic (e.g.: (Blythe, 1999)). Thus, the execution of a goal task under a given condition may be seen according to Utility Theory as a lottery (Russel & Norvig, 1995):

$$Lottery(T) = \left[p^1 \times p_1^1, E_1^1; p^1 \times p_2^1, E_2^1; \ldots; p^m \times p_{n_m}^m, E_{n_m}^m \right]$$

, where p^i is the probability of the condition c_i, p_j^i is the probability of the j^{th} effect, E_j^i, of condition c_i.

The EU of T may be then computed as follows:

$$EU(T) = \sum_{k,j} p^k \times p_j^k \times EU(E_j^k)$$

The computation of $EU(E_j^k)$ is performed predicting the motivations that could be elicited by achieving/executing the goal task (Castelfranchi, Conte, Miceli, & Poggi, 1996; Reisenzein, 1996). We confined the set of motivations to surprise, curiosity and hunger[1]. As said above, two methods may be used for predicting the intensities of those motivations: based on the non-procedural component of the effects, or based on the procedural component.

If we take into account the procedural component of the effects, the intensities of surprise, curiosity and hunger felt by the agent when the effect takes place are estimated based on the information available in the effect about the changes produced in the world.

Surprise is given by (Macedo & Cardoso, 2001a):

$$SURPRISE(Agt, Obj_k) = UNEXPECTEDNESS(Obj_k, Agt(Mem)) = 1 - P(Obj_k)$$

, where Obj_k is the direct object of task T when E_j^k takes place, i.e., the entity that is visited (for the case of exploratory behaviour).

Curiosity is computed as follows (Macedo & Cardoso, 2001b):

$$CURIOSITY(Agt, Obj_k) = DIFFERENCE(Obj_k, Agt(Mem))$$

The measure of difference relies heavily on error correcting code theory (Hamming, 1950): the function computes the distance between two objects represented by graphs, counting the minimal number of changes (insertions and deletions of nodes and edges) required to transform one graph into another.

The drive hunger is defined as the need of a source of energy. Given the capacity C of the storage of that source, and L the amount of energy left ($L \leq C$), the hunger elicited in an agent is computed as follows:

$$HUNGER(Agt) = C - L$$

The following function is used to compute $EU(E_j^k)$:

$$EU(E_j^k) = \frac{\alpha_1 \times U_{surprise}(E_j^k) + \alpha_2 \times U_{curiosity}(E_j^k) + \alpha_2 \times U_{hunger}(E_j^k)}{\sum_i \alpha_i} = \frac{\alpha_1 \times SURPRISE(E_j^k) + \alpha_2 \times CURIOSITY(E_j^k) + \alpha_2 \times HUNGER(E_j^k)}{\sum_i \alpha_i}$$

, where, $\alpha_2 = -1$ and α_i ($i \neq 2$) may be defined as follows:

$$\alpha_i = \begin{cases} 1 \Leftarrow C - HUNGER(Agt) - D > 0 \\ 0 \Leftarrow otherwise \end{cases}$$

, where D is the amount of energy necessary to go from the end location of goal task T to the closer place where energy could be recharged, and C is the maximum amount of energy that could be stored by the agent.

If we take into account the non-procedural component of the effects, we avoid the computations of the intensities of the motivations. In fact, doing so, we are taking into account the intensities of the emotions, drives and other motivations in previous occurrences of the tasks and respective effects. This emotional/motivational information collected from previous occurrences of a task is a kind of Damásio's *somatic marker*. For this reason, tasks are called *somaticly-marked tasks*. When a task is about to occur again, the planning agent may compute its EU based on this data. In fact, this seems to be faster than the alternative approach of estimating the emotions that a task may elicit based on the values of the variables of the state of the world such as the time duration, fuel consumed, etc. Anyway, the same formula (present above) is used to compute $EU(E_j^k)$.

3.3 Propagation of the Properties Upward

After the primitive tasks have the conditional effects and respective probabilities, the probability and EU computed, these properties are propagated bottom-up (from primitive to non-primitive tasks), from the sub-

[1] The agents that make use of the planning approach described in this paper have been used to explore unknown environments, and to create things. Among motivations, surprise, curiosity and hunger have been closely related with this exploratory and creative behaviour (Berlyne, 1950; Boden, 1995; Izard, 1991).

tasks to the task of a decomposition and from the subtasks (heads of alternative decompositions) to the abstract task of an abstract decomposition). Notice however that the goal of this propagation is twofold: to complete the non-primitive tasks so that they can be ranked according to their EU when they are heads of alternative decompositions, and to know the overall EU of the abstract plan which is given by the EU of the main task of the plan.

4 Plan Execution and Replanning

Finding the optimal plan consists simply of traversing the abstract plan, selecting the most EU subtask of an abstract task. Backtracking occurs when an alternative decomposition fails execution. In this case, the next alternative decomposition that follows the previous in the EU ranking is selected for execution.

5 Related Work

Our work is closely related to HTN planning. This methology has been extensively used in planning systems such as UMCP (Erol et al., 1994), SHOP and SHOP2 (Nau et al., 2001). Unlike these planners, the planner presented in this paper don't use methods as part of the domain theory for task decomposition, but instead methods that are implicitly included in cases that describe previous planning problem solving experiences. SiN (Muñoz-Avila et al., 2001) also uses a case-based HTN planning algorithm, in which cases are instances of methods.

Among decision-theoretic planners, DRIPS (Haddawy & Doan, 1994) is probably the most closely related to the planner presented here. Actually, DRIPS shares a similar representation approach for abstract plans (an abstraction/decomposition hierarchy) and for actions. Besides, it also returns the optimal plan according to a given utility function. However, in contrast to DRIPS, in our planner the variant of a HTN that represents abstract plans is automatically built from cases and not given as input for the planning problem. Besides, it includes temporal, utility ranking and adaptation links in addition to decomposition links. Another major difference is that, in our planner, the EU of tasks and of alternative plans are computed when the abstract plan is built, while in DRIPS this occurs when the optimal plan is searched. Besides, in our planner, there is the possibility of computing the EU of tasks based on the non-procedural component of their effects, which avoids some additional computations at the cost of being less accurate. Moreover, finding the optimal plan in our planner consists simply of traversing the HTN with backtracking (or replanning) points located at the subtasks of an abstract task. In our planner the propagation of properties upward in the hierarchy is closely related with the approach taken in DRIPS for abstracting actions (Haddawy & Doan, 1994). A propagation of properties in the planning tree, bottom-up and left-to-right, is also used in GraphHTN (Lotem & Nau, 2000) in order to improve the search algorithm.

Another important work that addressed planning in agents inhabiting dynamic, uncertain environments is that of (Wilkins, Myers, & Wesley, 1994).

The relationship between emotions and plans has been considered previously by several authors (e.g.: (Bates, 1994; Gratch, 1999; Oatley & Johnson-Laird, 1987; Simon, 1967; Sloman, 1987)). Our main additional contribution to this works is considering somaticly-marked tasks (Damásio, 1994).

6 Conclusions and Future Work

We have presented an approach for decision-theoretic, HTN planning. In this approach emotions and motivations play a central role in that the EU of the tasks is based on the intensity of the emotions and other motivations they elicit. Two approaches have been proposed to compute the EU of tasks based on motivations: based on the procedural or non-procedural (factual) component of the effects of the tasks. The latter approach seems to be faster and is deeply related with Damásio's somatic-marker hypothesis. However, additional experiments are required to assess these ideas. In the future, we plan to perform such experiments.

Aknowledgements

The PhD of Luís Macedo is financially supported by PRODEP III.

References

Allen, J. (1983). Maintaining knowledge about temporal intervals. *Communications of the ACM, 26*(11), 832-- 843.

Bates, J. (1994). The Role of Emotion in Believable Agents. *Communications of the ACM, 37*(7), 122-125.

Berlyne, D. (1950). Novelty and curiosity as determinants of exploratory behavior. *British Journal of Psychology, 41*, 68-80.

Blythe, J. (1999). Decision-Theoretic Planning. *AI Magazine, Summer 1999*.

Boden, M. (1995). Creativity and unpredictability. *SEHR, 4*(2).

Castelfranchi, C., Conte, R., Miceli, M., & Poggi, I. (1996). Emotions and goals. In B. Kokinov (Ed.), *Perspectives on Cognitive Science* (pp. 131-145). Sofia: New Bulgarian University.

Damásio, A. (1994). *Descartes'error, Emotion Reason and the Human Brain*. New York: Grosset/Putnam Books.

Erol, K., Hendler, J., & Nau, D. (1994). UMCP: A sound and complete procedure for hierarchical task-network planning, *Proceedings of the International Conference on AI Planning Systems* (pp. 249-254).

Gratch, J. (1999). Why you should buy an emotional planner, *Proceedings of the Agents'99 Workshop on Emotion-based Agent Architectures*.

Haddawy, P., & Doan, A. (1994). Abstracting probabilistic actions, *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence* (pp. 270-277). San Mateo, CA: Morgan Kaufmann.

Hamming, R. (1950). Error Detecting and Error Correcting Codes. *The Bell System Technical Journal, 26*(2), 147-160.

Izard, C. (1991). *The Psychology of Emotions*. NY: Plenum Press.

Kolodner, J. (1993). *Case-Based Reasoning*. San Mateo, CA: Morgan-Kaufmann.

LeDoux, J. (1996). *The Emotional Brain*. New York: Simon and Schuster.

Littman, M., & Majercik, S. (1997). Large-Scale Planning Under Uncertainty: A Survey, *Workshop on Planning and Scheduling for Space* (pp. 27:21--28).

Lotem, A., & Nau, D. (2000). New advances in GraphHTN: Identifying independent subproblems in large HTN domains, *dings of the International Conference on AI Planning Systems* (pp. 206-215).

Macedo, L., & Cardoso, A. (1998). Nested-Graph structured representations for cases. In B. Smyth & P. Cunningham (Eds.), *Advances in Case-Based Reasoning - Proceedings of the 4th European Workshop on Case-Based Reasoning* (Vol. 1488, pp. 1-12). Berlin: Springer-Verlag.

Macedo, L., & Cardoso, A. (2001a). Modelling Forms of Surprise in an Artificial Agent. In J. Moore & K. Stenning (Eds.), *Proceedings of the 23rd Annual Conference of the Cognitive Science Society* (pp. 588-593). Mahwah, NJ: Erlbaum.

Macedo, L., & Cardoso, A. (2001b). SC-EUNE - Surprise/Curiosity-based Exploration of Uncertain and Unknown Environments., *Proceedings of the AISB'01 Symposium on Emotion, Cognition and Affective Computing* (pp. 73-81). York, UK: University of York.

Meyer, W., Reisenzein, R., & Schützwohl, A. (1997). Towards a process analysis of emotions: The case of surprise. *Motivation and Emotion, 21*, 251-274.

Muñoz-Avila, H., Aha, D., Nau, D., Breslow, L., Weber, R., & Yamal, F. (2001). SiN: Integrating Case-based Reasoning with Task Decomposition, *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI-2001)*. Seattle, WA: Morgan Kaufmann.

Nau, D., Muñoz-Avila, H., Cao, Y., Lotem, A., & Mitchell, S. (2001). Total-order planning with partially ordered subtasks, *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*. Seattle, WA: Morgan Kaufmann.

Oatley, K., & Johnson-Laird, P. (1987). Towards a cognitive theory of emotions, cognition and emotion. *Cognition and Emotion, 1*(1), 29-50.

Ortony, A., & Partridge, D. (1987). Surprisingness and Expectation Failure: What's the Difference?, *Proceedings of the 10th International Joint Conference on Artificial Intelligence* (pp. 106-108). Los Altos, CA: Morgan Kaufmann.

Reisenzein, R. (1996). Emotional Action Generation. In W. Battmann & S. Dutke (Eds.), *Processes of the molar regulation of behavior*. Lengerich: Pabst Science Publishers.

Reisenzein, R. (2000). The subjective experience of surprise. In H. Bless & J. Forgas (Eds.), *The message within: The role of subjective experience in social cognition and behavior*. Philadelphia, PA: Psychology Press.

Russel, S., & Norvig, P. (1995). *Artificial Intelligence - A Modern Approach*. Englewood Cliffs, NJ: Prentice Hall.

Simon, H. (1967). Motivational and emotional controls of cognition. *Psychological Review, 74*, 29-39.

Sloman, A. (1987). Motives, mechanisms and emotions. *Cognition and Emotion, 1*, 217-234.

Thrun, S. (2002). Robotic mapping: A survey. In G. Lakemeyer & B. Nebel (Eds.), *Exploring Artificial Intelligence in the New Millenium*. San Mateo, CA: Morgan Kaufmann.

Wilkins, D., Myers, K., & Wesley, L. (1994). Cypress: Planning and Reacting under Uncertainity. In M. Burstein (Ed.), *ARPA/Rome Laboratory Planning and Scheduling Initiative Workshop Proceedings* (pp. 111-120). San Mateo, CA: Morgan Kaufmann Publishers Inc.

Younes, H. (2003). Extending PDDL to model stochastic decision processes, *Proceedings of the ICAPS-02 Workshop on PDDL*.