Assisting Elders via Dynamic Multi-tasks Planning – A Markov Decision Processes based Approach

François Courtemanche
Department of Computer Science
University of Sherbrooke
J1K 2R1 Quebec, Canada
1 819 821 8000 - 63000
f.courtemanche@usherbrooke.ca

Mehdi Najjar
Department of Computer Science
University of Sherbrooke
J1K 2R1 Quebec, Canada
1 819 821 8000 - 63000
mehdi.najjar@usherbrooke.ca

Blandine Paccoud
DOMUS Research Lab
University of Sherbrooke
J1K 2R1 Quebec, Canada
1 819 821 8000 - 63825
blandine.paccoud@usherbrooke.ca

André Mayers
Department of Computer Science
University of Sherbrooke
J1K 2R1 Quebec, Canada
1 819 821 8000 - 62041
andre.mayers@usherbrooke.ca

ABSTRACT
This paper presents a novel planning approach to assist elders with memory deficit to carry out complex daily activities. The proposed planner uses Markov decision processes (MDPs) in dynamic multi-tasks planning to help memory-impaired elders achieving and finalising their activities of daily living (ADLs) already undertaken. The article also reports empirical results of the experimental validation and discusses distinctions between our approach and related works.

Categories and Subject Descriptors
I.2.1 [**Computing Methodologies**]: Artificial Intelligence, Planning .

General Terms
Algorithms, Design, Experimentation, Human Factors, Theory.

Keywords
Ambiant Intelligence, Smart Assistance, Activity of Daily Living, Multi-Tasks Planning, Markov Decision Process.

1. INTRODUCTION
Since the seventies, assistance in smart homes has been defined as a support to occupiers for the completion of their activities of daily living (ADLs) [4] and particular attention was made for cognitively-impaired people. Nowadays, the improvement of the life quality in the developed nations has systematically generated an increase of the life expectancy. Nevertheless, the increasing number of elderly person implies more resources for aftercare, paramedical care and natural assistance in their habitats. The situation further complicated if elders suffer from memory disorders [6]. In this case a permanent assistance is necessary whenever they are. However, and in order to grant to the memory impaired elder a minimal freedom to act in a not-alarming situation, an intelligent assistance must not dictate her/him what to do according to a predefined static stereotyped behavioural plan, but should adapt dynamically suggested plans depending on what the elderly occupier wants to make. A consequence of this starting premise is the necessity of a prior knowledge about the ADLs and the environment, but also – and especially – a constantly inferred knowledge about the occupier intentions.

This paper presents a novel planning approach to assist elders with memory troubles to carry out ADLs. The proposed planner uses Markov Decision Processes (MDPs) in dynamic multi-tasks planning to help memory-impaired elders achieving and finalising their ADLs already undertaken. The remainder of the article is organised as follow. In section 2, we expound the theoretical background of MDPs. Section 3 presents our modular architecture that contains various modules permitting to the planner (which also takes part of the architecture) to act according to the proposed dynamic approach. The fourth section is devoted to a detailed description of the planning module functioning. After reporting on experimental validation and presenting empirical results in section 5, originalities of the proposed planning approach are addressed in the sixth section. We also discuss relations and distinctions between our approach and other related works. Concluding remarks are given in section 7.

Ambi-sys 2008, February 11-14, 2008, Quebec, Canada.

2. THEORETICAL BACKGROUND

To give a definition of MDPs, we take up the introductive proposal of Dietterich [3] by considering the general AI problem in which an agent interact with an environment. At each time step, the agent observes the environment, chooses and executes an action and receives a real value reward. The goal of the agent is to choose actions in order to maximise the sum of these rewards. For the case where the agent can perceive the entire state of the environment and where the actions are stochastic (i.e., the state resulting from an action is a probabilistic function of the previous state and the chosen action), the resulting sequential decision problem is known as a Markov decision process (MDP) [7].

A MDP is a 4-tuple <S, A, P, R> where S is a finite set of states, A is a finite set of actions, P is a transition distribution P: S × A × S → [0,1] such that P(s,a,s') is a probability distribution over S for any s Є S and a Є A; and R: S × A → IR is a bounded reward function. Intuitively, P(s,a,s') denotes the probability of moving to state s' when action a is performed at state s, while R(s,a) is the immediate reward associated with the resulting transition.

Because MDPs provide a very general model of sequential decision-making under uncertainty, they have provided a foundation of much recent work on probabilistic planning where it is assumed the agent knows the state transition distribution P and the reward function R. Thus, given a MDP, the objective is to construct a policy that maximises the expected accumulated reward over some horizon of interest. I. e., to infer a policy π: S → A which indicates for each state s Є S what action a Є A should be performed. The optimal policy π* maximises the expected cumulative reward received by the agent. There are many algorithms for finding optimal policies, such as the dynamic programming algorithms value iteration and policy iteration [1]. Most of these algorithms involve computing a function known as the value function V(s) which estimates the expected cumulative reward of starting in state s and following the optimal policy. Formally, the value function is given by the Bellman equation :

$$V(s) := \max_a \sum_{s'} P(s'|s,a)[R(s'|s,a) + V(s')]$$

A simple variant of the value iteration algorithm is expressed by Algorithm 1 illustrated in figure 1.

$V_0(s) \leftarrow 0 \quad \forall s \in \mathcal{S}$
$V_1(s) \leftarrow R(s,a) \quad \forall s \in \mathcal{S} \quad \forall a \in \mathcal{A}$
$t \leftarrow 1$
while $\exists s \in S$ such that $|V_t(s) - V_{t-1}(s)| > \epsilon$
  forall $s \in \mathcal{S}$
    forall $a \in \mathcal{A}$
      $Q(s,a) \leftarrow R(s,a) + \gamma \sum_{s' \in S} T(s,a,s') V_{t-1}(s')$
    $V_t(s) \leftarrow \max_a Q(s,a)$
    $\pi(s) \leftarrow \arg\max_a Q(s,a)$
  $t \leftarrow t+1$

Figure 1. The value iteration algorithm.

3. THE MODULAR ARCHITECTURE

The modular architecture is composed of five (5) modules: the scenarios generation module (SGM), the recognition module (RM), the analysis module (AM), the diagnosis module (DM) and the planning module (PM). These modules operate and handle shared data represented in XML structures. The modules communicate by messages. A data base gathers randomly generated scenarios (via the SGM) which are used during simulations for the reconstitution of real events. Figure 2 illustrates the general view of the architecture.

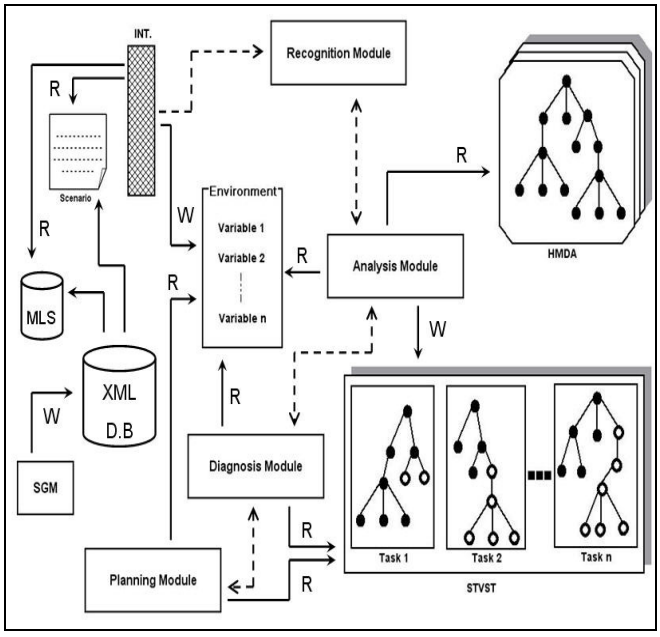

Figure 2. General view of the architecture.

The overall environment is an accessible structure. Data in the environment allow the modules to reason in order to properly achieve their functionalities. Modifying the

environment is done via an exclusive write access granted to only one component of the architecture (this detail will be clarified below). To make the environment easier to consult, reading (R) is managed by a circular order. This can change if a module notifies another and requests its intervention when detecting anomalies or critical situations. The various modules of the architecture reason on data that are collected by parsing an XML file which contains the scenario. As mentioned above, the letter represents a set of events reconstructing simultaneous achievements of several daily activities. Although they are generated randomly, these events are selected according to logical constraints. For example, for the scenario of the realisation of the "coffee preparation" activity, if the events "coffee-maker alarm is ringing", "coffee cup is full" and "sugar bowl is open" were chronologically generated, then when the event "coffee cup is clean" is generated, it will not be accepted in the scenario. On the other hand, the event "holding the coffee spoon" will be added to the scenario.

An interpreter of scenarios (INT) scans the XML file and – at each temporal unit – sends information relating to one event to the recognition module (RM). Having the exclusive write access (W) on the environment, the interpreter updates it. The environment is a structure that recreates what really occurs in the intelligent apartment thanks to variables which represent all the detectable housing elements (e.g., furniture, kitchen utensils, taps, household electric appliances, etc). In a real context, the state of each detectable element is determined by a sensor.

3.1 Memorising Activities

The set of temporary views of started tasks (STVST) allows to constantly informing on the level of progression of each task in terms of accomplished achievements of its sub-tasks (recursively, of the sub-sub-tasks of each sub-task). The STVST is updated by the analysis module (AM) and consulted by the diagnosis module (DM). The hierarchical model of the defined activities (HMDA) defines, for each activity, its hierarchical structure, the concerned variables of the environment and their final states when the activity is accomplished. The HMDA also takes account of constraints of order, sequence and possible mutual exclusivenesses between sub-tasks of an activity.

3.2 The Recognition Module

The recognition of activities is based on data that simulate information transmitted by sensors in the intelligent apartment. At each unit of time and for each sent datum, a Hidden Markov Model (HMM) of the recognition (incorporated into the RM) identifies the activity in progress [8]. This is done relying on the probability calculus of transitions between the defined states of the HMM. Probabilities of transitions are calculated and refined during the learning stage. This is generally done at the beginning of the simulation. An automatic training of the HMM is carried out by means of typical scenarios stored in the machine learning scenarios database (MLS). In order to reinforce learning, some generated scenarios of the SGM, that present particular cases, are also added to the MLS. Once an activity is recognised, the RM notifies the analysis module (AM). This examines the sequence of activities. It is based on the hierarchical model of the defined activities (HMDA) which links up each activity to its sub-tasks. In a recursive fashion, each sub-task is divided into an ordered set of "sub-subtasks". This recursion stops when components of a sub-task represent the perceived events that are transposed in the environment.

3.3 The Analysis Module

The AM works symbiotically and in a bidirectional way with the RM. At each temporal unit, the latter notifies the analyser on a change in an activity in progress or a creation of a new one. The AM consults the environment to update the partial state of the activity. When detecting a new activity, the AM creates a view, initialises it and adds it to the set of temporary views of started tasks (STVST). The analysis consists of (1) scouring in the HMDA the hierarchical structure of the activity in a bottom/up way, starting from the leaf nodes (the non-decomposable events) in order to update one STVST partial view and (2) informing the diagnosis module (DM) of the last change.

3.4 The Diagnosis Module

The DM possess three (3) functionalities: (1) it evaluates the impact of the last detected event and decides if necessary measures have to be taken, (2) communicates its decisions to the planning module (PM) and (3) reports critical cases via the interface when detecting anomalies in the behaviour of the simulated person. The impact of an event is measured with regards to the change introduced in the activity in progress and in relation to the possible constraints imposed by the other already started activities. A recent event can develop an activity, suspend another, create a new one, bring about its resumption or belong to a set of disparate and singular events that are not defined in the HMDA. The impact is evaluated regarding to the partial states of all started tasks. A critical situation is detected when at least one constraint imposed by an activity in progress is violated. A constraint can be of chronological nature. For example, if the DM notes a time exceeding for the realisation of a sub-task or for the suspension of an activity. Constraints can also represent relations of sequences dependence. If no critical situation is detected, the DM simply notifies the PM. The planning module is deeply depicted in the next section.

4. THE PLANNING MODULE

The planning module (PM) takes as input information provided by the analysis module. It aims to maintain the

order of the undertaken sub-activities as it is defined in the HMDA. At each unit of time, the PM receives an ordered list of sub-activities (for example, as illustrated in figure 3) for which it must provide a resolution optimal plan.

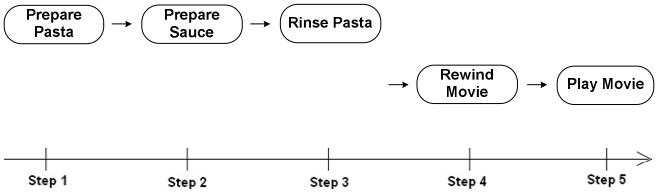

Figure 3 – An example of an ordered list of sub-activities given to the planner in input.

A first stage consists in sorting this list to optimize the latency between sub-activities. For example, – and referring to figure 3 – if the minimum time required between achieving preparing pasta and starting rinsing pasta is about 10 minutes and if preparing tomato sauce takes about 7 minutes, then there is a slack period (3 minutes) between the two last sub-activities. Thus, the PM moves a subsequent sub-activity towards this place to fill the latency (see figure 4). In this sense, the planner respects the order of the activities undertaken by the occupant but finds the best way of interleaving their achievements. In a real context, this kind of temporal dependence between sub-activities is frequent when realising activities of the daily living (ADL), our hierarchical decomposition of activities allows a more profitable planning in terms of duration of the suggested plan.

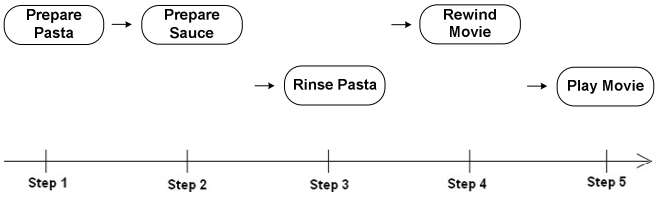

Figure 4 – Interleaving sub-activities thanks to latency optimisation.

The second stage consists in finding an optimal sequence of actions to achieve each sub-activity starting from the current state of the environment (variables' values at a given time t). The planner applies the Value Iteration algorithm [1] to find a partial plan related to each sub-activity. The Value Iteration algorithm aims to find the minimal set of actions leading from the initial state of the activity (values of the variables of the environment at t) to a final state. Figure 5 shows the proposed (by the planner) optimal sequence of actions to carry out each one of the sub-activities started by the elder.

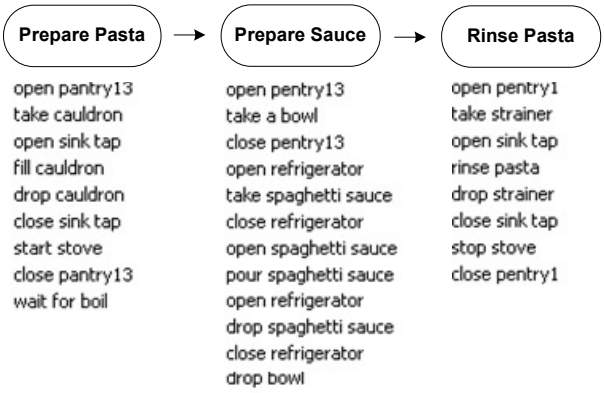

Figure 5 – An example of optimal sequence of actions.

During the third stage, all the partial plans are assembled to form the total plan. Figure 6 shows the suggested series of actions allowing the elderly person to finish all the started activities. This plan is valid at t, according to the current intentions of the occupier. At t+1, the list of the activities, provided to the planning module (by the analysis module) will be modified according to the last action undertaken by the person. This modified list reflects the inferences made by the recognition module on data of the environment to determine the current activity in progress.

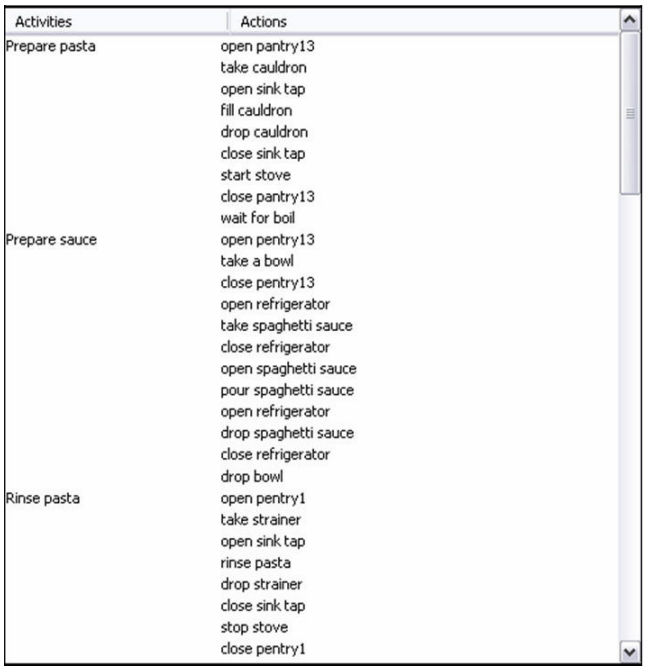

Figure 6 – The suggested series of actions at t to finish all the started activities.

Since the occupier of the intelligent apartment is completely free to follow the proposed plan or to ignore it, the system must be reactive and able to be readjusted at any time to provide assistance to the elder. Thus, this dynamic approach of planning allows a real-time monitoring and assisting a subject evolving/moving in its environment.

5. EXPERIMENTAL VALIDATION

To validate our approach we have done multiple series of simulations representing different sequences of activities a subject can proceed to. Figure 7 shows diagrams illustrating the duration of the total plans proposed to the user in relation to their complexity. The complexity represents the number of simultaneous activities (each one composed of sub-activities) carried out by the subject. For each complexity level the activities where chosen randomly. The duration is expressed in terms of time unit needed to execute each of the multiple actions dictated by the plans. The ordinate axe represents the time unit mean of all series of 250 simulations done for each complexity level. It enables us to appreciate the reduction of plans duration by the optimization algorithm. The variations between the two columns of each complexity level represent mainly the time spared by reducing latencies between sub-activities. This is the effect of a smart parallelism in the realisation of activities. The randomly chosen series of activities highlight the planning dynamicity because it represents all choices a user can make over time (this will be explained in more details in section 6.3).

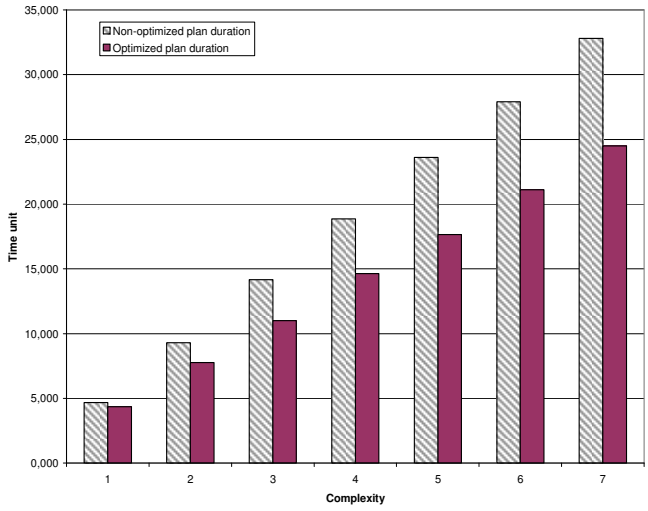

Figure 7 – The duration of the proposed plans in relation to their complexity.

Figure 8 shows the plan duration improvement brought about the proposed optimization of the latencies. The linear evolution of the gain with regard to the complexity of the plans is expressed by the third curve. The amount of having gains on plan duration is conceivable as the complexity raises. This is explained by the fact that opportunities to optimise latencies grows with the number of activities (and sub-activities) carried out at the same time by the subject. The more the subject is undertaking activities simultaneously, the more the optimisation algorithm is effective.

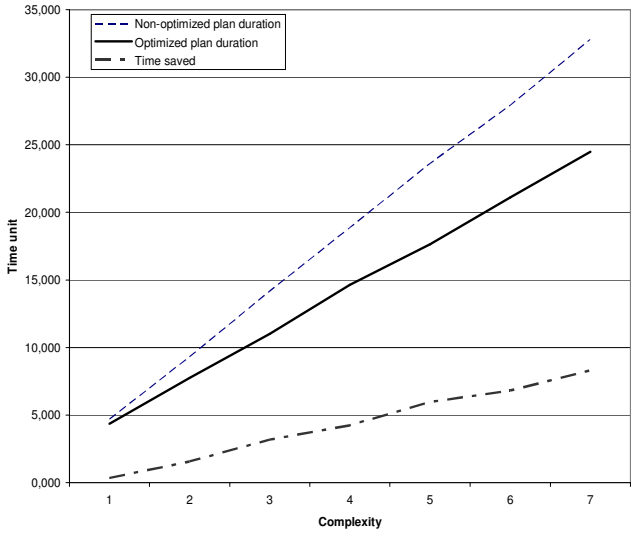

Figure 8 – The plan duration improvement brought about the optimization of latencies.

6. DISCUSSION

As mentioned before, this section discusses relations and distinctions between our planning approach and other related works. Originalities of our planner are also addressed.

6.1 The MDP Partitions and Sub-MDPs Connections

In order to solve a planning problem in terms of actions plan, Dean & Lin [2] propose some techniques of (i) regions' partitioning (where each region represents a sub-problem) and (ii) combination of partial solutions whose each one would result from a partition. These techniques (DTPSD, for Decomposition Techniques for Planning in Stochastic Domains) rest on the use of parameters describing the relation between a partition R (a sub-MDP resulting from the decomposition of the initial problem. i.e. the full MDP) and its neighbours. These parameters are initially defined and remain unchanged since the structure of the problem never changes. Thus, the problem to be solved is seen as a task to achieve (mono-task) which does not take account of the notion of time. In our approach, each partition R does not have fixed and predefined neighbours. These will change constantly during the resolution of the problem which, in this case, consists of several concurrent tasks (multi-tasks). The vicinity of a partition is dynamically given according to the interleaving of the various tasks in progress and whose simultaneous achievements progress in time. Whereas our planning approach proceeds chronologically, DTPSD is timeless and static. Partitions in DTPSD share the same variables of environment but they do not share the values of these

variables. The coherence of each variable value (for example, Cupboard#2-Open = True) for the various partitions is maintained thanks to static transitions [2]. Our dynamic partitioning can not be possible via this type of transitions. In our case, the DTPSD strong link notion is replaced by a weak connection: the share out by all partitions (sub-MDPs) of a pool of environment variables which allow specifying at any time the internal states of these variables. The coherence of the variables' values is maintained using a general structure – that we call the environment (see figure 2) – containing the values of all the variables. As these values can change at each unit of time, each partition consults the environment to know in which state the variables are. This lead to assuring the coherence while allowing a dynamic partitioning.

Meuleau et al. [5] present a technique for computing approximately optimal solutions to stochastic resource allocation problems modeled as MDPs. Their approach exploits two key properties to avoid explicitly enumerating the very large state and action spaces associated with these problems. First, the problems are composed of multiple tasks whose utilities are independent. Second, the actions taken with respect to (or resources allocated to) a task do not influence the status of any other task. Each task is therefore viewed as an MDP. However, these MDPs are weakly coupled by resource constraints: actions selected for one MDP restrict the actions available to others. In addition, the second property makes the approach unusable for a dynamic partitioning with various levels of dependences.

6.2 Scheduling Partial Plans

Scheduling partial plans allows developing a total plan which takes into account – intelligently – the various constraints related to the problem. More precisely, in smart homes and in multi-goals planning context, scheduling is used to plan simultaneous realisations of several activities while ensuring that the plan will not be confusing nor too complex for a memory or cognitively impaired person. For example, if the two activities to be achieved are <to watch a movie> and <to make Pasta>, then it is necessary to avoid a plan which would repetitively send the elderly from one room to another. i.e., <turning on the VCR> → <Putting on the stove> → <Rewinding the video tape> → <Taking a cauldron>, etc. In this sense, the necessity of partial plans is crucial in multi-goals planning and their scheduling makes it possible to benefit from relations between certain goals.

MaxQ [3] offers a tree graphical notation for describing the goal/subgoals structure of a task. The MaxQ tree contains two kinds of nodes: Max nodes and Q nodes. Max nodes with no children denote primitive actions. Those with children represent subtasks. The immediate children of each Max node are Q nodes. Each Q node represents an action that can be performed to achieve its parent's subtask. Figure 9 shows a MaxQ graph for the taxi domain mentioned in [3]. MaxQ does not consider the scheduling of partial plans, since the approach suppose one total static goal related to the achievement of one task. The MaxQ tree never changes. Its corresponding task has always the same fixed sequence of sub-activities. Actions represented by the Max nodes (under each Q node) can define the contents of partial plans. However, since the tree is fixed, there cannot be scheduling of partial plans. In addition, MaxQ is Q-Learning based. This learning technique determines a priori the values of the parameters of each sub-MDP (probabilities of transition and rewards). This is possible only if the hierarchy of the Max nodes and the Q nodes is fixed (see [3] for more details). If the tree changed in time, then all calculations would have to be started again. This static structure allowing the Q-Learning cannot be applied in multi-tasks planning that requires scheduling of the partial plans.

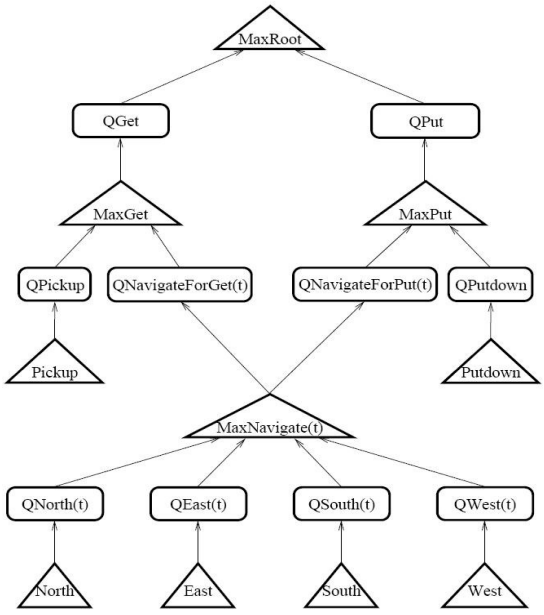

Figure 9 – The MAXQ graph for the taxi domain

The tree-based organisation is useful and serves two main goals which are consequently dependent. The first goal is how to give more coherence to the whole plan. The second is about optimizing the plan by managing the latencies. To broach the first goal, let us suppose that the general context of the problem is to spend one ordinary day in an intelligent habitat (approximately a hundred of environment variables which reflect the states of realisation of ten or so activities). This problem is broken up into various partitions. Each one of them is split up in turn in several sub-regions. Finally, this leads to a significant number of partitions. Since the topology of our partitioning is dynamic (i.e., the neighbours of an area change constantly), if we wish to avoid synthesizing senseless plans (for example, an unreasonable

simultaneous achievement of activities) it is necessary to establish temporary link between some partitions (for example, those belonging to the same activity). In this sense, the tree-based structuring of activities is used to solve the problem of the temporal abstraction [3]. The tree structure is useful to determine which partitions must be dependent between them for a certain time (the reader can refer to [3, p.4-6] for more details and explanations). As mentioned above, the second goal is to optimize the plan by an effective management of the latencies. Planning the complete realisation of an activity before passing to the achievement of another does not allow benefiting from idle times between sub-activities of the same activity. For example, waiting for boiling to infuse tea. We associate a tree to our MDPs in order to add a hierarchical aspect to the topology of the problem decomposition. This aspect makes it possible to interlace the achievements of sub-activities of several activities in progress (for example, to rewind a video tape while water heats in the kettle and pasta cook on the stove). This offers a finer and more intelligent treatment in planning.

6.3 The Optimality of Plans

The last addressed key issue leads us to argue about the objective to find the shortest path in planning. MaxQ introduces MDPs-based models within the framework of a stochastic problem of finding the shortest path in planning. Our approach uses MDPs within the framework of decision-making in multitasks realisation context. For MaxQ and DTPSD, the shortest path problem consists in finding by algorithmic means which local plan (policy) to choose for each partition so that the joint of these regional plans gives a total plan going from the first partition to the last using the minimum of possible actions. In dynamic multi-tasks planning where the solution of the problem consists in satisfying a set of goals, the notion of "from the first partition towards the last" has less of sense. Here the objective is to provide a total plan allowing the satisfaction of several simultaneous goals. Since the topology of the partitioning varies according to the current states of the subgoals and, hierarchically, according to those of the sub-subgoals (because of the dynamicity in time) we cannot consider a fixed state space as for DTPSD and MaxQ. Hence, it is necessary to make more flexible the representation of the states[1] (as it is impossible to have variables of transition) what involves in some cases few actions in excess (for example, opening and closing the same cupboard twice). But this is not really disadvantageous, since multi-tasks planning does not have the same objective as the shortest path problem. However, dynamic multi-tasks planning makes more difficult the shortest path search. Each time that a sub-MDP seeks the best sequence of actions to carry out for planning a sub-activity, a part of the set of the beliefs related to the problem is hidden from it. For example, a sub-MDP does not know preconditions of its neighbour sub-MDPs because, due to the dynamic partitioning, it cannot know its neighbours in advance. Thus, a sub-MDP cannot be able to consider the effects that its proposed partial plan will have on the others partial plans computed by its neighbour sub-MDPs. Vis-à-vis the compromise between the quality of the solution and the flexibility of the representation, this choice seems quiet justified by the nature of the targeted people (cognitively impaired elders). Insofar as we want to give the possibility to elders to interrupt/resume and to change activities dynamically, it is tolerable that sometimes an action involves additional actions when realising later activities.

6.4 Originalities

The planning approaches – as DTSPD – aim to conceive an optimal plan that the user will have to follow to achieve a goal initially defined. Our approach reverses the dependence between the planning and the user. It is the planner who adjusts its suggested plan regarding to the user's needs; and not the opposite. Our objective is to assist an elder moving/evolving within a controlled environment in order to provide him/her a logistic support in the realisation of its activities of daily living (ADL). Thus, the planning philosophy is to provide a plan which is always a function of the multiple current intentions of the occupier. This latter is free to follow or not the step proposed by the planner. If s/he does not do it, then the plan will have to be readjusted according to the choice of the person. This leads to adding new goals and/or changing the order of the realisation of existing sub-goals. In this sense, we think that our approach has innovated in the context of the dynamicity of goals and the flexibility of the planning representation.

Our first originality lies in the dynamic goals treatment. The DTPSD planning approach is functional only in a mono-task context. Thus, the key issue of dynamic goals is not addressed. In the MaxQ approach, the structure of the hierarchical tree cannot change without having to start all calculations. In our approach, it is possible to include new goals (new tasks/activities) in the planning process, to withdraw others and/or to change their order. This is done in real time without need for starting again any algorithms.

The second originality relates to the dynamic partitioning. As mentioned above, we provide a solution to the impossibility of making fixed and predefined partitioning according to a DTPSD-based philosophy in dynamic multi-

[1] To define the possible states of an activity, our planning approach only uses variables of environment that are relevant to the activity (or even to its sub-activities). Indeed, we cannot define nor make use of concepts such as the border or the periphery of region (activity), as in DTPSD.

tasks planning; because the topography (i.e., the repartition of the neighbours regions) is variable. For example – and thanks to the optimisation caused by the effective management of the latencies – a sub-activity, such as <to make boil water> can as well be followed by <to prepare the tomato salsa> or by <to rewind a video-cassette>.

7. CONCLUSION

We have presented a novel dynamic multi-tasks planning approach to assist memory-impaired elder to carry out complex activities of daily living (ADLs). The planner uses Markov Decision Processes (MDPs) to occupiers of a smart home achieving and finalising their ADLs already undertaken. In addition, our planning module adapts dynamically suggested plans depending on what the elderly person wants to do in a not-alarming situation context. We are currently investigating a new idea for integrating Partially Observable MDPs into the planner. Detailed aspects of this research work and its experimental validation will be presented in future papers.

8. ACKNOWLEDGMENTS

The authors want to thank (1) Jérémy Bauchet for his agreement on the use of the Actirec environement for the realisation of the graphical simulator of the intelligent ambient, (2) Alexandre Dion, Wojtek Jurewicz, Gilbert Samson, Jonathan Berriault and François Lizotte for their help on the realisation of the modular architecture.

9. REFERENCES

[1] Bellman, R. E. *Dynamic Programming*. Princeton University Press, 1957.

[2] Dean, T., Lin, S.-H. *Decomposition techniques for planning in stochastic domains.* TR CS-95-10, Department of Computer Science, Brown University, Providence, Rhode Island 02912, USA. 1995.

[3] Dietterich, T. G., The MAXQ Method for Hierarchical Reinforcement Learning. *In Proceedings of the Fifteenth International Conference on Machine Learning*, 118-126, 1998.

[4] Katz, S., Ford, A. B., Moskowitz, R. W., Jckson, B. A. and Jaffe, M. W. Studies of illness in the aged. The Index of ADL: a standardised measure of biological and psychosocial function. *Journal of the American Medical Association* 185, 914-919, 1963.

[5] Meuleau, N., Hauskrecht, K.-E. K., Peshkin, L., Pack Kaelbling, L., Dean, T. and Boutilier C. Solving very large weakly coupled Markov decision processes. *In Proceedings of the fifteenth national/tenth conference on Artificial intelligence/Innovative applications of artificial intelligence*, 165 - 172, 1998.

[6] Pigot, H., Bauchet, J. and Giroux, S. Assisted devices for people with cognitive impairment. *In Technology for aging disability and independence*. Vol 2. Wiley and sons (Eds.) 2006.

[7] Puterman, M. L. Markov Decision Processes: Discrete Stochastic Dynamic Programming. J. Wiley & Sons, New York, 1994.

[8] Rabiner, L. R. A tutorial on Hidden Markov Models and selected Applications in Speech Recognition. *In Proceedings of the IEEE*, 77(2), p.257-288, 1989.