A SCHEDULING TECHNIQUE OF PLANS WITH PROBABILITY AND TEMPORAL CONSTRAINTS

Bassam Baki and Maroua Bouzid
GREYC
University of Caen
Bd Marchal Juin
14032 Caen Cedex

Keywords: Decision support systems, Temporal Constraint Reasoning, Planning, AND/OR Graph and probability.

Abstract: In this paper, we address the problem of scheduling plans with probability and temporal constraints. We illustrate our problem with an AND/OR graph, where we try to find a plan of tasks that satisfies all temporal constraints and precedence relations between tasks, has a high probability of execution, a minimal cost and a reduced time. Each task has a set of temporal constraints, a set of probabilities and a set of constant costs. Our planner uses the temporal constraint propagation technique to simplify the resolution of a given problem.
We describe one approach to deal with a problem that has paid a little attention of planing community. This problem is to combine temporal and probabilistic planning.

1 INTRODUCTION

While today's planners can handle large problems with durative actions and time constraints, few researches have considered the problem concerning uncertainty and the different durations of actions. For example, IxTeT ((Ghallab and Laruelle, 1994), (Laborie and Ghallab, 1995)) is a time-map manager that handles symbolic constraints (precedence), numeric constraints (intervals) and sharable resources. This planner provides an initial complete plan which is then run by temporal executive following a cycle: integrate external messages, repair the plan if needed, decide which actions to execute. Other planners, like "C-Buridan" ((Draper et al., 1994), (Kushmerick et al., 1994)) deal with probabilistic information-producing actions and contingent execution. their algorithm is sound and complete but they do not consider any kind of constraints between tasks. Their solutions are satisfying rather than optimal.

Many existing works have been developed on temporal planning or probabilistic planning but little attention has been paid to the combination of these two planning techniques.

In this paper, we address the problem of scheduling plans with probability and temporal constraints. We illustrate our problem with an AND/OR graph, where we try to find a plan of tasks that satisfies all temporal constraints and precedence relations between tasks, has a high probability of execution, a minimal cost and a reduced time.

This problem consists in finding a set of tasks to be executed respecting all temporal constraints and precedence relations between tasks. By temporal constraints we mean the start times, the end times and possible execution durations of tasks. By precedence relation we mean the order of execution of tasks and the delays between tasks. Our planner uses the temporal constraint propagation technique (Bresina and Washington, 2000) to simplify the resolution of a given problem. Practically, that can consist in withdrawing the values which do not belong to any solution. This filtering avoids many attempts at resolution which are likely to fail.

In the following, we consider that an agent is concerned with achieving a set of tasks connected together in a network that forms an acyclic directed AND/OR graph where nodes are tasks and edges correspond to precedence relations. Given a set of tasks, a start time, an end time, a set of durations, a set of probabilities and a set of constant costs for each task, a time delay and a precedence relation between tasks, our approach allows the agent to determine the set of tasks to be executed. The goal will be achieved and all temporal and precedence constraints will be satisfied in order to guarantee a correct execution.
A plan leading to the achievement of the desired goals and satisfying temporal constraints would be consid-

ered as *admissible*. From all *admissible* plans, we select the one with the minimal expected utility to be executed.

The paper is organized as follows: in Section 2, we describe some basic temporal concepts concerning the knowledge of an agent, tasks and the representation of a temporal precedence graph. In Section 3 we formulate our problem; then in Section 4 we describe a temporal constraint propagation algorithm. Section 5 shows how to calculate probabilities of intervals of tasks, and in Section 6 we select the *admissible* plan which has the minimal expected utility. Finally, we present some analysis and experiment results in section 8 and we conclude in Section 9.

2 FORMAL FRAMEWORK

In this section, we describe the notion of a task and a temporal precedence graph of tasks that we use throughout the rest of the paper.

2.1 Task Description

The goal of the agent is to choose a subset of executing tasks to convert some initial states into some desired goals.

Definition 1 *We call a task, the action realized by an agent throughout a duration of time. For a task t, one associates the list $<(I_t^-, I_t^+, \Delta_t, Pr_t, C_t)>$, where :*

- *I_t^- is the earliest start time, i.e., the earliest time at which the execution of the task can start;*
- *I_t^+ is the latest end time, i.e., the time at which the execution of the task must finish;*
  $[I_t^-, I_t^+]$ is the time window referring to absolute time during which task t can be executed;
- *$\Delta_t = \{d_t^1, d_t^2, ..., d_t^m\}$ is a set of possible durations of time such that $d_t^i \in \Re$ $(i = 1...m)$, is a period of time necessary to accomplish the task t;*
- *$Pr_t = \{pr_t^1, pr_t^2, ..., pr_t^m\}$ is a set of probabilities such that $pr_t^i \in \Re$ $(0 \leq pr_t^i \leq 1$ and $i = 1...m)$ is the probability to execute t during $d_t^i \in \Delta_t$;*
- *$C_t = \{c_t^1, c_t^2, ..., c_t^m\}$ is a set of costs payed to accomplish the task t such that $c_t^i \in \Re$ $(i = 1...m)$ is the cost to execute t during $d_t^i \in \Delta_t$.*

In this framework, we shall consider a task to be an atomic activity which can be undertaken and accomplished in whole. No medium states are taken into account - either the task is successfully accomplished or it fails.

2.2 Precedence Constraints

In real world, The execution of task depends on certain conditions like the time, the resources and/or the execution of other tasks. In this last case, we talk about precedence constraints in the form of partial order of a set of tasks. In this paper, We distinguish between two kinds of precedence constraints :

1. **Conjunctive Precedence Constraint**: let $\{t_1, t_2, \ldots, t_k\}$ be tasks. There is a conjunctive precedence constraint between t and the set of tasks $\{t_1, t_2, \ldots, t_k\}$, denoted $[t_1, t_2, \ldots, t_k] \rightarrow t$, if t can be executed only if all tasks $t_1, t_2, \ldots, t_k$ have already been executed.
2. **Disjunctive Precedence Constraint**: let $\{t_1, t_2, \ldots, t_k\}$ be tasks. There is a disjunctive precedence constraint between t and the set of tasks $t_1, t_2, \ldots, t_k$, noted $t_1|t_2|\ldots|t_k \rightarrow t$, if t can be executed if at least one of tasks $t_1, t_2, \ldots, t_k$ is executed.

   A special case is when t has a unique predecessor $(t' \rightarrow t)$, in this situation we talk about simple precedence constraint.

Note also that, it is often the case that after executing the preceding task one should wait for some time before the execution of the following task becomes possible. We shall call δ_{t_i,t_j} the *time delay* between tasks t_i and t_j.

2.3 Temporal Precedence Graph

The goal of the agent is to execute a subset of tasks represented by an acyclic temporal graph where nodes represent tasks (described in 2.1) and arcs represent precedence constraints between tasks (described in 2.2). More formally :

Definition 2 *Let T be a set of tasks as described in 2.1, E be a set of conjunctive and disjunctive precedence constraints such that $E = \{c | c = [t_1, t_2, \ldots, t_k] \rightarrow t$ or $c = t_1|t_2|\ldots|t_k \rightarrow t$ where $t_1, t_2, \ldots, t_k \in T\}$ and D be a set of delay constraints such that $D = \{\delta_{t_i,t_j} | \delta_{t_i,t_j} \in \Re, t_i, t_j \in T$ and $\exists$ an arc from t_i to $t_j\}$, a temporal precedence graph is an acyclic oriented graph noted $G = (T, E, D)$ where nodes are elements of T and arcs are elements of E.*

Note that, nodes of the set T of any temporal precedence graph G can be divided into three disjoint sets $T = T_I \cup T_M \cup T_F$ of tasks having no preceding tasks, intermediate tasks having preceding tasks and being predecessors for other tasks and final tasks being no predecessors.
Figure 1 represents a temporal precedence graph $G = (T, E, D)$, where $T = \{t_1, ..., t_{18}\}$, E represents the

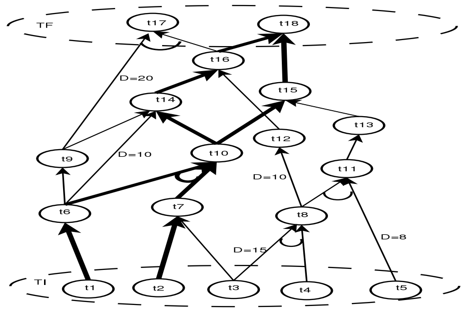

Figure 1: A temporal precedence graph of tasks

precedence constraints between tasks and D represents delay between tasks. We represent conjunctive precedence constraints by arcs.

3 STATING THE PROBLEM

Given a temporal precedence graph of tasks $G = (T, E, D)$ where T is a set of temporal tasks as described in 2.1, E represents conjunctive and disjunctive precedence constraints as described in 2.3 and D is a set of delay between tasks, the problem to be solved is to find a single plan of executable tasks in T so that some goals which are of interest are satisfied. In fact a plan is specified by a partial order of tasks. More formally :

Definition 3 *A temporal plan generation problem is a couple (G, T') where $G = (T, E, D)$ is a temporal precedence graph of tasks and $T' \subset T_F \subset T$ is a subset of final tasks in T.*

The plan should satisfy all the constraints i.e. AND/OR graph, precedence and temporal constraints. We are interested in finding a plan that has a high probability to be executed during a reduced time and with a reduced cost. To do that, we proceed in four stages:

1. Determining of **Feasible Plans**. A feasible plan is a subgraph of the initial graph G where constraints are conjunctive or simple. This subgraph represents path in the graph starting by initial tasks and ending by goal tasks. We use both depth-first and backwards search methods to find all feasible plans. T_G being the set of all goal tasks and T_I the set of all initial tasks, we start with the goal tasks to be accomplished, and we finish when all tasks in T_I are accomplished.
   For example, consider the problem given by $T_G = \{t_{18}\}$ and $T_I = \{t_2\}$ for the graph given in Figure 1. Feasible plans, marked with a bold line, are $[\{[t_2, t_7], [t_1, t_6]\}, t_{10}, t_{14}, t_{15}, t_{18}]$ and $[\{[t_2, t_7], [t_1, t_6]\}, t_{10}, t_{16}, t_{18}]$.

2. Determining of **Admissible Plans**. We call admissible plan, a feasible plan where all temporal constraints of tasks and constraints between tasks are satisfied.

3. Selection of **The Most Likely Temporal Plan**. This step selects from all admissible plans, the best likely one to be executed. This last must have a high probability of execution and reduced cost and time.

4. Selection of **The Most Likely Temporal Scheduling**. This step selects from all scheduling of the selected admissible plan, the one with the minimal total expected utility.

We denote by $\mathcal{P}_F$ the set of all feasible plans in temporal precedence graph G and by $T(\mathcal{P})$ the subset of T of all the tasks occurring in the feasible plan $\mathcal{P}$.

In the next section we present the propagation algorithm of intervals execution allowing to determine from the feasible plan set, the set of admissible plans.

4 CONSTRAINT PROPAGATION ALGORITHM

Given a temporal plan generation problem (G, T') where G is a temporal precedence graph and T' is a subset of final tasks, we determine for each task in each feasible plan the set of temporal intervals during which a task can be executed by propagating temporal constraints through the graph. Recall that a feasible plan is a subgraph where constraints are conjunctive or simple.
This propagation organizes the graph into levels so that: l_0 is the level containing initial tasks (T_I), l_1 contains all nodes that are constrained only by initial tasks, l_i contains all nodes whose predecessors include nodes at level l_{i-1}. For each node in any given level l_i, we compute all its possible execution intervals from its predecessors.
Given a task t, we call s_t (respectively e_t) a possible start time of t (respectively a possible end time of t), S_t (respectively E_t) the set of possible start times of t (respectively the set of possible end times of t) and $\mathcal{I}_t$ the set of all possible execution intervals of the t. A possible execution interval is formed by a possible start time and a possible end time. $\mathcal{I}_t = \{I_t^1, I_t^2, ..., I_t^m\}$ where $I_t^i = [s_t^i, e_t^i]$ is the possible execution interval to task t taking its duration $d_t^i \in \Delta_t$.
In the next section, we describe the algorithm used to calculate the execution intervals of each task in each feasible plan $\mathcal{P}$.

4.1 Execution Intervals of Tasks

Given a task t belongs a feasible plan $\mathcal{P}$ ($t \in T(\mathcal{P})$) where $\Delta_t = \{d_t^1, d_t^2, ..., d_t^m\}$ is its set of durations, I_t^- is its earliest start time and I_t^+ is its latest end time, the possible times for starting and ending execution of t are calculated as follows :

- *level* l_0 : $t_i \in T_I$ (t_i is an initial task) : Suppose that the agent can begin its execution at a given time noted $start\_time$.
  - $S_t = \{s_t | s_t = max(I_t^-, start\_time)\}$
  - For $i = 1$ to $i = m$ where m is the number of possible execution durations of task t
    $E_t = \{e_t^i | e_t^i = s_t + d_t^i\}$
  - $\mathcal{I}_t = \{I_t^i = [s_t, e_t^i]$ where $i = 1..m\}$. I_t^i is a possible execution interval to task t.
- *level* l_i : for each task in level l_i, its possible start times are computed as all the times at which the predecessor tasks can finish. Thus we define the set of possible start and end times of each task at level l_i as follows :
  1. If $t \in T_M \cup T_F$ where $\Delta_t = \{d_t^1, d_t^2, ..., d_t^m\}$ is the set of execution durations of t and t has a unique task predecessor t' ($t' \rightarrow t$) where $E_{t'} = \{e_{t'}^1, ..., e_{t'}^p\}$ then :
     (a) Determine the set S_t (initialized to $\emptyset$) like that :
       - For $i = 1$ to $i = p$ (p is the number of possible execution end times of task t')
         $S_t = S_t \cup \{s_t^i = max(I_t^-, e_{t'}^i + \delta_{t',t})\}$

     (b) Determine the set E_t (initialized to $\emptyset$) like that :
       - For $i = 1$ to $i = p$
         - For $j = 1$ to $j = m$
           $E_t = E_t \cup \{e_t^{ij} = s_t^i + d_t^j\}$

       We call $I_t^{ij} = [s_t^j, e_t^{ij}]$ a possible execution interval of t where $s_t^j \in S_t$ and $e_t^{ij} \in E_t$.
  2. If $t \in T_M \cup T_F$ where $\Delta_t = \{d_t^1, d_t^2, ..., d_t^m\}$ is the set of execution durations of t which has a set of direct task predecessors $\{t_1, t_2, \dots, t_n\}$ ($[t_1, t_2, \dots, t_n] \rightarrow t$) and if for each task t_i ($i = 1..n$), $E_{t_i} = \{e_{t_i}^1, e_{t_i}^2, ..., e_{t_i}^{j_i}\}$ then :
     (a) Determine the set S_t (initialized to $\emptyset$) of possible execution start times of t like that :
       - For each task t_i
         - For $k = 1$ to $k = j_i$
           $S_t = S_t \cup \{s_t^k = max(I_t^-, max(e_{t_i}^k + \delta_{t_i,t}))\}$

     (b) Determine the set E_t (initialized $\emptyset$) of possible execution end time of task t like that :
       - For $k = 1$ to $k = p$ (p represents the cardinality of the set S_t calculated in 2a)
         - For $r = 1$ to $r = m$ (m is the number of possible execution durations of task t)
           $E_t = E_t \cup \{e_t^{kr} = s_t^k + d_t^r\}$

       We call $I_t^{kr} = [s_t^k, e_t^{kr}]$ a possible execution interval of t where $s_t^k \in S_t$ and $e_t^{kr} \in E_t$.

4.2 Admissibility of Plans

A *feasible* plan $\mathcal{P} \in \mathcal{P}_F$, in order to become an *admissible* one, must satisfy all the temporal constraints. More formally, for any task $t \in T(\mathcal{P})$, for each $e_t^i \in E_t$, the following condition : $e_t^i \leq I_t^+$ must hold. If a possible end time of E_t exceeds I_t^+, we consider that execution interval $I_t^i = [s_t^i, e_t^i]$ is not valid. In the other hand, if all possible end times of E_t exceed I_t^+, we consider that task t is not able to execute. Thus we consider the plan which contains such task is not admissible.
We denote by $\mathcal{P}_A$ the set of all admissible plans $\mathcal{P}$. We have $\mathcal{P}_A = \bigcup \mathcal{P}$, where $\mathcal{P}$ is an admissible plan.

5 PROBABILISTIC TEMPORAL PROPAGATION

We describe in this section how we can weight each of those intervals calculated as above, with a probability. This probabilistic weight allows us to know the probability for a task to be executed during a given interval of time. For that, a probability propagation algorithm among the graph of tasks is described using for each node its execution time probability and the end-time probabilities of its predecessors.
The probability of a possible execution interval I_t^i depends on its start time (the end time of the previous tasks) and the probability of execution time pr_t^i. To simplify, we consider in the rest of this paper, that no time delay between tasks.

In the next section, we compute the probability that the execution for a task t to occur during an interval I_t^i where s_t^i is a possible start time and e_t^i is a possible end time of task t.

5.1 Probability Propagation Algorithm

Before a task t can start its execution, all its direct predecessors must be finished. The probability for the execution of t start at s_t^i is defined by :

1. If t has an only direct predecessor t' ($t' \rightarrow t$) in the admissible plan $\mathcal{P}$ where $E_{t'} = \{e_{t'}^1, ..., e_{t'}^p\}$ and $S_t = \{s_t^1, s_t^2, ..., s_t^m\}$ is the set of possible execution start time of task t then :
   - The probability that task t starts its execution at s_t^i when its predecessor t' finishes its execution at $e_{t'}^j$, noted $pr_{start}(s_t^i | e_{t'}^j)$, is calculated as following :

- For $i = 1$ to $i = m$ do
  - For $j = 1$ to $j = p$ do
    - If $s_t^i < e_{t'}^j$ then $pr_{start}(s_t^i | e_{t'}^j) = 0$
    - If $s_t^i \geq e_{t'}^j$ then $pr_{start}(s_t^i | e_{t'}^j) = 1$

2. If t has a set of direct predecessors $\{t_1, t_2, \ldots, t_n\}$ in the admissible plan $\mathcal{P}$ i.e. $[t_1, t_2, \ldots, t_n] \to t$ then let $t_i \in \{t_1, t_2, \ldots, t_n\}$ where $E_{t_i} = \{e_{t_i}^1, e_{t_i}^2, \ldots, e_{t_i}^{j_i}\}$ such that j_i represents the number of possible end times of task t_i and let $S_t = \{s_t^1, s_t^2, ..., s_t^m\}$ be the set of possible start times of t then :
   - The probability that task t starts its execution at s_t^r ($r = 1...m$) when each direct predecessor $t_i \in \{t_1, t_2, \ldots, t_n\}$ of t finishes its execution at $e_{t_i}^k$ ($k = 1..j_i$) is calculated as following :
   - If $s_t^r < max\{e_{t_i}^k\}$ then $pr_{start}(s_t^r | e_{t_i}^k) = 0$
   - If $s_t^r \geq max\{e_{t_i}^k\}$ then $pr_{start}(s_t^r | e_{t_i}^k) = 1$

A special case has to be considered for the first tasks. Indeed, these initial tasks ($\in T_I$) have no predecessors, the probability of starting the execution of an initial task t at s^t is given by : $pr_{start}(s_t) = 1$.

5.2 Execution Intervals Probability

In this section, we describe the method used to calculate the execution intervals probability which depends on execution durations and execution start times.
Let $pr_t(d_t^i | s_t^i)$ be the probability that the execution of task t takes d_t^i units of time when it starts at s_t^i.
The probability of an execution interval I_t^i of task t that has an only direct predecessor t' is the probability $Pr_w(I_t^i | e_{t'}^j)$ that interval $I_t^i = [s_t^i, e_t^i]$ is the interval during which a task t is executed, if the predecessor task ends at $e_{t'}^j$. This probability measures the probability that a task t starts its execution at s_t^i and ends at e_t^i. It is defined as following :

$$Pr_w(I_t^i | e_{t'}^j) = pr_{start}(s_t^i | e_{t'}^j) * pr_t(d_t^i | s_t^i)$$

where $e_{t'}^j$ is the end time of the last executed task.
Indeed, a task t can start its execution only when its predecessor t' finishes. That is why the probability of execution interval depends on the predecessor end time.

In the case where t, such that $\Delta_t = \{d_t^1, d_t^2, ..., d_t^m\}$, has a set of direct predecessors $\{t_1, t_2, \ldots, t_n\}$, where $t_i \in \{t_1, t_2, \ldots, t_n\}$, ends its execution at $e_{t_i}^{k_{t_i}}$ ($k_{t_i} = 1..j_i$ s.t. j_i represents the number of possible execution end times of task t_i), we have :

$$Pr_w(I_t^r | e_{t_i}^{k_{t_i}}) = pr_{start}(s_t^r | e_{t_i}^{k_{t_i}}) * pr_t(d_t^r | s_t^r)$$

where $r = 1..m$.

A special case has to be considered for the first task. The probability that an initial task t executes in the interval I_t^r is equal to the probability that it starts its execution at s_t and takes d_t^r units of time. More formally :

$$Pr_w(I_t^r) = pr_{start}(s_t) * pr_t(d_t^r | s_t)$$

For example, suppose 0.7 is the probability for the execution of task t that has an only direct predecessor t', to take 2 units of time when it starts at 5 ($Pr_t(d_t^i = 2 | s_t^i = 5) = 0.7$). We assume that the end execution of the predecessor task t' is 3. The probability of the interval $[5, 7]$ is :
$Pr_w(I_t^i = [5, 7] | e_{t'}^j = 3) = 1 \times 0.7 = 0.7$
If for example task t' ends its execution at 7, we obtain :
$Pr_w(I_t^i = [5, 7] | e_{t'}^j = 7) = 0 \times 0.7 = 0$

6 SELECTION OF THE MOST LIKELY ADMISSIBLE PLAN

As the order of searching for plans is arbitrary, the search may produce several admissible plans. In this case, further criteria will be applied (Probability, cost and time) to analyze and compare them so as to select one to be executed. In this section we describe the method we use to choose the most likely admissible plan to be executed.

We are interested in finding a unique execution plan that has a high probability and reduced cost and time. To do that, we calculate expected utilities of cost and time of all tasks. Then, we calculate the expected utility of tasks of each admissible plan, finally we select the one with the smallest expected utility to be the most likely plan to be executed. We detail this assumption in the next section.

6.1 Expected Utilities of Cost and Time of Tasks

For each task in an admissible plan $\mathcal{P}$, we calculate its expected utility of cost and time. The expected utility of cost is calculated in function of costs associated of execution durations and of execution probabilities associated of execution intervals (calculated in 5.2).

The expected utility of time is calculated in function of possible execution durations and of execution probabilities associated with execution intervals (calculated in 5.2). More formally :

Let t, where $\Delta_t = \{d_t^1, d_t^2, ..., d_t^m\}$ is the set of possible durations of t, $C_t = \{c_t^1, c_t^2, ..., c_t^m\}$ is the set of execution costs of t where c_t^k is the cost to accomplish t taking duration d_t^k ($k = 1..m$).

1. If t has an only direct predecessor t' such that $E_{t'} = \{e^1_{t'}, e^2_{t'}, \ldots, e^r_{t'}\}$ is its set of possible execution end times, and if the probability of t to be executed in duration d^k_t where its predecessor t' has finished its execution at $e^j_{t'}$ is noted $Pr_w(I^k_t | e^j_{t'})$ (section 5.2), then :
   - Expected Utility of cost of t is calculated as following :

     $$\mu(cost(t|t')) = \sum_{j=1}^{r} \sum_{k=1}^{m} Pr_w(I^k_t | e^j_{t'}) * c^k_t$$

     To simplify, we denote $\mu(cost(t|t'))$ by $\mu(cost(t))$.
   - Expected Utility of time of t is calculated as following :

     $$\mu(time(t|t')) = \sum_{j=1}^{r} \sum_{k=1}^{m} Pr_w(I^k_t | e^j_{t'}) * d^k_t$$

     To simplify, we denote $\mu(time(t|t'))$ by $\mu(time(t))$.
2. If t has a set of direct predecessor $t_1, t_2, \ldots, t_n$ then, let $E_{t_i} = \{e^1_{t_i}, e^2_{t_i}, \ldots, e^{r_{t_i}}_{t_i}\}$ be the set of possible execution end times of $t_i \in \{t_1, t_2, \ldots, t_n\}$ then :
   - Expected Utility of cost of t is calculated as following :

     $$\mu(cost(t|t_1, \ldots, t_n)) = \sum_{i=1}^{n} \mu(cost(t|t_i))$$

     To simplify, we denote $\mu(cost(t|t|t_1, \ldots, t_n))$ by $\mu(cost(t))$.
   - Expected Utility of time of t is calculated as following :

     $$\mu(time(t|t_1, \ldots, t_n)) = \sum_{i=1}^{n} \mu(time(t|t_i))$$

     To simplify, we denote $\mu(time(t|t_1, \ldots, t_n))$ by $\mu(time(t))$.

In the next section, we calculate the total expected utility of each admissible plan $\mathcal{P}$.

6.2 Expected Utility of Plans

The expected utility of a plan $\mathcal{P}$ is calculated in function of expected utilities of cost and time (calculated above).
The total expected utility of cost of a plan $\mathcal{P}$ is calculated by adding all expected utilities of cost of all tasks in this plan. More formally :

$$\mu(cost(\mathcal{P})) = \sum_{i=1}^{n} \mu(cost(t_i))$$

where n is the number of tasks in $\mathcal{P}$.
The total expected utility of time of a plan $\mathcal{P}$ is calculated as following:

- $\mu(time(\mathcal{P}))$ is initialized to $max(\mu(time(t_i)))$ where $t_i \in T_F$
  – If t has an only direct predecessor t' with expected utility $\mu(time(t'))$, then we add $\mu(time(t'))$ to the total expected utility of time of the plan. More formally :

  $$\mu(time(\mathcal{P})) = \mu(time(\mathcal{P})) + \mu(time(t'))$$

  – If t has a set of direct predecessors $\{t_1, t_2, \ldots, t_n\}$ then, we add the maximum of expected utilities of tasks $t_1, t_2, \ldots, t_n$ to the total expected utility of time of the plan. More formally :

  $$\mu(time(\mathcal{P})) = \mu(time(\mathcal{P})) + max_{i=1}^{n} \mu(time(t_i))$$

To calculate the total expected utility for each plan $\mathcal{P} \in \mathcal{P}_A$, we assign a value to each plan according to the preferences of the user. We translate these preferences into a function $\mu(\mathcal{P})$: μ called function of utility. This function is prone to the cost and the time that are balanced by coefficients α and β. This allows us to adjust the relative importance of the various utilities according to the preferences of the user. This utility function is described by :

$$\mu(\mathcal{P}) = \alpha\mu(cost(\mathcal{P})) + \beta\mu(time(\mathcal{P}))$$

where $\alpha + \beta = 1$ and $P \in \mathcal{P}_A$.

Among all admissible plans, we choose the one with the smallest utility to be executed :

$$\mathcal{P}^* = argmin_{\mathcal{P} \in \mathcal{P}_A} \mu(\mathcal{P})$$

This plan is the most likely one to be executed satisfying all constraints and that has a high probability to be executed with reduced cost an time of execution. In the next section, we determinate the most likely scheduling to be executed.

7 SELECTION OF THE MOST LIKELY SCHEDULING

We call scheduling of a plan $\mathcal{P}$, the set of tasks of $\mathcal{P}$ where each task is executed in a well defined interval. If there is a simple precedence constraint between two tasks t_i and t_j ($t_i \rightarrow t_j$), then the execution interval of task t_i must occur before the one of task t_j. More formally, let $T(\mathcal{P}) = \{t_1, t_2, \ldots, t_n\}$ be the set of tasks of an admissible plan $\mathcal{P}$, and $\mathcal{I}_{t_i} = \{I^1_{t_i}, I^2_{t_i}, \ldots, I^{j_{t_i}}_{t_i}\}$ be the set of execution intervals of task t_i in $T(\mathcal{P})$, we have :

$$\mathcal{P}_{ord} = \{(t_1, I^{k_{t_1}}_{t_1}), (t_2, I^{k_{t_2}}_{t_2}), ..., (t_n, I^{k_{t_n}}_{t_n})\}$$

is a possible scheduling in $\mathcal{P}$ where $k_{t_i} = 1..j_{t_i}$ such that if $t_i \rightarrow t_j$ (t_i and $t_j \in \mathcal{P}_{ord}$) then $e_{t_i}^{k_{t_i}} \leq s_{t_j}^{k_{t_j}}$.

We have also :

$$\mathcal{P} = \bigcup \mathcal{P}_{ord}$$

In the worst case, the total number of scheduling of a plan $\mathcal{P}$ is equal to the product of cardinalities of all possible execution interval sets of all tasks in $T(\mathcal{P})$. More formally this number is equal to :

$$No(\mathcal{P}_{ord}) = j_{t_1} * j_{t_2} * \ldots * j_{t_n}$$

In the next section, we calculate the expected utility of cost and the expected utility of time for each task $(t, I_t^{k_t}) \in \mathcal{P}_{ord}$.

7.1 Expected Utilities of Cost and Time of Tasks of Scheduling

for each $\mathcal{P}_{ord} = \{(t_1, I_{t_1}^{k_{t_1}}), (t_2, I_{t_2}^{k_{t_2}}), ..., (t_n, I_{t_n}^{k_{t_n}})\}$, we calculate the expected utility of cost and the expected utility of time for each task t_i to be executed in $I_{t_i}^{k_{t_i}}$ More formally :

1. If $(t, I_t^{k_t}) \in \mathcal{P}_{ord}$ has an only direct predecessor $(t', I_{t'}^{k_{t'}}) \in \mathcal{P}_{ord}$ where $(t, I_t^{k_t}) \rightarrow (t, I_t^{k_t})$ then :
   - The expected utility of cost of $(t, I_t^{k_t})$ is calculated as following :
   $$\mu(cost((t, I_t^{k_t})|(t', I_{t'}^{k_{t'}}))) = Pr_w(I_t^{k_t}|e_{t'}^{k_{t'}}) * c_t^{k_t}$$
   - The expected utility of time is calculated as following :
   $$\mu(time((t, I_t^{k_t})|(t', I_{t'}^{k_{t'}}))) = Pr_w(I_t^{k_t}|e_{t'}^{k_{t'}}) * d_t^{k_t}$$
2. If $(t, I_t^{k_t}) \in \mathcal{P}_{ord}$ has a set of direct predecessors $\{(t_1, I_{t_1}^{k_{t_1}}), (t_2, I_{t_2}^{k_{t_2}}), ..., (t_p, I_{t_p}^{k_{t_p}})\}$ then, let $(t_i, I_{t_i}^{k_{t_i}}) \in \{(t_1, I_{t_1}^{k_{t_1}}), (t_2, I_{t_2}^{k_{t_2}}), ..., (t_p, I_{t_p}^{k_{t_p}})\}$ then :
   - The expected utility of cost of $(t, I_t^{k_t})$ is calculated as following :
   $$\mu(cost((t, I_t^{k_t})|(t_1, I_{t_1}^{k_{t_1}}), (t_2, I_{t_2}^{k_{t_2}}), ..., (t_p, I_{t_p}^{k_{t_p}})))$$
   $$= \sum_{i=1}^{p} \mu(cost((t, I_t^{k_t})|(t_i, I_{t_i}^{k_{t_i}})))$$
   - The expected utility of time of $(t, I_t^{k_t})$ is calculated as following :
   $$\mu(time((t, I_t^{k_t})|(t_1, I_{t_1}^{k_{t_1}}), (t_2, I_{t_2}^{k_{t_2}}), ..., (t_p, I_{t_p}^{k_{t_p}})))$$
   $$= \sum_{i=1}^{p} \mu(time((t, I_t^{k_t})|(t_i, I_{t_i}^{k_{t_i}})))$$

In the next section, we calculate the total expected utility of a scheduling $\mathcal{P}_{ord}$.

7.2 Expected Utility of Scheduling

To calculate the total expected utility of a scheduling $\mathcal{P}_{ord}$, we calculate first the total expected utilities of cost and time of $\mathcal{P}_{ord}$ as following :

1. The total expected utility of cost of $\mathcal{P}_{ord}$ is calculated by adding all expected utilities of all tasks in $\mathcal{P}_{ord}$. More formally :
   $$\mu(cost(\mathcal{P}_{ord})) = \sum_{i=1}^{n} \mu(cost(t_i, I_{t_i}^{k_{t_i}}))$$
   where n is the number of tasks of $\mathcal{P}$.
2. The total expected utility of time of $\mathcal{P}_{ord}$ is calculated as following :
   - $\mu(time(\mathcal{P}_{ord}))$ is initialized to $max(\mu(time(t_i, I_{t_i}^{k_{t_i}})))$ where $t_i \in T_F$
   - If $(t, I_t^{k_t})$ has an only direct predecessor $(t', I_{t'}^{k_{t'}})$ such that its expected utility is $\mu(time(t', I_{t'}^{k_{t'}}))$ then we add this expected utility to the total expected utility of time of plan. More formally :
   $$\mu(time(\mathcal{P}_{ord})) =$$
   $$\mu(time(\mathcal{P}_{ord})) + \mu(time(t', I_{t'}^{k_{t'}}))$$
   - If $(t, I_t^{k_t})$ has a set of direct predecessors $\{(t_1, I_{t_1}^{k_{t_1}}), (t_2, I_{t_2}^{k_{t_2}}), ..., (t_p, I_{t_p}^{k_{t_p}})\}$ then, we add the maximum of expected utilities of these tasks to the total expected utility of time of plan. More formally :
   $$\mu(time(\mathcal{P}_{ord})) =$$
   $$\mu(time(\mathcal{P}_{ord})) + max_{i=1}^{p} \mu(time(t_i, I_{t_i}^{k_{t_i}}))$$

For each scheduling $\mathcal{P}_{ord} = \{(t_1, I_{t_1}^{k_{t_1}}), (t_2, I_{t_2}^{k_{t_2}}), ..., (t_n, I_{t_n}^{k_{t_n}})\}$ of the admissible plan $\mathcal{P}$ chosen to be executed, we calculate its utility function, where $\alpha + \beta = 1$ and, defined by :

$$\mu(\mathcal{P}_{ord}) = \alpha * \mu(cost(\mathcal{P}_{ord})) + \beta * \mu(time(\mathcal{P}_{ord}))$$

From all scheduling formed by $\mathcal{P}$ we choose the one with the smallest expected utility to be executed :

$$\mathcal{P}_{exec}^{*} = argmin_{\mathcal{P}_{ord} \in P} \mu(\mathcal{P}_{ord})$$

This plan determines the expected intervals of execution in which tasks should be executed by the agent.

8 ANALYSIS AND EXPERIMENTS

The number of execution intervals for each task t depends on the number of its direct precedence tasks, its

time window $[I_t^-, I_t^+]$ and the cardinality of its duration set $|\Delta_t|$. When one of these constraints increases, this number increases. The number of plans in a graph depends on the number of execution intervals of tasks, the number of tasks and the number of precedence constraints. This number increases when the number of execution intervals or the number of tasks increases and it decreases when the number of precedence constraints increases.
In the worst case, the number of execution intervals for an initial task is equal to the cardinality of its duration set. The number of execution intervals for an intermediate or final task which has an only direct predecessor is equal to the cardinality of its duration set multiplied by the cardinality of the set of the ends of the direct predecessor task. The number of execution intervals for an intermediate or final task which has a set of direct predecessors is equal to the cardinality of its duration set multiplied by all the cardinalities of the sets of the ends of the direct predecessor tasks. More formally, in the worst case :

- If $t \in T_I$ then $|\mathcal{I}_t| = |\Delta_t|$
- If $t \in T_M \cup T_F$ where t has an only direct predecessor t' then $|\mathcal{I}_t| = |\Delta_t| * |E_{t'}|$
- If $t \in T_M \cup T_F$ and t has a set of direct predecessors $\{t_1, t_2, \ldots, t_n\}$ then $|\mathcal{I}_t| = |\Delta_t| * \prod_{i=1}^{n} |E_{t_i}|$

In our experimental tests, we used these parameters : the total number of tasks, the number of plans we can obtain if all temporal constraints are correct (worst case) and the number of plans obtained without the ones violating temporal constraints.

We remarked that theoretically, for 50 tasks with 5 execution intervals for each, we obtain 250000 plans but with our approach, we generate only 20569 plans. These first experimental results consolidate our idea of the founded good of the approach. However, additional experimental tests on the other factors, as the size of the temporal windows and the number of precedence constraints, are to be analyzed.

9 CONCLUSION

In this paper we present an approach of temporal probabilistic task planning where we construct a plan of tasks that satisfies all constraints and executes with a high probability during a reduced time and with a reduced cost. Our approach allows the agent to determine the set of tasks to execute and when to execute them by respecting all temporal and precedence constraints. This approach is one of the first techniques combining probability and time in planning.

Future work will be focused on other kind of experimental factors and on comparing our approach with other ones. Another issue consists in finding other heuristics to choose the most likely plan to be executed and reducing the search space then comparing them with the heuristics presented in this paper.

REFERENCES

Baki, B., Beynier, A., Bouzid, M., and Mouaddib, A. (2004). Temporal Probabilistic Task Planning in an Emergency Service. In *Complex Systems Intelligence and Modern Technological Applications, CSIMTA-04*, pages 427–434.

Bresina, J. and Washington, R. (2000). Expected Utility Distributions for Flexible Contingent Execution. In *Proceedings of the AAAI-2000 Workshop: Representation Issues for Real-World Planning Systems*.

Bresina, J. L., Dearden, R., Meuleau, N., Ramakrishnan, S., Smith, D. E., and Washington, R. (2002). Planning Under Continuous Time and Resource Uncertainty: A Challenge for AI. In *AIPS Workshop on Planning for Temporal Domains*, pages 91–97.

Dean, T., Kaelbling, L., Kirman, J., and Nicholson, A. (1993). Planning With Deadlines in Stochastic Domains. In *Proceedings of the 11th National Conference on Artificial Intelligence*, pages 574–579. AAAI.

Draper, D., Hanks, S., and Weld, D. (1994). Probabilistic Planning with Information Gathering and Contingent Execution. In *Proceedings of the Second International Conference on AI Planning Systems, AIPS-94*, pages 31–63.

Ghallab, M. and Laruelle, H. (1994). Representation and Control in IxTeT, a Temporal Planner. In *Proceedings of the Second International Conference on Artificial Intelligence Planning Systems, AIPS-94*, pages 61–67.

Kushmerick, N., Hanks, S., and Weld, D. (1994). An Algorithm for Probabilistic Least-Commitment Planning. In *Proceedings of the Twelfth National Conference on Artificial Intelligence,AAAI-94*, pages 1073–1078.

Laborie, P. and Ghallab, M. (1995). Planning with Sharable Resource Constraints. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, IJCAI-95*, pages 1643–1651.

Lemai, S. and Ingrand, F. (2004). Interleaving Temporal Planning and Execution in Robotics Domains. In *AAAI*, pages 617–622.

Pemberton, J. C. and Greenwald, L. G. (2002). On the Need for Dynamic Scheduling of Imagings Satellites. In *Proceedings of the American Society for Photogrammetry and Remote Sensing, ASPRS-02*.

Zilberstein, S. (1995). The Utility of Planning. *SIGART Bulletin*, 6(1):42–47.