Interleaving Temporal Planning and Execution in Robotics Domains

Solange Lemai and **Félix Ingrand***
LAAS/CNRS,
7 Avenue du Colonel Roche, F-31077 Toulouse Cedex 04, France
{*slemai,felix*}*@laas.fr*

Abstract

Many autonomous systems such as mobile robots, UAVs or spacecraft, have limited resource capacities and move in dynamic environments. Performing on-board mission planning and execution in such a context requires deliberative capabilities to generate plans achieving mission goals while respecting deadlines and resource constraints, as well as run-time plan adaption mechanisms during execution. In this paper we propose a framework to integrate deliberative planning, plan repair and execution control in a dynamic environment with stringent temporal constraints. It is based on lifted partial order temporal planning techniques which produce flexible plans and allow, under certain conditions discussed in the paper, plan repair interleaved with plan execution. This framework has been implemented using the IxTeT planner and used to control a robotic platform.

Introduction

The development of a complex autonomous system relies on the arrangement of the software components in a closed loop architecture, usually organized around two levels. A *decisional level* generates a plan of actions which achieves the mission goals, and executes this plan in a robust way. A *functional level*, interfaced with the hardware, executes the refined actions of the plan. The decisional level usually embeds a time-consuming planning process, as well as a time-bounded reactive execution process. We are interested in the key problem of defining how these two processes should interact to balance deliberation and reaction.

Various strategies have been applied so far. Some approaches propose the use of a strong executive, enhanced with deliberative capabilities. Procedural executives, such as TDL (Simmons & Apfelbaum 1998) or PRS (Ingrand *et al.* 1996), support action decomposition, synchronization, execution monitoring and exception handling. In PROPEL (Levinson 1995), and PropicePlan (Despouys & Ingrand 1999), the planner is used by the executive to anticipate by simulating subplans, or to generate a new subplan corresponding to the current situation. These systems provide mostly a reactive behavior, whereas a look-ahead far in the future is necessary to achieve a strong level of autonomy (e.g. to manage the level of a limited resource during the entire mission).

In the "batch planning" approach of the Remote Agent (Muscettola *et al.* 1998), planning and plan execution are two separate processes. The agent can perform replanning (but with long standby periods) and back to back planning.

Other approaches propose to interleave planning and execution in a continuous way. The planning process remains active to adapt the plan when new goals are added or to resolve conflicts appearing after a state update; and actions which are ready to be executed are committed to execution even if the plan is not yet completely generated. Examples include IPEM (Ambros-Ingerson & Steel 1988), based on the classical Partial Order Planning framework; ROGUE (Haigh & Veloso 1998), which has been deployed on mobile robots; or the multi-agent architecture CPEF (Myers 1999).

Still, very few approaches really take into account timing problems, such as actions with duration and goals with deadlines, and their effects on the planning and execution processes. Examples of temporal systems include CASPER and IDEA. In the CASPER system (Chien *et al.* 2000), state and temporal data are regularly updated and potential future conflicts are resolved using iterative repair techniques. However this approach does not handle conflicts which appear within the replanning time interval. The IDEA approach (Finzi, Ingrand, & Muscettola 2004) uses temporal planning techniques at any level of abstraction (mission planning as well as reactive execution). This system offers look-ahead abilities with flexible planning horizons and the use of a common language at any level of abstraction, but does not yet handle non-unary resources.

In this paper we propose a framework to integrate deliberative planning, plan repair and execution control that takes into account resource level updates and temporal constraints. These processes are embedded thanks to two interacting components which explicitly represent and reason about time: a temporal planner and a temporal executive. First the planner elaborates a complete plan. This plan is then run by the temporal executive following a cycle: integrate external messages, repair the plan if needed, decide to execute actions. This cycle enables interaction with the controlled system by taking into account runtime failures and timeouts, and updating the plan accordingly. Interleaving plan repair and execution is motivated by the facts that some parts of the plan may remain valid and executable, and that the plan can be temporally flexible and thus allow postponing and inserting actions. Plan repair uses non linear planning techniques under certain assumptions discussed in this paper. This framework assumes that the planning system has the following properties: a temporal representation based on state variables and a Partial Order Causal Link (POCL) planning process that generates flexible plans based on CSP managers, particularly the time-map relies on a Simple Temporal Network (STN) (Dechter,

*Part of this work was funded by a contract with CNES and ASTRIUM.

Meiri, & Pearl 1989). The proposed framework has been implemented in IxTeT-eXeC, using the planning system IxTeT.

The paper is organized as follows. The first section presents the general behavior of the IxTeT-eXeC system, especially the interactions between the executive and the controlled system, and briefly introduces the underlying planning system. The second section details the dynamic replanning framework and addresses the issues raised by the interleaving of planning and execution processes on the same plan. The last section illustrates the performances of IxTeT-eXeC on a rover exploration scenario involving goal deadlines and resource contention.

Overview of the IxTeT-eXeC system

General behavior

The key component in IxTeT-eXeC is the temporal executive (TeXeC) which interacts with the planner and the controlled system. It controls the temporal network to decide the execution of actions and maps the timepoints to their real execution time. The execution of an action a with grounded parameters p_a, starting timepoint st^a, ending timepoint et^a, and identifier i_a is started by sending the command (*START* a p_a i_a) to the controlled system. If the action is *non preemptive*, et^a is not controllable, and TeXeC just monitors if a is completed in due time. Otherwise et^a is controllable. If the action did not terminate by itself, it is stopped with the command (*END* i_a) sent as soon as possible if a is *early preemptive*, as late as possible if it is *late preemptive*.

TeXeC integrates in the plan the reports sent by the controlled system upon each action completion. A report returns the ending status of the action (*nominal*, *interrupted* or *failed*) and a partial description of the system state. If nominal, it contains just the final levels of the resources, if any, used by the action. Otherwise, it also contains the final values of the other state variables relevant to the action.

Besides completion reports, TeXeC reacts to two types of event: user requests to insert a new goal and sudden alterations of a resource capacity. Other situations that forbid further execution of the plan without plan adaptation are:
– *temporal failures* The STN constrains each timepoint t to occur inside a time interval $[t_{lb}, t_{ub}]$. Thus two types of failure lead to an inconsistent plan: the corresponding event (typically, the end of an action) happens *too early* or *too late* (time-out).
– *action failure* The system returns a non nominal report.
– *resource level adjustment* If an action has consumed/produced more/less than expected, the plan may contain future resource contention.

TeXeC starts and controls the processes of plan adaptation. To take advantage of the temporal flexibility of the plan, the dynamic replanning strategy has two steps. A first attempt is to repair the plan while executing its valid part in parallel. If this fails or if a timepoint times out, the execution is aborted and IxTeT completely replans from scratch.

The IxTeT planning system

IxTeT is a lifted POCL temporal planner based on CSPs (Laborie & Ghallab 1995). Its temporal representation describes the world as a set of *attributes*: logical attributes (e.g. robot_position($?r$)), which are multi-valued functions of time, and resource attributes (e.g. battery_level()) over which borrowings, consumptions or productions can be specified. We note $LgcA$ and $RscA$, respectively the sets of logical and resource attributes. $LgcA_g$ and $RscA_g$ designate the sets of all possible instantiations of these attributes.

The evolution of a logical attribute value is represented through the proposition *hold*, which asserts the persistence of a value over a time interval, and the proposition *event*, which states an instantaneous change of values. The propositions *use*, *consume* and *produce* specify, respectively, the borrowing over an interval, the consumption or the production at a given instant of a given resource quantity.

```
task MOVE(?initL,?endL)(st,et){
 ?initL,?endL in LOCATIONS;
 event(ROBOT_POS():(?initL,IDLE_POS),st);
 hold(ROBOT_POS():IDLE_POS,(st,et));
 event(ROBOT_POS():(IDLE_POS,?endL),et);
 event(ROBOT_STATUS():(STILL,MOVING),st);
 hold(ROBOT_STATUS():MOVING,(st,et));
 event(ROBOT_STATUS():(MOVING,STILL),et);
 hold(PTU_POS():FORWARD,(st,et));
```

```
variable ?di,?du,?dist;
variable ?duration;
distance(?initL,?endL,?di);
distance_uncertainty(?du);
?dist = ?di * ?du;
 speed(?s);
?dist = ?s * ?duration;
?duration = et - st;
}latePreemptive
```

Figure 1: Example of action model.

An action (also called *task*) consists of a set of *events* describing the change of the world induced by the action, a set of *hold* propositions expressing required conditions or the protection of some fact between two events, a set of resource usages, and a set of constraints on the timepoints and variables of the action. An example is given in Figure 1.

A plan relies on two CSP managers. An STN handles the timepoints and their binary constraints (ordering, duration). The other CSP manages atemporal symbolic and numeric variables and their constraints (binding, domain restriction, sum, etc.). Mixed constraints between temporal and atemporal variables can also be expressed (e.g. $?dist = ?speed * (et - st)$) (see (Trinquart & Ghallab 2001)). These CSP managers compute for each variable a minimal domain which reflects only the necessary constraints in the plan.

The search explores a tree $\mathcal{T}$ in the partial plan space. In a POCL framework, a partial plan is generally defined as a 4-tuple (A, C, L, F), where A is a set of partially instantiated actions, C is a set of constraints on the temporal and atemporal variables of actions in A, L is a set of causal links[1] and F is a set of flaws. A partial plan stands for a family of plans. A partial plan is considered to be a valid solution if all these plans are coherent, that is F is empty.

The root node of $\mathcal{T}$ consists of: the initial state (initial values of all instantiated attributes), expected availability profiles of resources, goals to be achieved (desired values for specific instantiated attributes) and a set of constraints between these elements. The branches of $\mathcal{T}$ correspond to resolvers (new actions or constraints) inserted into the partial plan in order to solve one of its flaws. Three kinds of flaws are considered:
– *open conditions* are events or assertions that have not yet been established. Resolvers consist in finding an establishing event (in the plan or a new action) and adding a causal link that protects the attribute value between the establishing event and the open condition.
– *threats* correspond to pairs of *event* and *hold* which values are potentially in conflict. Such conflicts are solved by adding temporal or binding constraints.
– *resource conflicts* are detected as over-consuming sets of potentially overlapping propositions. Resolvers include insertion

[1] A causal link $a_i \xrightarrow{p} a_j$ denotes a commitment by the planner that a proposition p of action a_j is established by an effect of action a_i. The precedence constraint $a_i \prec a_j$ and binding constraints for variables of a_i and a_j appearing in p are in C.

of resource production action, etc.
A planning step consists in detecting flaws in the current partial plan, selecting one, choosing a resolver in its associated list of potential resolvers and inserting it into the partial plan. This planning step is repeated until a solution plan is found. The algorithm is complete and the flaw and resolver choices are guided by diverse heuristics not discussed here.

The advantages of the CSP-based functional approach are numerous in the context of plan execution. Besides the expressiveness of the representation (handling of time and resources), the flexibility of plans (partially ordered and partially instantiated, with minimal constraints) is well-adapted to their execution in an uncertain and dynamic environment. Plans are further constrained at execution time. Finally, the planner, performing a search in the plan space, can be adapted to incremental planning and plan repair.

Temporal plan execution and replanning

Interleaving partial order planning and execution may insert flaws in the plan. We formally define under which conditions such a partial plan remains executable.

Definitions

We extend the previous definition of a partial plan to the definition of P_t: a **partial plan partially executed up to time *t***.

Definition 1 $P_t = (RA_t, FA_t, S_t, G_t, C_t, L_t, F_t)$.

RA_t is the set of currently *running actions* ($a \in RA_t$ if $st^a_{ub} < t$ and $et^a_{ub} > t$), FA_t is the set of *future actions* ($a \in FA_t$ if $st^a_{ub} \geq t$). S_t represents the *state of the world* at time t. It is composed of 2 sets: $LgcS_t$ contains the last value of each attribute $la \in LgcA_g$, [2] $RscL_t$ contains the level at time t of each resource $r \in RscA_g$. G_t is the set of *goals* not yet completely achieved at time t (and eventually not established) [3]. C_t is the set of constraints on the variables appearing in FA_t, RA_t, S_t and G_t. L_t is the set of causal links supporting future actions. F_t is the set of flaws present in the partial plan at time t.

The level of a resource at a certain time in the future cannot be computed, since it depends on the partial order of actions using this resource. But at time t the past part of the plan is completely instantiated and linearized. Two cases have to be considered: if no running action modifies r, the exact level can be computed (case (1)); if at least one action in RA_t requires the resource, only an estimate is available (case (2)).

In case (1), the exact level is computed in IxTeT according to formula (**a**). We note $C(r)$ the resource capacity. Let p be a production, belonging to the action a^p, of a quantity $?q_p$ of the resource r at time t^p and $P(r)$ be the set of productions of resource r in the plan. Similarly, $C(r)$ is the set of consumptions c, belonging to the action a^c, of a quantity $?q_c$ at time t^c, and $B(r)$ is the set of borrowings b, belonging to the action a^b, of a quantity $?q_b$ between st^b and et^b. Then, if no running action modifies r, $RscL_t(r) = Lev^{past}_t(r)$ with: (**a**)

$$Lev^{past}_t(r) = C(r) + \sum_{\substack{p \in P(r)/t^p_{ub} < t \\ \text{and } a^p \notin RA_t}} ?q_p - \sum_{\substack{c \in C(r)/t^c_{ub} < t \\ \text{and } a^c \notin RA_t}} ?q_c$$

[2] In IxTeT, $LgcS_t$ contains the last executed *event* for each la.

[3] In IxTeT, a goal is represented by a grounded proposition $hold(GoalAtt(g) : GoalValue, (st^g, et^g))$. G_t contains goals such that $et^g_{ub} \geq t$.

This level is a variable ranging over $[lev^{past}_{lb}, lev^{past}_{ub}]$.

In case (2), the uncertainty follows from the insufficient piece-wise constant model of the resource usage. The previous level definition is completed by an estimation of the level modification by the running actions ($Lev^{RA}(r)$) according to formula (**b**). The total amount produced, consumed or borrowed by an action is represented by a variable $?q$ in $[q_{min}, q_{max}]$. At a given time in the course of the action, the only information that the planner can deduce is that the amount produced/consumed up to now is in $[0, q_{max}]$ or that the amount borrowed is in $[q_{min}, q_{max}]$. An estimation of the level would then be: (**b**)

$$lev^{RA}_{min} \leq RscL_t(r) - Lev^{past}_t(r) \leq lev^{RA}_{max}$$

$$\text{with } lev^{RA}_{min} = - \sum_{c \in C(r)/a^c \in RA_t} q_{c_{max}} - \sum_{b \in B(r)/a^b \in RA_t} q_{b_{max}}$$

$$lev^{RA}_{max} = \sum_{p \in P(r)/a^p \in RA_t} q_{p_{max}} - \sum_{b \in B(r)/a^b \in RA_t} q_{b_{min}}$$

Finally, the level of a resource r at time t is comprised in the interval $[RscL_{lb}, RscL_{ub}]$ with, in case (1):

$$RscL_{lb} = lev^{past}_{lb}, RscL_{ub} = lev^{past}_{ub},$$

and in case (2):

$$RscL_{lb} = lev^{past}_{lb} + lev^{RA}_{min}, RscL_{ub} = lev^{past}_{ub} + lev^{RA}_{max}.$$

A timepoint in the temporal network may correspond to a goal timepoint or to an action starting or ending timepoint.

Definition 2 (executable timepoint) *A timepoint T is executable at time t if all timepoints T^p that must directly precede it in the temporal network have already been executed ($T^p_{lb} = T^p_{ub} < t$) and if $t \in [T_{lb}, T_{ub}]$.*

A goal is instantaneously achieved or persistent (achieve and maintain a property between st^g and et^g).

Definition 3 (achievable goal) *A goal g is achievable at time t if st^g is executable and if $g \notin F_t$.*

Let A^f_t be the set of actions that are involved in F_t. [4]

Definition 4 (executable action) *A future action a is executable at time t if its start timepoint is executable and if $a \notin A^f_t$.*

Definition 5 (executable plan) *A partial plan P_t is executable at time t if the constraint networks are consistent and if $RA_t \cap A^f_t = \emptyset$.*

Execution cycle

The solution plan produced by the planner is run by the executive following a three phases "sense/plan repair/act" cycle. The executive wakes up when a message has been received, or when it is time to execute some timepoint or when a plan repair process is in progress. Let us call $ExecutingPlan$ the plan being executed. *Sensing* consists in integrating messages in $ExecutingPlan$ which may partially invalidate it. If

[4] The determination of A^f_t is straightforward in the case of open conditions and resource conflicts. In a threat case, an action a_k has effects in contradiction with the establishment of proposition p by the causal link $a_i \xrightarrow{p} a_j$ and $(a_i \prec a_k \prec a_j)$ is consistent. A^f_t contains a_k and a_j.

$ExecutingPlan$ contains new flaws, a *plan repair* consists in keeping the structure of the plan (the ordering of actions) and taking advantage of the flexibility to try and find a solution plan. Planning is distributed on several cycles to allow reactivity to events and concurrent execution of the valid part. *Acting* consists in determining which timepoints have to be executed, processing them or detecting time-out. The user defines the maximum time allowed to the cycle, also called *timestep* (ts).

Distributing planning on several cycles raises two problems:

On which plan relies the execution in the "act" part, especially if no solution has been found? This plan has to be *executable*. At each planning step, the node is labeled if the current partial plan is *executable*. When the maximum time allowed to plan repair in the cycle is reached, the last labeled partial plan becomes $ExecutingPlan$.

On which plan and which search tree relies the planning process in the next cycle? If no decision has been made meanwhile (no timepoint execution, no message reception), the search tree can be kept as is and further developed during the next "plan repair" part. However, if the plan has been modified, a new search tree whose root node is the new $ExecutingPlan$ is used, and the planning decisions made in previous cycles are final.

The following subsections further detail the different phases of the cycle. Basically, all modifications made to $ExecutingPlan$ have to guarantee that an *executable* plan is available after each phase of the cycle. If this condition does not hold, the cycle is stopped and a complete replanning is mandatory. During a cycle without plan repair, $ExecutingPlan$ remains a solution plan.

Message integration

A message can be: a report upon action completion, a new goal request or a notification of a capacity alteration.

Report A report is associated with the ending timepoint et^a of the corresponding action a. If the message is received inside the bounds $[et^a_{lb}, et^a_{ub}]$, et^a is set to the current time t (equivalent to posting the constraint $(et^a - origin) = t$ in the STN). Otherwise, two situations arise. If there is no flexibility left in the plan, it is not executable anymore. Else, a new end timepoint, set to t and constrained to occur before the executable timepoints, is created and the failed one is relaxed. The network is then recomputed. In IxTeT, such an operation keeps the network consistent since the only constraint that can be specified between two actions a and a' is a precedence constraint which upper bound is flexible: $(st^{a'} - et^a)$ in $]0, +\infty[$. If the report contains information about the state, S_t is updated in the following way:

Resource level - For each resource r, the report returns the current "real" level l_r. $RscL_t(r)$ is compared to l_r and adjusted.
— If $l_r \in [RscL_{lb}, RscL_{ub}]$, the plan is consistent with reality. If the exact level can be computed (case (1) - see Definitions), its value is updated by adding the constraint $RscL_t(r) = l_r$.
— If $(l_r < RscL_{lb})$, the over-consumption is reflected on the plan by adding a consumption of quantity $c = RscL_{lb} - l_r$. In case (1), the new level ($RscL'_t(r) = RscL_t(r) - c$) is updated by adding the constraint $RscL_t(r) = RscL_{lb}$, equivalent to $RscL'_t(r) = l_r$. If some running action modifies r (case (2)), the part of the level due to past actions ($L^{past}_t(r)$) is updated by considering the worst case and adding the constraint $L^{past}_t(r) = lev^{past}_{lb}$.
— If $(l_r > RscL_{ub})$, the over-production is reflected on the plan by adding a production of quantity $p = l_r - RscL_{ub}$. Similarly, the level is updated by adding the constraint $RscL_t(r) = RscL_{ub}$ in case (1), and $L^{past}_t(r) = lev^{past}_{ub}$ in case (2).

The plan remains executable in case (1) since no running action is concerned by r, whereas conflicts may appear in case (2), requiring execution abortion. If the plan is executable, plan repair is requested in case of over-consumption and in case of over-production of a reservoir resource.

State variables - $LgcS_t$ contains the last value for each instantiated logical attribute. If the report is nominal, $LgcS_t$ is updated with the effects of a expected in the plan. Otherwise, it is updated with the values returned in the report. A value is not inserted if it leads to a non executable plan (threatens some proposition of a running action a_r). In that case and if a_r is preemptive, its interruption is requested. Else, the value is inserted and causal links which contradict it are broken. This update leads to an executable plan with open conditions on which plan repair can be processed.

Goal request One specifies for each goal: a priority, a duration interval I^d, an estimation of its minimal achievement duration d_{achiev} and a time interval I^g for its start timepoint st^g. A goal is rejected if the following set of constraints is not consistent with the plan at current time t: $\{((et^g - st^g)$ in $I^d)$, $((st^g -$ origin$)$ in $I^g)$, $(st^g$-origin $\geq t + d_{achiev})\}$. Moreover the goal is constrained to occur after each running action a such that the goal proposition threatens a proposition in a. A goal integration results in an executable plan with an open condition.

Capacity alteration As a consequence of an external event a resource capacity has increased or decreased by a quantity q. The capacity is updated. If the plan is not executable anymore, execution is aborted. Otherwise a plan repair is requested in case of decrease or increase of a reservoir resource.

Causal links removal After message integration, the plan may contain flaws (open conditions and/or resource conflicts) on a set of grounded attributes Att^f, possibly repaired thanks to the insertion of new actions. Let us consider Att^i the set of the attributes appearing in the potentially inserted actions. Additional causal links, protecting propositions in the plan on attributes in Att^i, have to be broken to allow the insertion of these actions in the current plan structure.

The determination of Att^i is based on information given by an abstraction hierarchy verifying the Ordered Monotonicity Property (Knoblock 1994; Garcia & Laborie 1995) and generated offline from the model description. Notably, this hierarchy points out the primary effects of an operator, which justify its insertion to solve a flaw. Let us call *main attributes* of an action the attributes appearing in its primary effects. Att^i, initialized with Att^f, is computed by searching the action operators for which at least one attribute att_m in Att^i is a main attribute. This operator is partially grounded (by binding its corresponding parameter with att_m) and the (eventually grounded) attributes appearing in the operator and not yet taken into account are added to Att^i. The algorithm proceeds recursively until a fixed point is reached.

Finally, the partial plan is executable and the sets of actions independent from the failures remain executable.

Plan repair

The plan repair is similar to the IxTeT search process in the plan space. The root of the search tree $\mathcal{T}$ is $ExecutingPlan$, partially invalidated. $\mathcal{T}$ is developed according to an ordered depth

first search strategy. Planning is distributed, if necessary, on several cycles and each time a new timepoint is inserted, it is constrained to occur after the end of the current cycle. Planning during one cycle is done one step at a time until it results in a dead-end (there is no solution), or a solution is found or a deadline is reached. This deadline corresponds to the share (μ) of the timestep ts allocated to the plan repair part (this parameter is tuned by the user).

Some aggregation mechanisms (corresponding to past forgetting) allow a reduction of the search space. In IxTeT, the establishing events are looked for in $LgcS_t$ and executed resource propositions are aggregated in one proposition.

This plan repair process is not guaranteed to find a valid plan, still it can avoid aborting execution and completely replanning at each failure. By invalidating only a part of the plan, the amount of decisions is rather limited and a repaired plan may be found in a few cycles. Plan repair is especially efficient and useful for temporally flexible plans and plans with some parallelism. This mechanism is also efficient to compensate for inadequate models of actions. Consider a move(L_1,L_2) action, which is defined as a late preemptive action in the IxTeT-ExeC model. If the robot takes longer than expected in the model (e.g. due to obstacle avoidance), the action is interrupted. The controlled system returns the intermediate location L_i and, if some temporal flexibility remains, a new move(L_i,L_2) is immediately inserted and launched. This example is representative of the failures that frequently break plan execution.

Action

Each timepoint is associated with an *execution time* t_{exec}. If T is a start or goal timepoint, or an end timepoint of an early preemptive action, $t_{exec} = T_{lb}$. If T is an end timepoint of a late preemptive action, $t_{exec} = T_{ub} - ts$. If T is an end timepoint of a non preemptive action, $t_{exec} = T_{ub}$. During the "act" part, the executive determines the set of timepoints to execute during the current cycle ($ExecTPs$): these timepoints are executable and their execution time happens before the end of the cycle. $ExecTPs$ is updated after each timepoint execution to take into account newly executable timepoints. The detail of a timepoint execution depends on its type. If a goal is not achievable or if an action is not executable and if some flexibility remains, their execution is postponed. The execution time of a preemptive end timepoint is set when the corresponding report is received (expected in the next cycle). Finally, timeouts are detected when reports have not been received in time.

Complete replanning

Let us call $P_{t_s} = (\emptyset, FA_{t_s}, S_{t_s}, G_{t_s}, C_{t_s}, L_{t_s}, F_{t_s})$ the plan obtained once execution is stopped. An initial plan is extracted from P_{t_s} as:
$P_{t_i} = (\emptyset, \emptyset, S_{t_i}, G_{t_i}, C_{t_i}, \emptyset, F_{t_i})$, with
$S_{t_i} = S_{t_s}$, $G_{t_i} = \{g \in G_{t_s} / \textit{temporal constraints on g are coherent with current time}\}$, $C_{t_i} = \{c \in C_{t_s} / \textit{c is a constraint just on variables appearing in } S_{t_i} \textit{ and } G_{t_i}\}$($C_{t_i}$*notably contains constraints on origin and horizon timepoints)*, and $F_{t_i} = G_{t_i}$.

POCL planning can not be interrupted at any time and come up with an applicable plan. Still we have to guarantee that at the end of the replanning process, there remains enough time to execute the solution plan and meet the goal deadlines. We propose to add a specific flexible timepoint T^{end} to P_{t_i}, that corresponds to the end of the planning process. T^{end} is only constrained to occur between t_i and the end of the horizon. Each time a new timepoint is inserted by the planning process, it is constrained to occur after T^{end}. Thus T^{end}_{ub} decreases as new actions or new temporal constraints are added, and there is not enough time to execute the current plan if $T^{end}_{ub} <$ current time. Note however that T^{end}_{ub} can increase when backtracking.

The strategy is then to plan one step at a time until it results in a dead-end, or a solution is found, or a time limit l is reached. l is defined as $l = T^{end}_{ub} - d$, d being a *slack* duration to save enough time at the end of planning for cycle initialization. l is updated after each planning step. Planning is stopped when l is reached unless the next step corresponds to a backtrack node. In that case, and if the next step increases l, planning is pursued.

If planning is aborted without finding a solution, some goals are rejected and a new attempt is done. A new plan P_{t_i} is extracted from P_{t_s}. At this point, some goals may have been rejected due to temporal constraints inconsistent with the current time. Otherwise, we abandon the goal with the lowest priority, and, if several goals have the same priority, the goal with the less flexibility for its achievement (this flexibility is computed as (st^g_{ub} - d_{achiev})). This criterion has been chosen to keep the goals that are more likely to be achieved in due course.

Integration and example of scenario

IxTeT-ExeC has been integrated in the decisional level of the LAAS architecture (Alami *et al.* 1998) and used to control an iRobot ATRV. IxTeT-ExeC interacts with the user and the functional level through a procedural executive (OpenPRS). The plan execution is controlled by both executives as follows. TeXeC decides when to start or stop an action in the plan and handles plan adaptations. OpenPRS expands the action into commands to the functional level, monitors its execution and can recover from specific failures. It finally reports to TeXeC upon the action completion.

However, we face some difficulties to evaluate our work since there is no benchmark (involving time and resource issues) to apply the system to. We illustrate the performances of IxTeT-ExeC with an example of failure scenario for a simulated rover with an exploration mission. In such a domain, the quantitative effects and durations can be estimated in advance for planning but are accurately known only at execution time (e.g. the actual compression rate of an image or the actual duration of a navigation task), thus requiring regular updates and look-ahead capabilities to manage unforeseen situations and resource levels.

Goals are a list of targets to take a picture of, constraints correspond to deadlines and limited resources (memory storage, battery level). The planning model contains five actions: move, take_picture, move_ptu (change the orientation of the cameras through commands to the pan&tilt unit), download images (to free memory storage) and recharge battery (the robot stays still while solar panels recharge the battery). The sequence of actions to achieve a goal is: go to the target, change the camera orientation, take a picture.

IxTeT-ExeC is interfaced with the procedural system OpenPRS which simulates the robot state evolution. Figure 2 describes an example of IxTeT-ExeC reactivity to a scenario with multiple failures. The initial plan contains 5 goals (g_1,...,g_5). The third goal has a strict deadline and a higher priority. Since storage capacity is limited to 3 images, the plan contains a download action after g_3. The battery initial level is sufficient to complete the plan. The following events occur:
— (e_1) **action failure** - The first take picture fails, plan repair

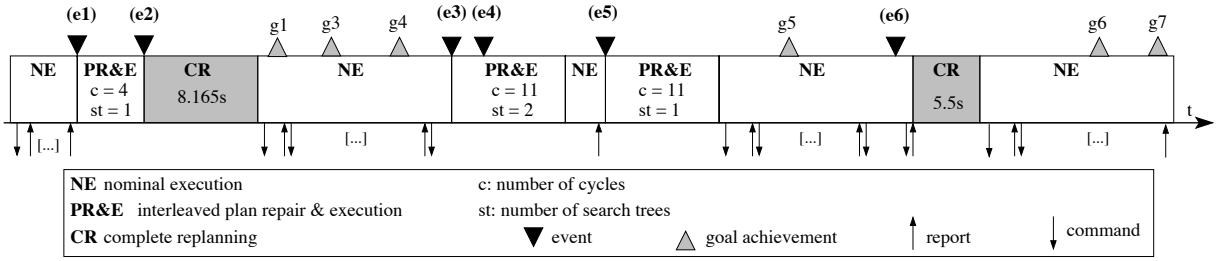

Figure 2: Example of scenario and reactions.

begins.

— (e_2) **timeout** - Due to g_3 deadline, the next executable time-point times out. After a complete replanning, g_2 has been removed. Further plan execution achieves g_1, g_3 and g_4.

— (e_3) **new goal** - During the download action, a new flexible goal g_6 is received. Plan repair begins.

— (e_4) **new goal** - A new goal g_7 is received 3s later. The repaired plan finally contains three goals not yet achieved.

— (e_5) **sub-production** - The download action has produced less than expected, the new level allows to take only two images. Plan repair leads to the insertion of a download action before the last take picture action.

— (e_6) **capacity alteration** During a move action, a sudden drop in battery level forbids the plan completion without recharging. Even if the current action completion is not directly threatened by the drop, it takes part in the future resource contention. Thus execution is aborted and a complete replanning leads to the insertion of an immediate recharge action. The plan can then be successfully completed. Note that the battery and memory storage levels are regularly adjusted. This example has been run on a Pentium IV, with $ts = 1.3s$ and $\mu = 61\%$. The average cycle duration during nominal execution is 0.22s.

Conclusion and prospectives

We have presented the IxTeT-eXeC system which combines a temporal lifted POCL planner with a temporal executive to integrate deliberative planning, execution monitoring and replanning while respecting real-time constraints. This approach cannot account for all the possible execution failures in all their generality, nevertheless, in many situations where some temporal and resource flexibility has been left, one can expect the presented repair techniques to greatly improve the overall performance of the system by:

— reducing the number of complete replannings,
— improving the system reactivity to unexpected events,
— taking into account new goals on the fly,
— managing the resources capacity changes,
— managing the uncertainty in the model description (actions duration, consumption/production).

Still, IxTeT-eXeC effectiveness can be increased by improving replanning strategies (rejected goals selection, state update requests) and handling temporal non controllability earlier in the planning process and propagation algorithms (which may currently squeeze contingent durations). We plan to use the STNU framework and adapt the polynomial Dynamic Controllability Checking algorithms proposed in (Morris, Muscettola, & Vidal 2001). The current results are very encouraging and, providing we use proper heuristics for the planning process, we envision using this approach for more complex domains. More results are available in (Lemai 2004).

References

Alami, R.; Chatila, R.; Fleury, S.; Ghallab, M.; and Ingrand, F. 1998. An architecture for autonomy. *IJRR*.

Ambros-Ingerson, J., and Steel, S. 1988. Integrating planning, execution and monitoring. In *National Conference on Artificial Intelligence*.

Chien, S.; Knight, R.; Stechert, A.; Sherwood, R.; and Rabideau, G. 2000. Using iterative repair to improve the responsiveness of planning and scheduling. In *National Conference on Artificial Intelligence*.

Dechter, R.; Meiri, I.; and Pearl, J. 1989. Temporal constraint networks. In *KR*.

Despouys, O., and Ingrand, F. 1999. Propice-plan: Toward a unified framework for planning and execution. In *ECP*.

Finzi, A.; Ingrand, F.; and Muscettola, N. 2004. Robot action planning and execution control. In *International Workshop on Planning and Scheduling for Space*.

Garcia, F., and Laborie, P. 1995. Hierarchisation of the search space in temporal planning. In *European Workshop on Planning*.

Haigh, K. Z., and Veloso, M. M. 1998. Planning, execution and learning in a robotic agent. In *AIPS*.

Ingrand, F.; Chatila, R.; Alami, R.; and Robert, F. 1996. PRS: A high level supervision and control language for autonomous mobile robots. In *International Conference on Robotics and Automation*.

Knoblock, C. 1994. Automatically generating abstractions for planning. *Artificial Intelligence* 68.

Laborie, P., and Ghallab, M. 1995. Planning with sharable resource constraints. In *IJCAI*.

Lemai, S. 2004. IxTeT-eXeC: planning, plan repair and execution control with time and resource management. *PhD Thesis, INPT, LAAS-CNRS*.

Levinson, R. 1995. A general programming language for unified planning and control. *Artificial Intelligence* 76.

Morris, P. H.; Muscettola, N.; and Vidal, T. 2001. Dynamic control of plans with temporal uncertainty. In *IJCAI*.

Muscettola, N.; Nayak, P. P.; Pell, B.; and Williams, B. 1998. Remote agent : To boldly go where no ai system has gone before. *Artificial Intelligence* 103.

Myers, K. L. 1999. CPEF: Continuous planning and execution framework. *AI Magazine* 20(4):63–69.

Simmons, R., and Apfelbaum, D. 1998. A task description language for robot control. In *Conference on Intelligent Robotics and Systems*.

Trinquart, R., and Ghallab, M. 2001. An extended functional representation in temporal planning : towards continuous change. In *ECP*.