Controlability and Makespan Issues with Robot Action Planning and Execution

Matthieu Gallien and **Félix Ingrand**
LAAS-CNRS*
7, Avenue du Colonel Roche, 31077 Toulouse Cedex 4, France
{matthieu.gallien,felix}@laas.fr

Abstract

In recent years, there has been an increasing interest in getting planners to "execute" their plans in the real world. Among them, the temporal "partial order causal link" planner IxTeT has been endowed with a temporal executive, and has been successfully tested on an ATRV robot. Yet new problems arises from the fact that the plan are now being executed. In domains such as robot planning, many tasks have an uncertain duration (a move in a partially unknown environment takes an uncertain amount of time). The current solution is to use an interval constraint to represent the possible duration in a Simple Temporal Network. By doing this, the planner or the executive are allowed to reduce the possible durations by propagating other constraints. Hence, this reduction may lead to a plan execution failure. This problem, known as the "controllability" problem, as been addressed by a number of people. For example, the 3DC+ algorithm has been proposed to provide dynamic controllability. Yet to our knowledge, no implementation has been made showing the advantage of such approach. This paper presents the results we got using 3DC+ in IxTeT.

Having now a temporal planner which now "complies" with non controllable temporal constraints, we addressed a second problem which arises at execution time. IxTeT being a least commitment planner, the system does not try to minimize the makespan. In fact, the default heuristic is tuned more to spread the actions over the given horizon (to minimize conflict) than to finish the job at the earliest. We implemented a new heuristic which keeps the least commitment strategy yet minimizing the makespan of the plan. This heuristic gives good statistical results on the model we use for our exploration rover type problem.

We have tested these two recent modifications on the robot, but also using a very accurate rover simulator. We present real results and simulated one which show the improvement over the previous version/heuristic. We also proposed a solution to a drawback identified during these tests.

Introduction

Future rover exploration missions will require a high level of autonomy in order to achieve goals such like navigation out of visual range. On board task planning should allow the rover to react without needing a new plan to be uploaded from Earth. Another important challenge is to execute those plan autonomously and reacting when something goes wrong.

During the past years some planners have proven their ability to handle complex situations required by autonomous systems. Some of these systems (e.g. RAXPS (Jonsson *et al.* 2000), CASPER (Chien *et al.* 2005)) have been deployed. Reasoning about time is necessary to address these planning problems. The planner must be able to take into account strict deadline, temporal windows for some tasks, durative actions, and durative goals.

Many temporal formalisms were developed in the past years[1]. The STN[2] formalism is often used. The requests on these networks are solved very efficiently by polynomial algorithms. Nowadays, an extension to uncertain constraints has been studied and polynomial algorithms have been proposed.

Actual robotic space exploration mission are very expensive, with a high requirement for quality scientific returns. During the MER mission, the use of MapGen (Ai-Chang *et al.* 2003) has allowed a 25% increase of such return (Rajan 2004). In an autonomous planner, optimization can be made in two ways: finding directly a good plan or searching several plans to find the better one. Due to limited computational capacity, the second approach is often unreasonable. So we have to modify the planner to search for high quality solutions.

The IxTeT planner[3] (Ghallab & Laruelle 1994) was developed to handle robotic planning problems. It was extended to handle complex resources (Laborie & Ghallab 1995), continuous constraints and constraints between both atemporal and temporal variables (Trinquart & Ghallab 2001). Further work (Lemai 2004) added a temporal executive to IxTeT. The resources management has been extended to handle continuous resources usage (Lemai & Ingrand 2004).

New issues were raised while experimenting with IxTeT new executive. Many plans failed because the executive or

*Part of this work has been funded by a grant from the ESF (European Social Fund)

[1] Originally IxTeT was a temporal formalism (Ghallab & Mounir-Alaoui 1989) used to check consistency other qualitative temporal relations.

[2] STN: Simple Temporal Network

[3] IxTeT is a system used for chronicle recognition, planning and temporal execution.

the planner reduced the allowed duration of some tasks. In fact, most of the tasks (move, communication , image compression, etc) have uncertain durations. So we decided to implement an existing temporal formalism with explicit uncertainties (STNU). The planner produces temporal contingent plans, i.e. plans that do not fail because of a reduction of an uncertain duration.

Another issue is that while executing the plan, the robot often remains inactive. In fact, the plans produced with a least commitment heuristic may contain unjustified wait periods. Though, they may be necessary to adapt the plan during execution, they reduce the performance of the overall system. This raises a need for plans with a shorter makespan. We solve this problem by modifying the planning heuristic.

These new contributions must be validated by experimental returns. So, we integrated IxTeT and a procedural executive (OpenPRS) to a simulated rover controlling architecture. This simulator allows to make accurate comparisons between the modified versions and the old one.

During the analysis of the results, we identify a drawback of the plan repair mechanism. We propose a solution to it by modifying the way the plan are repaired. This is an ongoing work and some preliminary results are shown.

This paper is organized as follows. The first and second sections describe the IxTeT planner and the embodied executive. The third one describes the validation scenario and the fourth describes the experimental results. The last one introduces a work on improving the plan repair method. We conclude the paper with a discussion and future works.

The IxTeT Planner

Plan-space Search

IxTeT is a temporal constraint-based causal link planner. It uses CSP techniques[4] to maintain the consistency of the plan constraints. In particular, the planner is using a Simple Temporal Network (Dechter, Meiri, & Pearl 1991) for the temporal constraint.

The underlying representation of the plan is state variables ranging over finite and real valued domains. These plans use chronicles (Ghallab, Nau, & Traverso 2004) to describe the world, its evolution and the planning problem. The explicit representation of the time permits to have temporally extended goals, durative actions and rendez-vous or visibility windows.

Definition 1 *A* temporal assertion *on a state variable v is either an* event *or a* persistence condition *on v.*

- *An event, denoted $x@t : (v_1, v_2)$, specifies an instantaneous change of the value of x from v_1 to v at time t, with $v_1 \neq v_2$.*
- *A persistence condition, denoted $x@[t_1, t_2) : v$, specifies that the value of x persists as being equal to v over the interval $[t_1, t_2)$.*

In any temporal assertion, v, v_1 and v_2 can be defined by atemporal variables. t, t_1 and t_2 are temporal variables.

[4]Constraint Satisfaction Problem (Mackworth 1977) (CSP)

Definition 2 *A* chronicle *for a set of state variables $v_1, v_2, ..., v_n$ is a pair $\Phi = (F, C)$, where F is a set of temporal assertions about the state variables $v_1, v_2, ..., v_n$ and C is a set of constraints on variables used in the chronicle.*

Definition 3 *A plan $\mathcal{P}(S, \Phi, G, CA, F, T)$ is described by the state variables contained in S. Φ is a chronicle describing all the temporal assertions of the plan. F is the set of defaults in the plan. $CA \subset \Phi$ contains temporal assertions on variable of S describing the predicted evolution of contingent attributes. The goals are in $G \subset \Phi$, they are temporal persistence conditions on state variables of S. T is the set of tasks in the plan.*

The planner begins with a plan describing the initial situation, the initial goals of the problem and the known predicted evolutions of contingent attributes such as visibility windows. The search is performed until the plan contains no default. These defaults are temporal assertions unexplained in the current plan[5], conflicts between two temporal assertions or possible resource conflicts. At each search step, a default is chosen. One of the resolvants of this default is then applied.

The search toward a solution is performed using a dynamic abstraction hierarchy (Garcia & Laborie 1995) on the attributes of the planning domain. The hierarchy provides a dynamic ordering on the resolution of the defaults in the plan. This ordering can dramatically improve the speed of the planner.

The heuristic computes a cost for each resolver of each default. Then, it chooses the next default to solve ρ using an opportunistic strategy. A default is preferred if it has less resolvants than the other and if it is easier for the planner to choose one among them. The cheapest resolvant is then chosen. The choice is considered as easier for example if one resolvant cost is 0.1 and the other cost is 0.9. We have a better confidence in the automated choice if the difference between costs is high.

$$\frac{1}{\text{Opp}(\rho)} = \sum_{r \in \text{ resolvant}(\rho)} \frac{1}{1 + \text{cost}(r) - \text{cost}_{\text{min}}(\text{resolvant}(\rho))}$$

Theoretically the commitment of one resolvant is computed by estimating the number of completely instantiated reachable solution plans removed by the resolvant r. In fact, estimation were developed. A complete description of the cost functions can be found in (Lemai 2004).

We have implemented a new version of the heuristic with two modified costs. The first considers one single ordering resolvant. The second evaluates the cost of one persistence condition on a state variable.

1. Without the ordering constraint $(t_1 < t_2)$, t_1 can be instantiated any time before or after t_2 with respect to the other temporal constraints. The cost of the resolvant depends on the number of possible instantiation of t_1 after t_2 removed by the constraint.

[5]A temporal assertion is not explained by a plan if it is not an initial condition or if no causal link establishes the assertion.

$c(t_i, t_j)$ is the temporal minimum constraint between t_i and t_j. $d_{\min}(c(t_i, t_j))$, respectively $d_{\max}(c(t_i, t_j))$ is the minimum, respectively maximum duration of the constraint $c(t_i, t_j)$.

$$\text{old\_cost}(t_1 < t_2) = \frac{d_{\max}(c(t_1, t_2) \cap [-\infty, 0])}{d_{\max}(c(t_1, t_2))}$$

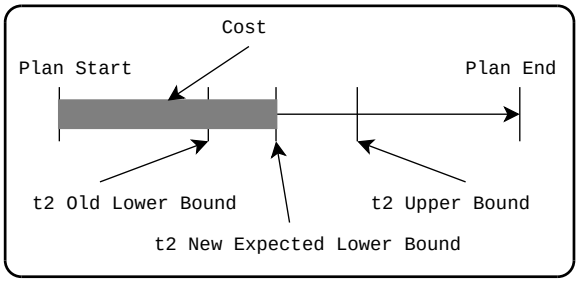

Figure 1: How the new heuristic evaluates the cost of an ordering constraint.

The new heuristic measures the earliness of the resolvant from the point of view of one timepoint. We choose the point of view of t_2[6] (see Fig.1). The new expected lower bound of t_2 is its old lower bound if the lower bound of t_1 is lower than the one of t_2. In the other case, it is the lower bound of t_1.

If $d_{\min}(c(t_{\text{plan\_start}}, t_1)) < d_{\min}(c(t_{\text{plan\_start}}, t_2))$ then

$$\text{new\_cost}(t_1 < t_2) = \frac{d_{\min}(c(t_{\text{plan\_start}}, t_2))}{d_{\min}(c(t_{\text{plan\_start}}, t_{\text{plan\_end}}))}$$

else

$$\text{new\_cost}(t_1 < t_2) = \frac{d_{\min}(c(t_{\text{plan\_start}}, t_1))}{d_{\min}(c(t_{\text{plan\_start}}, t_{\text{plan\_end}}))}$$

2. A causal link resolvant is the conjunction of an ordering constraint, an equality constraint between the values of the two connected temporal assertions and a persistence condition on the state variable. The cost of such resolvants is the sum of the cost of each elementary constraints and of a cost depending on the duration of the persistence condition.

   The old cost of the temporal persistence is the ratio of the minimum duration of the persistence by the minimum duration of the plan.

   The new cost is the sum of the equality constraints and the cost of the persistence condition. The temporal cost is ignored even if the constraint is necessary. This allows the heuristic to order the causal links only by their respective durations.

   We modified the cost of the persistence condition. The new cost is the ratio of the maximum duration of the persistence condition by the maximum duration of the plan.

   The new cost uses the maximum duration. It represents possible negative effects of causal link with a possible big upper bound on the duration.

[6]Tests have been made with t_1 without significant differences.

Underlying CSPs

Definition 4 *A CSP (Mackworth 1977)* $\Gamma = (V, D, C)$ *is defined by* $V = \{v_i \mid i = 1, ..., n\}$ *the set of all variables,* $D = \{d_i \mid i = 1, ..., n\}$ *the set of variable domains. Finally,* $C = \{c_i \mid i = 1, ..., p\}$ *is the set of constraints on variables contained in* V.

IxTeT uses classical CSPs algorithms for managing constraints on atemporal variables. It uses an STN for managing all the temporal constraints.

In some cases, one wants to link the effects of a task to its duration. For example, if the consumption of a resource depends on the duration of the task, you need a mixed constraint between temporal and atemporal variables. IxTeT features a mechanism (Trinquart & Ghallab 2001) to propagate these constraints.

For example, let t be a navigation task of expected duration d. The speed of the robot is represented by the variable s and the length of the navigation by l. The mixed constraint $c_{=}\{d * s = l\}$ links the duration of the task to the length of the trajectory.

Definition 5 *An STN (Dechter, Meiri, & Pearl 1991)* $\Theta = (V, D, C)$ *is a restricted form of CSP.* $V = \{v_1, ..., v_n\}$ *is the set of variables.* $D = \{d_i \mid \forall i \; d_i = \mathbb{R}\}$ *is the set of the variable domains.* C *is the set of constraints on variables of* V. *The constraints are all of the form:* $lb \leq v_i - v_j \leq ub$ *which is equivalent to* $v_i - v_j \in [lb, ub]$

On the STN, IxTeT uses a path consistency algorithm like PC-2 (Mackworth & Freuder 1985) that is able to compute the minimal network of an STN (thus removing all values not belonging to a solution). The time complexity is $O(n^3)$ for the complete algorithm and an incremental one[7] is only in $O(n^2)$. During planning, the planner uses the incremental algorithm. During a nominal execution, the same algorithm is used to propagate instantiations of action start and end timepoints. If a temporal failure invalidates some temporal constraints, a relaxation is done by removing the failed constraints and by repropagating all the remaining constraints with the complete algorithm.

It is possible to use a fast algorithm (Muscettola, Morris, & Tsamardinos 1998) for STN execution. It is based on the property of dispatchability. However it is much efficient than using an $O(n^2)$ algorithm, we cannot use it because the propagation of mixed constraints needs a complete algorithm.

Simple Temporal Network with Uncertainties

Definition 6 *An STNU (Vidal & Fargier 1999)* $\Theta = (V, D, C_{clb}, C_{ctg})$ *with* V *the set of variables,* D *the set of domains. These definitions are the same than the STN ones. The set* C_{clb} *is all the controllable constraints equivalent to STN constraints.* C_{ctg} *is a set of uncontrollable constraints*

[7]If only one constraint of the STN is restricted, the incremental algorithm can be used.

of the form lb $\leq v_i - v_j \leq$ ub. The duration of this constraints can only be observed.

The introduction of not controllable constraints changes the consistency notion inherited from the STN. Three main levels of controllability have been defined (Vidal & Fargier 1999). Schematically, an STNU is weakly controllable if the execution controller needs to know all the uncontrollable durations before executing the plan. An STNU is strongly controllable if the execution controller instantiates the controllable timepoint at the same time whatever the observed uncontrollable durations are. Finally, an STNU is dynamically controllable if the execution controller must take decisions knowing only the past observations and timepoint instantiations. Finally an STNU is pseudo-controllable if it is consistent and no uncontrollable durations are squeezed (Morris, Muscettola, & Vidal 2001).

The two possible choices are dynamic or strong controllability. The two possible properties can be checked in polynomial time. The first one is more flexible and may have a lower makespan[8]. The second one has a lower complexity during execution. This controllability is more restrictive than the dynamic one, fewer STNUs are strongly controllable. We decided to experiment the dynamic controllability due to these advantages.

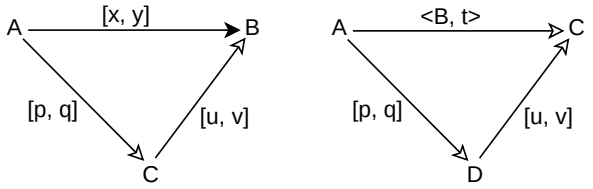

Figure 2: Basic constraint triangles analyzed by 3DC+. The left triangle is used when searching for new necessary constraints. The right one is checked during the propagation of a new ternary constraint called a "wait".

The algorithm 3DC+ (Morris, Muscettola, & Vidal 2001) checks if an STNU is dynamically controllable. It is polynomial in time, computes the minimal graph and the most flexible one. It is complete and correct. It allows to produce plans and to safely execute them.

The algorithm performs several cycle of checking on triangles containing at least one uncontrollable constraint. On each triangle (see Fig.2), a test is performed in order to identify the possible necessary constraints to enforced dynamic controllability. Four cases exist:

- In the case where C necessary follows B, nothing new is needed.
- In the case where C necessary precedes B, a new constraint on AC is needed : $(C - A)$ in $[y - v, x - u]$.
- In the other cases C and B are unordered. If $y - x \leq v$ then we must add the constraint $(C - A)$ in $[y - v, +\infty[$.
- Otherwise, a new type of constraint called "wait" is needed: $< B, y - v >$. It means that before executing C you must wait $y - v$ after the execution of A or the observation of B.

At the start of each cycle, the APSPG[9] is computed. Then, the new necessary constraints are added. After the analysis of all triangles, the algorithm propagates all the waits to ensure they will be never broken during execution. The termination condition is reached if the network is not pseudo-controllable or if no new constraint has been added.

We have made two little improvements to this algorithm. The first concerns the fact that we want to incrementally maintain an STNU dynamically controllable during planning. Before the planner adds a new constraint to the network, our algorithm removes all existing waits. This is due to the fact that some of them are no more necessary due to the new constraint. More work is needed to find a better removal criterion.

The original algorithm performs several complete propagation with a complexity of $0(n^3)$ in order to test the property of pseudo-controllability. We have replace this by maintaining always the network pseudo-controllable by incrementally propagating all addition of STN constraints during enforcement of dynamic controllability. If we suppose that k constraints are added during a cycle, the complexity is in $O(kn^2)$. Experimentally, we observed that k is rather constant. Yet the maximum bound of k is in $O(n)$, the maximum number of uncontrollable constraints is $n - 1$.

Autonomous Plan Execution

Temporal Plan Execution

During execution, IxTeT controls the beginning, ending and interruption of tasks running on board the rover, by sending commands to the robot. The executive begins with an initial plan produced by the planner. This section of the paper briefly summarizes the work described in (Lemai 2004).

IxTeT executes the plan following a cycle. If a new event is received, it updates the plan. After that, if needed, a plan repair process occurs. At the end, it converts the executable timepoints into commands and updates the plan accordingly. The cycle has an expected maximal duration denoted μ.

For simplicity, the executive only considers the start timepoint t_{start} and the end timepoint t_{end} of tasks.

Typically, the system starts a task $t(t_{start}, t_{end}, T, P)$ as soon as possible if its first timepoint t_{start} is executable and the actual plan supports its execution.

The system only interrupts tasks that can be preempted. Other tasks report their termination by sending a message summarizing the exact result of their execution.

[8]If all the uncontrollable durations are equal to their upper bounds, enforces dynamic controllability make a plan with a makespan equal to the one produced by enforcing the strong controllability. Otherwise, dynamic controllability is more time efficient.

[9]APSPG: is the All Pairs Shortest Path Graph of the STNU. If one uncontrollable duration is squeezed, then the STNU is not pseudo-controllable and thus not dynamically controllable. Our implementation uses a PC-2 algorithm (Mackworth & Freuder 1985) with a complexity of $O(n^3)$.

In general, the system decides to interrupt a task $t(t_{start}, t_{end}, T, P)$ when the current time $u > t_{end}$ or if the plan is no more valid. If the task is controllable and the plan still remains valid, the decision will be made at:

early time: $u_{early} \geq t^{lb}_{end}$ if the task is an early preemptive one.

late time: $u_{late} = t^{ub}_{end} - \mu$ if the task is a late preemptive one.

The executive has to check the validity of the task reports considering the current plan. If the task has modified the level of resources, the executive checks for possible future conflicts. If the report is not nominal, it is considered as a task failure report.

Unexpected Events During Execution of a Plan

The system may receive messages of new goals to satisfy, changes of resource capacity or task failures.

IxTeT integrates these messages into its current plan. During this insertion, it partially invalidates the current plan by removing some causal links and some temporal constraints. If the new plan is not valid, execution is immediately aborted. Otherwise it always tries to make local plan repair in order to restore the plan correctness.

The plan repair search is similar to the plan search, but it is interrupted by a maximum time statically defined before execution.

IxTeT interleaves the plan repair process with perception of new events and new decisions. During this, the system has to always keep a valid plan for integrating new events or execute something at each interruption of the plan repair.

The plan repair process can fails in two cases. The first is when there is no correct plan in all the search space. The second case is when a time failure occurs before finding a new plan. These time failures correspond to the impossibility to instantiate a temporal variable corresponding to the beginning of an action, or a goal. These failures occur when the current plan repair process does not yet find a plan supporting the task. If the plan repair process fails, all the running actions are interrupted and a complete replanning is done.

When a complete replanning is needed, the system builds a new initial plan containing the current state and the remaining predicted evolution of contingent state variables. The remaining goals are kept.

During the search, at each step of the planning process IxTeT verifies that enough time remains for executing the plan. This is done by adding to the plan a temporal variable corresponding to the end of the replanning. All new inserted tasks are constrained to occur after this temporal variable. If the upper bound of the special variable is lower than the actual time minus a constant time needed to initialize the executive, the search is stopped.

When a replanning fails, the system abandons one goal. The goals are ranked according to a fixed priority value. The goal or one of the goals with the fewer priority is suppressed. A new replanning attempt is made until a solution plan is found. The abandonned goals are reinserted before any new complete replanning, and if their temporal constraints are still valid. A goal is retried only one time.

The Experimental System

The Simulated Architecture

IxTeT runs on the robot Dala (see Fig.3) and on a simulator of this robot. The simulator allows us to perform accurate tests of the different IxTeT strategies presented in the paper. The environment and the initial conditions are exactly the same.

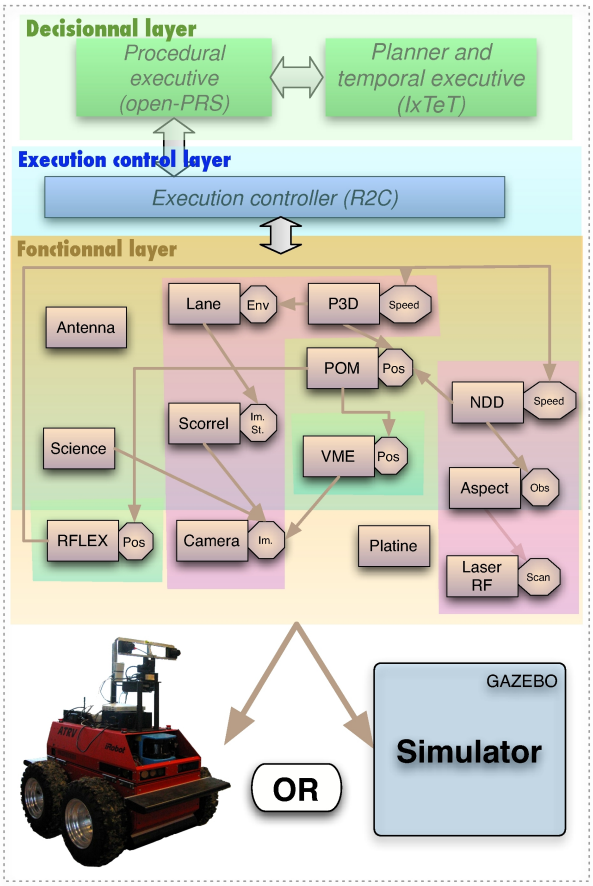

Figure 3: The LAAS architecture used for controlling the DALA mobile robot.

We use the LAAS architecture (Alami *et al.* 1998) (see Fig.3). The basic functions of the robot are encapsulated in software modules generated with GenoM. Normally, these basic modules are then run on board a robot. The simulator is described in (Joyeux *et al.* 2005).

In the top of the simulated functional layer, there is OpenPRS (Ingrand *et al.* 1996). It is used for actions monitoring and refining of the high level tasks into low level commands.

Evaluation Scenario

The main goal of the scenario is to evaluate the new modifications of IxTeT. A typical mission is defined as part of the evaluation scenario. The mission is then instantiated with different parameters to test the stability of the results.

Typical Mission This is an exploration rover like mission. The robot must acquire scientific data from several places. During its mission, it has to communicate with an orbiter during visibility windows. It has to be at a specific location at the end of the mission (for example back at its starting point).

For the need of the demonstration, we add a constraint on the motion of the cameras[10] (i.e. the MOVE_PAN_TILT_UNIT task, see Fig.7) to heat the motors before using them. The power constraints are such that heating is compatible with motion of the robot.

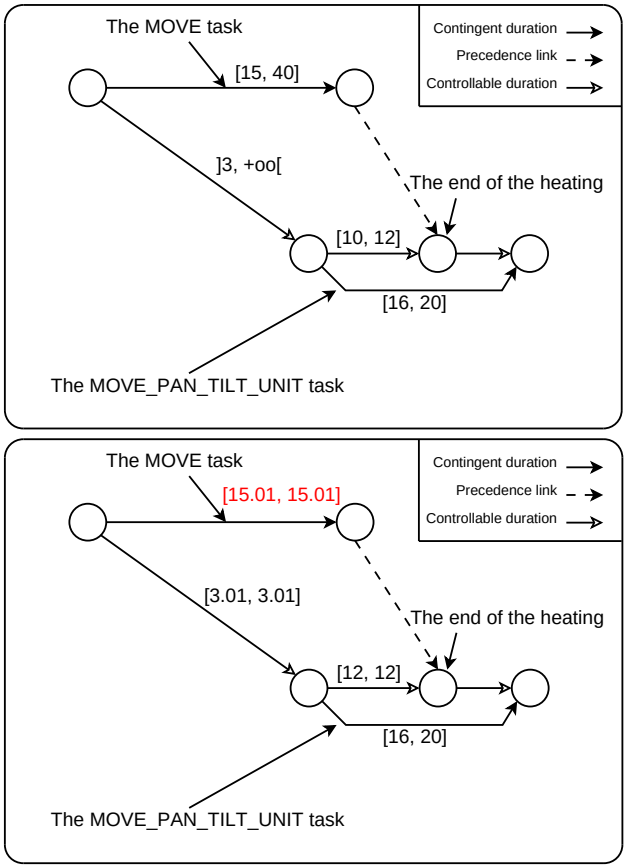

Figure 4: STN resulting of the insertion of a MOVE and a MOVE_PAN_TILT_UNIT tasks in a plan. The top one is before execution. The bottom one shows how the possible duration of the MOVE can be squeezed without using an STNU and 3DC+.

In the contingent plan model, the heating is allowed to be during a ground navigation. For the controllable model, the heating is made incompatible with the robot motions. This unnecessary constraint is added because the planner starts task as soon as possible. By doing this, it will start the "move cameras" task at a date whose propagation will squeezed the possible durations of the robot move (see Fig.4). This may induces an execution failure. The use of 3DC+ removes the need for the new constraint by adding a "wait".

[10]On the robot, the cameras are mounted on a pan and tilt unit.

Different Configurations Tested

IxTeT now features two different planning heuristics and two different time management systems. This defines four IxTeT instantiations.

Four different worlds have been defined. The first one has no obstacle. The second and the third ones each contain three obstacles. The last world contains all the obstacles in worlds two and three.

We have tested some goals configurations and some temporal window specifications. The values have been chosen randomly.

Results

The Makespan:

The new heuristic produces shorter initial plans with 3DC+ or without. During execution, the planner may need to repair several times the plan leading to a plan containing unnecessary tasks and really not optimal.

The stability of the heuristic is not as good as possible. The IxTeT planner does not have any geometric reasonning capacity. The plans produced are never optimal in the length of the planned path for the rover.

There is also the possibility that the ordering between navigations and communications is bad in sense of global makespan. The usage of the new heuristic for optimisation involved that if only one choice is not the good, it will never changed in the path to a solution. So, some plans are bad because the path of the robot is not the good one.

The usage of 3DC+ makes plans longer than with an STN, but this point is also relative to robustness as discussed in the next part.

The Robustness: The usage of STNU and dynamic controllability is a real improvement. In our tests (approximatively 250 runs), all goals were achieved except in one run with 3DC+. In runs without 3DC+, the system achieves 50% of the goals.

One another big advantage is that plans are saffer. It is possible that trivially unexecutable plans are produced when not using 3DC+. The effect is that some unachievable goals are not canceled when needed. It leads to failures to execute the higher priority goals instead of lower ones.

The usage of the new heuristic is catastrophic when used without 3DC+. It comes from the fact that task durations will be over squeezed and leads to near systematic plan failures.

The new heuristic does not give bad results when used with 3DC+. This is due to uncertainties that makes a less compressed schedule than with an STN. From the point of view of robustness, it is a good point.

Planning Duration and Other Time Measures: The planning time is really similar on little problem ($\leq$ 5 scientific goals). There is no significant difference using one heuristic or the other. In our tests, the problems are always easy due to needed flexibility (in order to have time for new goals). So this heuristic is not tested

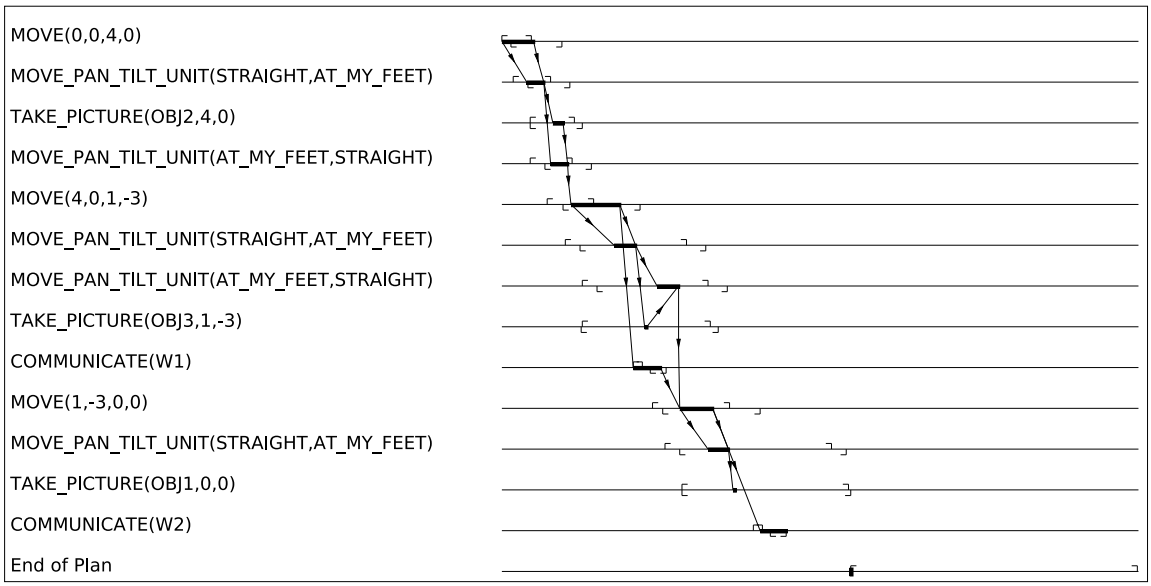

Figure 5: Typical initial plan of the robot during our experiment using an STNU as time formalism and using the makespan minimizing heuristic.

on hard problem. Yet, the usage of a complex world representation (numeric domains and actions effect depending of the duration of the task) makes harder to prove that a planning problem has no solution even if we use the old one.

The use of 3DC+ implies a higher temporal complexity at each planning and execution step. But contrary to the worst case, with a good implementation of the algorithm, the added cost is acceptable. During execution, the instantiation of a task start or stop takes 0.001 s with an STN and no more than 0.01 s with an STNU. The duration of plan repair steps are about 3 times higher with an STNU. This results shows that the cost of 3DC+ is compatible with its usage on an autonomous robot.

Prospectives and Current Work

We identify a drawback of the current plan repair process of IxTeT during our test. Sometimes, a repaired plan contains unnecessary tasks or not optimal tasks. For example, during our tests, we add new picture goals. The planner produces a plan resulting in two-way navigation from a new goal location and an existing waypoint. This may lead to a very low quality plan.

This situation arises when a new unsatisfied goal is added directly or during plan repair. This unsatisfied goal must not yet be an establisher of any temporal assertion in the plan. This goal must be on a state variables which has to take at least three different values (e.g. the position of the robot).

During the plan repair process, the planner establishes the new goal. If this goal is satisfied after any temporal assertion in the original plan, all is good. In the other case, the planner must insert tasks to reestablished the precedent values of the state variables. These tasks may be unnecessary, or their insertions impossible. The fig.8 illustrates the problem with a navigation.

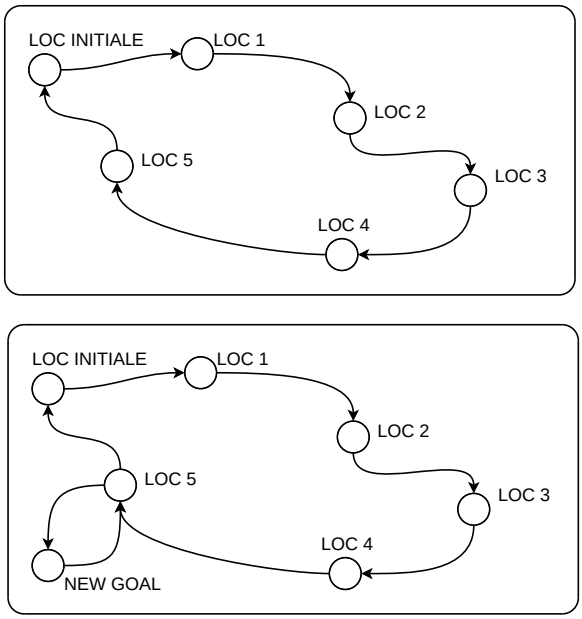

Figure 8: Initial plan (at top) and repaired plan after a new goal inserted (at bottom).

A Basic Solution

We describe this work using an example run. It is a mission with initially 5 picture goals and 2 communication goals. The initial plan is found in 1.7 s. It contains 22 tasks, 90

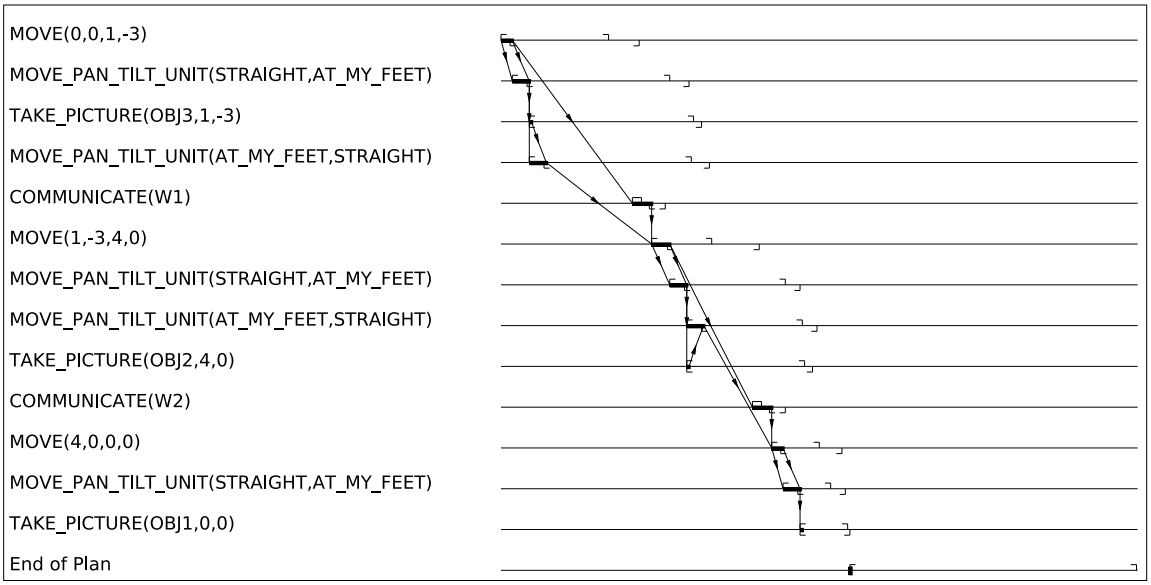

Figure 6: Typical initial plan of the robot during our experiment using an STNU and the commitment minimizing heuristic.

```
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;
  contingent ?duration = et − st;
}latePreemptive
```

```
task MOVE_PTU(?initL,?endL)(st,et){
  timepoint end_heat;
  ?initL,?endL in PTU_POSITIONS;
  hold(ROBOT_STATUS():STILL,(end_heat, st));
  event(PTU_STATUS():(COLD, HEAT),st);
  hold(PTU_STATUS():HEAT,(st,end_heat));
  event(PTU_STATUS():(HEAT,MOVING),end_heat);
  hold(PTU_STATUS():MOVING,(end_heat,et));
  event(PTU_STATUS():(MOVING,COLD),et);
  hold(PTU_INIT():TRUE,(st,et));
  hold(PTU_POS():?initL,(st,end_heat));
  event(PTU_POS():(?initL,PTU_POS_IDLE),end_heat);
  hold(PTU_POS():PTU_POS_IDLE,(end_heat,et));
  event(PTU_POS():(PTU_POS_IDLE,?endL),et);
  (end_heat − st) in [10,12];
  contingent (et − st) in [16,20];
}latePreemptive
```

Figure 7: An example of move and camera orientation (the cameras are mounted on a pan&tilt unit).

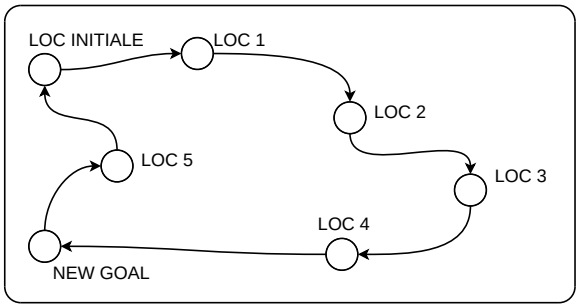

Figure 9: Expected plan after a plan repair

atemporal variables and 75 temporal variables. One picture goal is added during the first communication.

The problem illustrated in the fig.8 comes from a limited relaxation of the plan before the plan repair process. The solution described in the precedent sections of this paper, removes only causal links. A planner like IxTeT, adds constraints on variables to make causal links valid. If these constraints remains, the plan repair produces a suboptimal solution. The solution is to remove the constraints at the same time than the link.

We integrated the algorithms described in (Surynek & Barták 2004; Barták & Surynek 2005). We adapt the first one to continuous domains and use it to manage the filtering in the atemporal CSP. This permits to remove the atemporal constraints supporting a causal link.

The temporal constraints are not removed because the result is not satisfying. The main reason is that the planner introduces many more temporal constraints for solving conflicts between two temporal assertions than for justifying causal links. Relaxing only the constraints associated with causal links does not significantly relax the plan contrary to atemporal constraints.

We made another modification to the precedent algorithm. The executive selects all the causal links that may prevent the planner from finding a solution. It removes from this set the causal links that were added to justify a task. Precisely, during planning, the planner adds tasks to justify unestablished temporal assertions. The new task is then linked to the temporal assertion by a causal link. The final set is about 41 causal links in our example run.

We have try our solution using the simulator and the repaired plan is similar to the one illustrated on fig.9. The time needed for remove constraints is negligible: 0.02 s for 21 constraints. The planner finds a solution containing only necessary tasks or navigations. The duration of plan repair is approximatively 1.2 s for a plan with five picture goals plus

a new one. The needed relaxation of the mixed constraints[11] takes about 1.5 s. This is longer than a complete replanning but keep a plan more stable and allows to execute valid part of the plan.

This solution is yet limited to simple cases where actions partial order allows the planner to find a new solution. In the next part we discuss a solution to this problem.

A Future Work

Actually, we store the constraints associated to causal links. We are able to produce better plans using plan repair. We lack the capacity to change the order between tasks. In a typical mission, communications are made during fixed temporal windows while navigations are free. If we want the plan repair process to be complete, we must be able to move navigations around communications.

This may be done by storing which constraints have been inserted to solve conflict between temporal assertions. The information is associated with each involved assertions. If a plan repair process is needed, the executive removes only the constraints associated to temporal assertions of not yet executed tasks.

Conclusion

We have describe a temporal planner and executive whose plan execution raises new issues.

The first one is to deal with uncontrollable durations. We use a temporal framework with explicit uncertainties.

The second one is the bad quality of the plans when compared with a time optimal plan. We modify the search control of the planner to find better plans by modifying the planning heuristic.

A simulation architecture is used to evaluate the two solutions. An evaluation scenario is described and used to acquire results.

During the test, the heuristic has shown a good robustness. Yet, an identified drawback limits the performance of this work. A solution using the plan repair ability of IxTeT is briefly described in the last part of the paper.

The integration of an STNU shows that it is usable on a rover, contrary to the worst time complexity of $O(n^5)$ (Morris & Muscettola 2005)[12], with n the number of temporal variables. It shows a better robustness of the execution of the mission. If one goal is achievable, with 3DC+, it is executed.

But clearly, 3DC+ algorithm lacks the capacity to handle dynamic STNU. Some work have been made in this way (Stedl & Williams 2005). Yet to our knowledge, more work is needed to handle constraint removals.

References

Ai-Chang, M.; Bresina, J.; Charest, L.; Jónsson, A.; Hsu, J.; Kanefsky, B.; Maldague, P.; Morris, P.; Rajan, K.; and Yglesias, J. 2003. Mapgen: Mixed initiative planning and scheduling for the mars 03 mer mission. In *Proceedings of iSAIRAS*.

Alami, R.; Chatila, R.; Fleury, S.; Ghallab, M.; and Ingrand, F. 1998. An architecture for autonomy. *International Journal of Robotics Research, Special Issue on Integrated Architectures for Robot Control and Programming* 17(4):315–337.

Barták, R., and Surynek, P. 2005. An Improved Algorithm for Maintaining Arc Consistency in Dynamic Constraint Satisfaction Problems. In *International Florida AI Research Society Conference*.

Chien, S.; Tran, D.; Rabideau, G.; Cichy, B.; Davies, A.; Sherwood, R.; Castano, R.; Mandl, D.; Frye, S.; Trout, B.; D'Agostino, J.; Shulman, S.; and Boyer, D. 2005. The autonomous sciencecraft on earth observing one. In *i-SAIRAS-2005*.

Dechter, R.; Meiri, I.; and Pearl, J. 1991. Temporal Constraint Network. *Artificial Intelligence* 49(1-3):61–95.

Garcia, F., and Laborie, P. 1995. Hierarchisation of the search space in temporal planning. In *EWP*.

Ghallab, M., and Laruelle, H. 1994. Representation and Control in Ixtet, a Temporal Planner. In *AIPS*, 61–67.

Ghallab, M., and Mounir-Alaoui, A. 1989. Managing Efficiently Temporal Relations Through Indexed Spanning Trees. In *IJCAI*.

Ghallab, M.; Nau, D.; and Traverso, P. 2004. *Automated Planning : Theory and Practice*. Morgan Kaufmann.

Ingrand, F.; Chatila, R.; Alami, R.; and Robert, F. 1996. PRS: A High Level Supervision and Control Language for Autonomous Mobile Robots. In *IEEE International Conference on Robotics and Automation*.

Jonsson, A. K.; Morris, P. H.; Muscettola, N.; Rajan, K.; and Smith, B. D. 2000. Planning in Interplanetary Space: Theory and Practice. In *Artificial Intelligence Planning Systems*, 177–186.

Joyeux, S.; Lampe, A.; Alami, R.; and Lacroix, S. 2005. Simulation in the LAAS Architecture. In *ICRA Workshop on Interoperable and Reusable Systems in Robotics*.

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

Lemai, S., and Ingrand, F. 2004. Interleaving temporal planning and execution in robotics domains. In *AAAI*.

Lemai, S. 2004. *IxTeT-eXeC : planning, plan repair and execution control with time and resource management*. Ph.D. Dissertation, LAAS-CNRS and Institut National Polytechnique de Toulouse, France.

Mackworth, A. K., and Freuder, E. C. 1985. The Complexity of Some Polynomial Newtork Consistency Algorithms for Constraint Satisfaction Problems. *Artificial Intelligence* 25(1):65–74.

Mackworth, A. 1977. Consistency in networks of relations. *Artificial Intelligence* 8:99–118.

Morris, P. H., and Muscettola, N. 2005. Temporal Dynamic Controllability Revisited. In *AAAI*.

[11] A mixed constraint is relaxed by removing both the temporal constraint and the atemporal one. This leads to a complete propagation of the STN.

[12] It is a slightly modified version of 3DC+ with a cutoff mechanism.

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

Muscettola, N.; Morris, P.; and Tsamardinos, I. 1998. Reformulating temporal plans for efficient execution. In *Principles of Knowledge Representation and Reasoning*.

Rajan, K. 2004. Invited talk: Mapgen. In *IWPSS 2004, 4th International Workshop on Planning and Scheduling for Space, June 23 - 25*.

Stedl, J., and Williams, B. 2005. A fast incremental dynamic controllability algorithm. In *ICAPS Workshop on Plan Execution: A Reality Check*.

Surynek, P., and Barták, S. 2004. A New Algorithm for Maintaining Arc Consistency After Constraint Retraction. In *Principles and Practice of Constraint Programming*.

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

Vidal, T., and Fargier, H. 1999. Handling contingency in temporal constraint networks: from consistency to controllabilities. *JETAI* 11(1):23–45.