TLP-GP : Solving Temporally-Expressive Planning Problems

F. Maris P. Régnier

IRIT, Université Paul Sabatier,
118 route de Narbonne
31062 Toulouse, cedex 9, France.
{maris, regnier}@irit.fr

Abstract

This article describes an algorithm which solves temporally-expressive planning problems, that is problems for which all possible solutions require concurrency of actions. The planner TLP-GP which implements this algorithm constructs a simplified planning graph until the goals are attained, as in classic atemporal planners. It then establishes temporal constraints between actions and searches backward for a solution-plan in the planning graph using a disjunctive temporal constraint solver. If the search fails, the graph is extended to the next level and the search is restarted. This method can solve problems in a language whose expressivity is greater than PDDL 2.1. Preconditions can be required and effects can take place on any temporal interval relative to the start-time of an action. This algorithm can also take into account, in a very natural way, exogenous events as well as temporally extended goals. We also propose several different means of extending expressivity even further. TLP-GP is complete for the temporally-expressive sublanguages of PDDL 2.1.
We compared our planner with two state-of-the-art temporally-expressive planners such as LPGP and VHPOP. These experimental trials not only show the efficiency of our approach but also demonstrate the practical possibility of solving temporally expressive problems which up until now were unsolvable by existing techniques.

1. Introduction

One of the major challenges to be met in order to solve real-world planning problems is to take into account the time dimension. Indeed, many real-world problems require concurrent actions to be solved. Airport or station management, baking ceramics [Cushing et al., 2007.b] and cooking are just some examples.

Although many temporal planners have been built and compared in IPC competitions, several problems remain which prevent them from being used effectively to solve real applications. These planners are primarily based on one of three types of algorithms: search in extended state spaces, search in partial-plan spaces, and extensions of GRAPHPLAN [Blum, Furst, 1995].

Planning in extended state-space: this is currently the technique which produces the best results, but the expressivity of the languages remains weak. Planners of this type use the state of the world as the central attribute in a temporal state, which allows them to use the well-tried techniques of heuristic search employed in atemporal planning. They are very efficient, which is demonstrated by the fact that SGPLAN [Chen, Wah, Hsu, 2006] won the IPC 2004 and 2006 competitions, in the temporal planner category. Most of these systems restrict the possible starting times of an action to specific instants called decision-epoch, which makes them incomplete for many problems requiring simultaneous actions [Cushing et al., 2007.a]. They are only complete for certain sublanguages of PDDL2.1 [Fox, Long, 2003] which can eventually be reduced to STRIPS [Cushing et al., 2007.a]. Therefore, they can only solve those problems for which there is a sequential solution (temporally simple problems). However, almost all real-world problems, even if they can be solved by essentially sequential plans, require concurrent actions at some moment or other. In order to be able to envisage real applications, one must represent and solve problems for which all possible solutions require parallelism (temporally expressive problems) [Cushing et al., 2007.b].

Plan-space planning: This approach has also been extended to the temporal framework. The first HTN (Hierarchical Tasks Network) planners to introduce a temporal aspect to problems (durations and activation windows for actions) were DEVISER [Vere, 1983] and FORBIN [Miller, 1985] which used two specific modules for the management of temporal constraints

and the optimization of global duration. More recently, the HTN planner IxTeT [Ghallab, Alaoui, 1989], [Ghallab, Laruelle, 1994], [Laborie, Ghallab, 1995], used specific procedures to manage almost linearly the addition of temporal constraints in a lattice of time-points. IxTeT language is very expressive but no automatic translator between PDDL2.1 and the IxTeT input language exists so it is necessary to translate the problems into its own representation language. Moreover, the plan representation of IxTeT is very general and several plans produced by IxTeT could not be validated. Nevertheless, results[1] of the IPC'02 seems to demonstrate that IxTeT is less efficient than VHPOP [Younes, Simmons, 2003].

Classical partial-order planners (POP) have also been extended to the temporal framework. The planner UCPOP [Penberthy, Weld, 1992] thus gave rise to the planner VHPOP [Younes, Simmons, 2003]. In order to achieve this extension, temporal intervals are generally used to represent actions and propositions, and the causality relation between actions is replaced by a temporal order within partial plans. Conflict management is then achieved by a constraint satisfaction system. These POP-type techniques are particularly interesting when searching for optimal plans; the planner CPT2 is among the temporal planners which obtained the best results, notably at the IPC'06 planning competition in the optimal temporal planning category. Unfortunately, the chosen representation of actions restricts it use to temporally simple languages.

Extensions of GRAPHPLAN: the use of planning graphs [Blum, Furst, 1995] has also been extended to temporal problems by several planners. The system TGP [Smith, Weld, 1999] which uses a temporally simple language, was the first such planner. In the algorithms based on the TGP representation, the construction of the graph is entirely guided by a discrete time-line : at each step, the planner considers the next instant when a change can occur. It follows that each action starts immediately after the start or the end of another action. TPSYS [Garrido, Fox, Long, 2002] and CPPLANNER [Dinh, Smith, 2003] are other planners which provide a temporally expressive language. Unfortunately, the start-times of actions remain linked to fixed decision-epoch, which renders these algorithms incomplete for temporally expressive domains [Cushing et al., 2007.a]. TGP, TPSYS and CPPLANNER are nevertheless complete for the temporally simple sublanguages of PDDL2.1. Other systems, such as LPG-TD [Gerevini, Saetti, Serina, 2006] (or the planners SGPLAN [Chen, Wah, Hsu, 2006] and MIPS [Edelkamp, Helmert, 2001]) only consider sequential plans which they optimize using temporal information in order to render them concurrent. LPGP [Long, Fox, 2003] is one of the most expressive among recent planners. Unlike previous systems, it is complete for temporally expressive languages but for certain problems, a large number of levels need to be built, which makes them more difficult to solve.

Recent theoretical studies have brought to light the limitations of the current approaches to temporal planning. [Cushing et al., 2007.b] show that the domains and problems which have been used up until now in the IPC competitions are temporally simple and they propose a method to prove that a domain is temporally expressive. The winning planners in the IPC competitions, even if they are efficient in a restricted temporal framework, are therefore far from capable of solving real-world problems. The objective evaluation of these systems requires the setting up of new benchmarks corresponding to temporally expressive problems. [Rintanen, 2007] prove that solving such problems is EXP-SPACE complete.

In this article, we present an algorithm, called TLP-GP (Temporally Lifted Progression GraphPlan), based on the use of a simplified planning graph and a Disjunctive Temporal Problem (DTP) solver. Our planner can solve problems expressed in a language whose temporal expressivity is greater than that of PDDL2.1 since preconditions can be required and effects can occur on any temporal interval relative to the start-time of an action. TLP-GP can also take into account, in a natural way, exogenous events and temporally extended goals which must be true over a certain period of time. It is complete for the temporally expressive sublanguages of PDDL2.1. The article is organized as follows. In Section 2, we define and compare the expressivity of different languages used in classical temporal planners. In Section 3, after giving the representation of actions in TLP-GP, we describe, by means of an example, the expansion of the planning graph and then the extraction of a floating solution-plan. In Section 4, we present experimental results obtained on several new temporally expressive benchmarks. Finally we conclude by explaining how we can improve the expressivity of our planner's language and the efficiency of our algorithm.

2. Temporal planning languages expressivity

We use the notation of [Cushing et al., 2007.a]. $L^{<\text{preconditions}>}_{<\text{effects}>}$ denotes the expressivity of a temporal planning language, where <preconditions> and <effects> represent, respectively, the times when the

[1] http://www.cs.cmu.edu/afs/cs/project/jair/pub/volume20/long03a-html/node5.html

preconditions must hold and the times when the effects can occur. The values taken by <preconditions> and <effects> can be :

- s: start-time of an action;
- e: end-time of an action;
- o : over all the duration of the execution of an action.

This notation is completed by an extension to intervals. Thus [s,e] represents the fact that the preconditions (respectively, the effects) can be required (can occur) over any open, closed or mixed interval between the start-time and the end-time of the action. This notation can be further extended by letting [s+, e+] represent the fact that this (open, closed or mixed) interval is no longer restricted to lie within the start and end times of the action.

According to this notation, our planner TLP-GP uses a (temporally expressive) $L^{[s+,e+]}_{[s+,e+]}$ language. PDDL2.1 is a temporally expressive $L^{s,o,e}_{s,e}$ language. TGP uses a (temporally simple) L^{o}_{e} language, where each precondition must hold for the entire duration of the execution of the action and each effect must occur at the end of the action. TPSYS uses a $L^{o}_{s,e}$ language where each precondition must hold for the entire duration of the execution of the action and effects can occur at the start or the end of the action. CPPLANNER uses a $L^{o}_{[s,e]}$ language, in which effects can be produced at any point between the start and end times of the action. LPG-TD uses a $L^{s,o,e}_{s,e}$ language but this planner is incomplete for temporally expressive languages. LPGP and VHPOP uses a $L^{s,o,e}_{s,e}$ language and are complete for temporally expressive PDDL2.1 problems. We'll use these two planners for our experimental studies.

3. The TLP-GP algorithm

TLP-GP uses a similar method to that used by the family of BLACKBOX [Kautz, Selman, 1999] and GP-CSP [Do, Kambhampati, 2001] planners. The planning graph is built until the goal are obtained, following the classic algorithm for the atemporal case, without the calculation of mutexes. TLP-GP then looks for a solution-plan searching backwards in the planning graph, using a Disjunctive Temporal Problem (DTP) solver. To achieve this, it places temporal constraints between actions and uses an agenda which is different to that used by TGP.

Instead of a placing actions and propositions on a discrete time-line, we use real-valued temporal variables on which constraints are defined. At each back-chaining step, this set of constraints is fed into a solver which verifies global consistency. This operation is repeated until a solution is obtained. In case of failure, another level is added to the planning graph and the back-chaining process is restarted.

3.1. Representation of actions

Actions are represented by 4-tuples (<action-name>, <preconditions>, <effects>, <duration>). <effects> and <preconditions> are sets of propositions each with a corresponding temporal label. This label represents an interval, defined relative to the start-time of the action, during which a precondition must be verified or an effect is produced. A label [t,t] containing a single value t is written simply as [t]. <duration> represents the duration of the action. We use $\tau_s(A)$ to denote the temporal variable corresponding to the start-time of the action A. We introduce two imaginary and instantaneous actions A_I and A_G which correspond, respectively, to the initial state and the achievement of the goal. The start and end times of a plan are given by $\tau_I = \tau_s(A_I)$ et $\tau_G = \tau_s(A_G)$.

Exemple : consider the action (A, $\{a_{[-1,\ 2]}, b_{[0]}\}$, $\{\neg a_{[3]}, c_{[5,7]}\}$, 5). If $\tau_s(A)$ is the start-time of A, its duration is 5, the proposition a must hold between $\tau_s(A)$-1 and $\tau_s(A)$+2, b must hold at instant $\tau_s(A)$, $\neg$a appears at $\tau_s(A)$+3 and c appears at $\tau_s(A)$+5 and remains true at least until $\tau_s(A)$+7.

In the following, we will use the example of [Cushing et al., 2007.a, figure 3] to explain the algorithm TLP-GP. This problem is interesting because it cannot be solved by any planner based on either of the following assumptions (in particular decision epoch planners):

- Every action will start immediately after some other action has started or ended.
- The only conflicts preventing an earlier dispatch of an action, however indirect, involve actions which start earlier.

In this planning problem $\prod$ = <O, I, G>, the initial state is I = {}, the goal is G = {b, d, e} and the set of actions is O = {(A, {}, $\{a_{[0]}, \neg a_{[5]}, b_{[5]}, \neg d_{[5]}\}$, 5) ; (B, $\{a_{[0]}\}$, $\{c_{[0]}, d_{[4]}, \neg c_{[4]}\}$, 4) ; (C, $\{c_{[0]}\}$, $\{\neg b_{[1]}, e_{[1]}\}$, 1)}.

3.2. Expansion of the temporal planning graph

Unlike the planners TGP or LPGP, TLP-GP uses a planning graph without mutexes and constructed in an atemporal manner, without taking into account, to start with, either the duration or the start-time of actions. Conflicts between actions, including mutual exclusions, are managed entirely during the solution extraction phase by a constraint satisfaction system. This minimal usage of the planning graph means that TLP-GP does not have the same restrictions as other planners:

decision epochs or time-line which limits the completeness to temporally simple problems or a number of levels which is too large to allow the practical resolution of temporally expressive problems (LPGP). The method we have chosen consists in entrusting a large part of the work to a DTP solver. This method has the double advantage of producing floating temporal plans and considerably increasing the expressivity of the representation language that can be used. This strategy turned out to be very effective in the context of classical planning, thanks to the considerable progress in SAT solvers. The SATPLAN'04 system, the successor to BLACKBOX [Kautz, Selman, 99], was still (seven years later) the winner in the optimal-planner category of the IPC'06 competition.

Since an action can produce and destroy the same proposition at different times, we also store in the graph the negations of propositions. Finally, unlike other temporal planners based on GRAPHPLAN, the levels are not linked to a fixed time scale. Level 0 contains the dummy action A_I, which has no preconditions and produces all the propositions of the initial state, together with the corresponding effect arcs. In our running example, we omit level 0 since I is empty. For each level $n \geq 1$, we apply all those actions whose preconditions are all present at level n-1 and we add the corresponding precondition arcs to the graph. We then add the effects of these actions at level n together with the corresponding effect arcs. The graph is built in this way level by level until all the goals appear. Finally, we build an extra level containing the dummy action A_G, which has no effects and has all the goals as its preconditions, and we add the corresponding precondition arcs. Fig. 1 shows the graph corresponding to our running example.

The extraction algorithm is then called. Each arc in the planning graph has a corresponding temporal label according to its type:

- **Precondition arc** (proposition → action): the label represents the interval, relative to the start of the action, over which the precondition must be verified.
- **Effect arc** (action → proposition): the label represents an interval, relative to the start of the action, at the start of which the effect appears and during which it remains true. After the end of this interval, there is no guarantee that it still holds.

LPLG is the only planning-graph-based planner which can solve this problem. It achieves this by choosing decision times arbitrarily far in the future and in the past. However, even though all three actions of the plan are present from level 4 onwards, it is still necessary to extend the graph to level 6 to obtain a solution, whereas TLP-GP finds a solution to this same problem at level 3, without backtracking in the graph nor extending it.

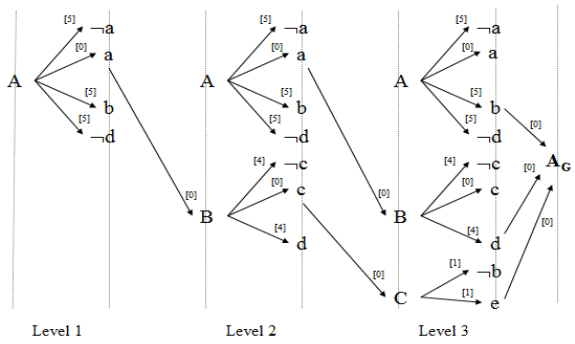

Figure 1: Expansion of the temporal planning graph

3.3. Extraction of a floating solution-plan

Once the planning graph has been extended to a level at which all goals are present, TLP-GP searches backward for a solution-plan. To this end, it places temporal constraints between actions and, instead of assigning actions and propositions to fixed time points, it uses mutually constraining real-valued temporal variables. At each step, the set of constraints is fed to a disjunctive temporal constraint satisfaction system which checks its satisfiability. This operation is repeated until a solution is found. In case of failure, an extra level is added to the planning graph and the back-chaining procedure restarted. The simplified extraction algorithm is given below. The search for a solution requires two data structures: *Agenda* and *Constraints*.

Agenda is a set of lists, one for each proposition. For a proposition p, the corresponding list Agenda(p) is composed of intervals of the form $[\tau_s(A)+\delta_1, \tau_s(B)+\delta_2]$, over which p must be true. When the interval consists of a single value, we denote it by $[\tau_s(A)+\delta]$. Two types of temporal intervals can be added to Agenda(p):

- Intervals which correspond to a causality relationship between actions (if the proposition p is produced by an action A in order to fulfill a precondition of an action B, p must remain true until used by B).
- Intervals which correspond to the appearance of the effects of a selected action (appearance of p).

Constraints is a list of disjunctions of (at most two) binary temporal constraints between time points:

- Constraints corresponding to causality relationships representing the fact that a precondition must be produced (by an action A at time $\tau_s(A)+\delta_1$) before it is required (by an action B at time $\tau_s(B)+\delta_2$). Such constraints are not, in fact, disjunctions since they are of the form $\tau_s(A)+\delta_1 \leq \tau_s(B)+\delta_2$.

- Constraints imposing the non-intersection of the two intervals $[\tau_s(A)+\delta_1, \tau_s(B)+\delta_2]$ and $[\tau_s(C)+\delta_3, \tau_s(D)+\delta_4]$ over which a proposition and its negation hold. In the most general case, these constraints are disjunctive since they are of the form $(\tau_s(B)+\delta_2 \leq \tau_s(C)+\delta_3) \vee (\tau_s(D)+\delta_4 \leq \tau_s(A)+\delta_1)$. The inequalities may or may not be strict depending on the type of intervals under consideration (open, closed or mixed). In most cases, this constraint simplifies to a non-disjunctive constraint.

The set of constraints created by TLP-GP therefore constitutes a disjunctive temporal problem (DTP). The problem of solving or verifying the consistency of a DTP is NP-hard [Dechter, Mieri, Pearl, 1991] but the performance of the algorithms to solve these problems are regularly improved[2]. In our running example (cf. figure 2), the ordering heuristics used in the choice of subgoals to be established and the choice of actions to establish them is the following: priority is given to subgoals which appear (for the first time) in the highest levels of the graph and, among actions which establish them, priority is given to those actions which appear (for the first time) in the lowest levels of the graph.

Extraction algorithm

```
Goals ← Pre(A_G);
For every effect e ∈ Eff(A_I):
   Add an interval I to Agenda(e) for the
   apparition of the proposition e;
End For;
While Goals ≠ ∅
   For every proposition p ∈ Goals:
      Goals ← Goals - p;
      Select (* Backtrack point *), using the
      heuristic, an action A to produce p;
      Goals ← Goals ∪ Pre(A);
      Add a precedence constraint between A and
      B, the action whose p is a precondition;
      Add an interval I to Agenda(p) to maintain
      the precondition p;
      For every interval I in Agenda(¬p):
         Add a constraint to forbid the
         overlapping of I and I';
      End For;
      For every effect e of A, e ≠ p:
         Add an interval I to Agenda(e) for the
         apparition of e;
         For every interval I' in Agenda(¬e):
                Add a constraint to forbid the
                overlapping of I and I';
         End For;
      End For;
      Check the consistency of constraints (call
      to the DTP solver);
      In case of failure, return to the back-
      track point to select another action A;
   End For;
End While;
If constraints is satisfiable
   Then return the floating solution-plan
   (selected actions and constraints)
   Else there is no solution in this level;
End If;
End.
```

[2] http://www.smtcomp.org/

To initialize the search, all the goals (the preconditions of A_G) are added to the list Goals. The intervals corresponding to the appearance of the propositions of the initial state (the effects of action A_I) are then added to Agenda in order to optimize the search.

In order to establish the goal e (present at level 3), we choose action C (at level 3). C_3 is therefore selected. $[\tau_s(C_3)+1; \tau_G]$ is added to Agenda(e) in order to preserve the subgoal e until the instant τ_G. $[\tau_s(C_3)+1]$ is added to Agenda(¬b) to take into account the effect ¬b. We then choose action B at level 2 to establish c, the precondition of C_3. B_2 is selected and $[\tau_s(B_2); \tau_s(C_3)]$ is added to Agenda(c) in order to maintain the precondition c until it is required. $\tau_s(B_2) \leq \tau_s(C_3)$ is added to Constraints to make sure that this precondition is produced before it is used. $[\tau_s(B_2)+4]$ is added to Agenda(¬c) and to Agenda(d) to take into account the appearance of the effects ¬c and d. In order to prevent a conflict between c and ¬c (brought to light by Agenda(c) and Agenda(¬c)), we add the disjunctive constraint $(\tau_s(C_3)<\tau_s(B_2)+4) \vee (\tau_s(B_2)+4<\tau_s(B_2))$ which simplifies, in this case, to $\tau_s(C_3)<\tau_s(B_2)+4$. We then select action A_1 to establish a which is the only precondition of B_2. $[\tau_s(A_1); \tau_s(B_2)]$ is added to Agenda(a) so that a remains true until it is needed, and $\tau_s(A_1) \leq \tau_s(B_2)$ is added to Constraint. $[\tau_s(A_1)+5]$ is added to Agenda(¬a), Agenda(b) and Agenda(¬d) to take into account the appearance of the effect ¬a, b and ¬d. To avoid a conflict between a and ¬a, we add the disjunctive constraint $(\tau_s(B_2)<\tau_s(A_1)+5) \vee (\tau_s(A_1)>\tau_s(A_1)+5)$ which simplifies to $\tau_s(B_2)<\tau_s(A_1)+5$. To avoid a conflict between b and ¬b, we add the disjunctive constraint $(\tau_s(A_1)+5<\tau_s(C_3)+1) \vee (\tau_s(A_1)+5>\tau_s(C_3)+1)$ (which is equivalent in this particular example to the constraint $\tau_s(A_1)+5 \neq \tau_s(C_3)+1$). To avoid a conflict between d and ¬d, we add the disjunctive constraint $(\tau_s(A_1)+5<\tau_s(B_2)+4) \vee (\tau_s(A_1)+5>\tau_s(B_2)+4)$ (which is equivalent to $\tau_s(A_1)+5 \neq \tau_s(B_2)+4$). We now try to establish the goal d. According to Agenda(d), this proposition is already established at $\tau_s(B_2)+4$. $[\tau_s(B_2)+4; \tau_G]$ is therefore added to Agenda(d) to preserve the subgoal until the instant τ_G. To avoid a conflict between d and ¬d, we add the disjunctive constraint $(\tau_s(A_1)+5<\tau_s(B_2)+4) \vee (\tau_s(A_1)+5>\tau_G)$ which simplifies to $\tau_s(A_1)+5<\tau_s(B_2)+4$ Finally we establish the goal b. According to Agenda(b), this proposition is already established at $\tau_s(A_1)+5$. $[\tau_s(A_1)+5; \tau_G]$ is added to Agenda(b) to preserve b until the instant τ_G. To avoid a conflict between b and ¬b, we add the constraint $\tau_s(C_3)+1<\tau_s(A_1)+5$.

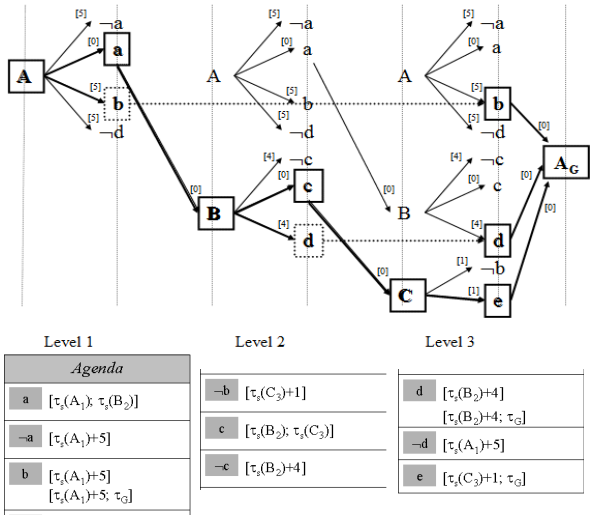

Figure 2: Extraction of a floating solution-plan

As was the case in all previous steps, the set of temporal constraints is satisfiable. This allows us to obtain the floating solution-plan (set of selected actions and constraints) given in Fig. 3. Time-points $\tau_s(x)+\delta$ defined relative to the start time $\tau_s(x)$ of an action x, which occur in the set of constraints are indicated by vertical dotted lines. The arrows indicate the maximum possible slide of the start and end times of each action. Square brackets are used to indicate whether time-points can coincide or whether they must be distinct.

In the example, the start of B_2 can slide forward up to the start of A_1 (bold arrow to the left of Fig. 3), but the end of A_1 must occur strictly before the end of B_2 (bold arrow to the right of Fig. 3). Respecting these constraints means that the goal d is eventually achieved by B_2 even though previously destroyed by A_1. This floating solution-plan is found by TLP-GP at level 3, without backtracking. It is very flexible and we can easily optimize its execution time by making all the actions start as soon as possible.

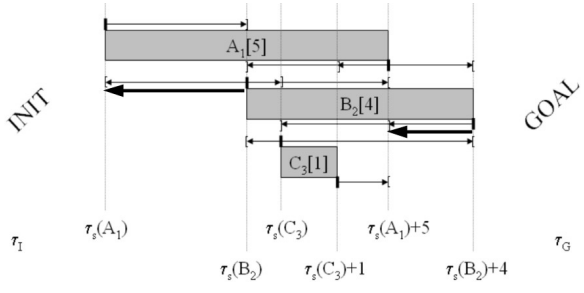

Figure 3: Floating solution-plan

4. Experimental trials

4.1. Generalities

TLP-GP is implemented in OCaml 3.09.2. The examples were tested on a 1.6 GHz Pentium M machine with a 512 Mbyte DDRAM. TLP-GP uses the SMT (Sat Modulo Theory) solver MathSat[3] 3.4 which is known to perform well on DTP (cf. results of the SMT-COMP'06[4] competition). When extracting a solution, the heuristic of TLP-GP gives priority to subgoals appearing (for the first time) in the highest levels of the graph and, to establish them, tries first the actions which appear (for the first time) in the lowest levels of the graph.

We compared TLP-GP with two state-of-the-art planners capable of solving temporally expressive problems: the LPGP planner which is an extension of GRAPHPLAN, and the partial-plan space planner VHPOP. Given that the IPC competition benchmarks were inappropriate, since temporally simple, we drew up several new temporally expressive benchmarks. All these benchmarks can be found at the adress which is given below[5]. The first test domains extend the problem of [Cushing et al., 2007.a, figure 3] in three different ways:

- the tempo-depth-n problems extend the original problem over a greater number of levels. The solution-plan requires n stages, each composed of triples of concurrent actions.
- the tempo-width-m problems require m triples of concurrent actions over only three levels of the graph in order to achieve the goal.
- the tempo-matrix-n×m problems combine the difficulties of the above two cases.

The last domain, tms-k-t-p (temporal machine shop, [Cushing et al., 2007.a]) is inspired by a real-world application. It concerns the use of *k kilns*, each with different baking times, to bake *p* ceramic *pieces* (*bake-ceramic*) of *t* different *types*. Each of these types requires a different baking time. These ceramics can then be assembled to produce different structures (*make-structure*). The resulting structures can then be baked again to obtain a bigger structure (*bake-structure*).

As the expressivity of LPGP and VHPOP is less than that of TLP-GP, we tested simplified, but nevertheless temporally expressive, versions of these benchmarks. In all our trials we limited the total search time to a maximum of one hour.

[3] http://mathsat.itc.it/
[4] http://www.smtcomp.org/2006/
[5] http://tlpgp.free.fr/

4.2. Results

Problem		TLP-GP		LPGP		VHPOP	
		Actions	Time	Actions	Time	Actions	Time
tempo-depth	02	6	0.46	18	1.00	6	0.16
	03	9	0.66	-	-	9	0.28
	04	12	0.96	-	-	12	0.33
	05	15	1.26	-	-	15	0.41
	06	18	1.58	-	-	18	0.49
	07	21	1.92	-	-	21	0.59
	08	24	2.36	-	-	24	0.70
	09	27	2.85	-	-	27	0.82
	10	30	3.28	-	-	30	1.39
	20	60	13.36	-	-	60	5.20
	30	90	28.89	-	-	90	21.29
	40	120	51.06	-	-	120	31.89
	50	150	75.45	-	-	150	58.05
	100	300	343.00	-	-	300	390.63
	150	450	1855.53	-	-	450	1336.43
	200	600	4658.95	-	-	*	*
tempo-width	02	6	0.47	18	0.02	6	18.83
	03	9	0.54	27	0.03	*	*
	04	12	0.77	36	0.09	*	*
	05	15	1.10	45	0.23	*	*
	06	18	1.55	54	1.80	*	*
	07	21	2.11	63	27.93	*	*
	08	24	2.85	72	538.70	*	*
	09	27	3.77	-	-	*	*
	10	30	3.92	-	-	*	*
	20	60	36.89	-	-	*	*
	30	90	140.77	-	-	*	*
	40	120	315.74	-	-	*	*
	50	150	576.45	-	-	*	*
	60	180	945.25	-	-	*	*
	70	210	1447.07	-	-	*	*
tempo-matrix	2x2	12	1.08	36	1.33	*	*
	3x3	27	3.99	-	-	*	*
	4x4	48	14.32	-	-	*	*
	5x5	75	46.57	-	-	*	*
	6x6	108	134.19	-	-	*	*
	7x7	147	358.47	-	-	*	*
	8x8	192	1058.75	-	-	*	*

- : cpu time ≥ 3600 s.
* : out of memory (512 Mbyte)

On the domain "tempo-depth", LPGP fails on the problem tempo-depth-03. VHPOP is more efficient than TLP-GP on most problems but needs more memory to solve the problem tempo-depth-200. TLP-GP solves all these problems. It is a little less efficient than VHPOP and produces a floating solution-plan for the first 40 problems in less than one minute. It still solves the problem tempo-depth-150 in less than one hour, with 1,197 simple constraints and 3,439 disjunctive constraints.

On the domain "tempo-width", LPGP successfully solves up to the problem tempo-width-08. VHPOP only solves the first problem and does not have enough memory to solve the following ones. TLP-GP solves all these problems. It gives a floating solution-plan for the 10 first problems in less than 5 s., for the 50 first problems in less than 10 minutes. The problem tempo-width-70 is the last problem solved by TLP-GP in less than one hour: it needs 30 mn. to be solved whereas the DTP contains 350 simple constraints and 34,857 disjunctive constraints.

On the domain "tempo-matrix", LPGP only solves the problem tempo-matrix-2×2 and, because of lack of memory, VHPOP does not solve any of these problems. TLP-GP solves all these problems up to tempo-matrix-8×8. It gives a floating solution-plan in less than 20 mn. whereas the DTP contains 488 simple constraints and 6,256 disjunctive constraints.

On the domain "tms-2-3", LPGP dœs not solve any problem. VHPOP solves the three first problems in less than 1 minute and fails to solve tms-2-3-06 because of lack of memory. TLP-GP solves 29 problems in less than 1 minute. The problem tms-2-3-80 is the last problem solved by TLP-GP ; it needs 30 mn. to be solved whereas the DTP contains 474 simple constraints and 6,873 disjunctive constraints. For this last problem, the floating solution-plan contains 277 actions.

Problem		TLP-GP		LPGP		VHPOP	
		Actions	Time	Actions	Time	Actions	Time
tms-2-3	03	9	0.74	-	-	8	0.06
	04	17	1.41	-	-	13	33.08
	05	17	1.47	-	-	14	37.67
	06	24	2.41	-	-	*	*
	07	25	2.63	-	-	*	*
	08	31	3.81	-	-	*	*
	09	31	4.02	-	-	*	*
	10	32	4.68	-	-	*	*
	20	73	24.07	-	-	*	*
	30	103	61.67	-	-	*	*
	40	143	167.52	-	-	*	*
	50	175	333.74	-	-	*	*
	60	215	629.22	-	-	*	*
	70	245	1047.03	-	-	*	*
	80	277	1820.51	-	-	*	*

5. Conclusion and future work

We have presented TLP-GP, a planner which can solve temporally expressive problems in a language whose expressivity is greater than PDDL2.1. No compromise needs to be made in terms of completeness in order to achieve this expressivity. On temporally expressive benchmarks, TLP-GP performed much better then two state-of-the-art planners (LPGP and VHPOP) capable of solving the same types of problem. These results indicate the possibility of representing and solving problems which are closer to real-world applications. The production of a floating solution-plan rather than a single fixed solution also allows for a greater flexibility during the execution of the plan.

In this article, because the expressivity of LPGP and VHPOP is less than that of TLP-GP, we have restricted our presentation to the simplest version of TLP-GP in which each precondition or effect p has an associated interval over which p must be true and not p false. This corresponds to "over all" preconditions in PDDL2.1, extended to effects by [Cushing et al., 2007.b]. However, to represent real-word domains, we have implemented a more expressive language in which we can represent the fact that a precondition or effect p must be true (and not p false) during a minimal duration d anywhere within an interval [a,b]. Our language also allows the user to disassociate p and not p by stipulating, for example, that p must be true at the end of an interval [a,b] over all of which not p cannot be established. It is also possible, in a natural way, to represent external events or goals which have a duration. The former can be easily coded as effects of the dummy action A_I and the latter as preconditions of the dummy action A_G. Benchmarks that use this language can be found at the adress which is given below[6].

We are at present working on improvements to TLP-GP by testing various more-informed heuristics to choose the order in which to assert goals and the order in which to try actions to establish them. We are also investigating the possibility of adding new constraints incrementally in a DTP solver instead of restarting the solver from scratch at each successive call.

References

[Blum, Furst, 1995] A.Blum, M.Furst, "Fast Planning Through Planning Graph Analysis", IJCAI, 1995.

[Chen, Wah, Hsu, 2006] Y.X.Chen, B.W.Wah, C.W.Hsu, "Temporal planning using subgoal partitioning and resolution in SGPlan", Journal of AI Research, 2006.

[Chen, Xing, Zhang, 2007] Y.X.Chen, Z.Xing, W.Zhang, "Long-Distance Mutual Exclusion for Propositional Planning", IJCAI, 2007.

[Cushing et al., 2007.a] W.Cushing, S.Kambhampati, Mausam, D.S.Weld, "When is temporal planning really temporal ?", IJCAI, 2007.

[Cushing et al., 2007.b] W.Cushing, S.Kambhampati, K.Talamadupula D.S.Weld, Mausam, "Evaluating temporal planning domains", ICAPS, 2007.

[Dechter, Mieri, Pearl, 1991] R.Dechter, I.Mieri, J.Pearl, "Temporal constraint networks", AI, 49: 61-95, 1991.

[Dinh, Smith, 2003] T.B.Dinh, B.M.Smith, "CPPlanner: A Temporal Planning System using Critical Paths", APES, 2003.

[Do, Kambhampati, 2001] M.B.Do, S.K Kambhampati, "Planning as Constraint Satisfaction: Solving the Planing Graph by compiling it into CSP", AI, 132, 2001.

[Edelkamp, Helmert, 2001] S.Edelkamp, M.Helmert, "The Model Checking Integrated Planning System", AI Magazine, 22(3), 2001.

[Fox, Long, 2003] M.Fox, D.Long, "PDDL2.1 : An Extension to PDDL for Expressing Temporal Planning Domains", Journal of AI Research, 20, 2003.

[Garrido, Fox, Long, 2002] A.Garrido, M.Fox; D.Long, "TPSYS: Temporal Planning with PDDL2.1", ECAI, 2002.

[Gerevini, Long, 2005] A.Gerevini, D.Long, "BNF Description of PDDL3.0", technical report, 2005.

[Gerevini, Serina, 2002] A.Gerevini, I.Serina, "LPG:A planner based on local search for planning graphs", AIPS, 2002.

[Gerevini, Saetti, Serina, 2006] A.Gerevini, A.Saetti, I.Serina, "An Approach to Temporal Planning and Scheduling in Domains with Predictable Exogenous Events", JAIR, 25, 2006.

[Ghallab, Alaoui, 1989] M.Ghallab, A.M.Alaoui, "Managing Efficiently Temporal Relations Through Indexed Spanning Trees", IJCAI, 1989.

[Ghallab, Laruelle, 1994] M.Ghallab , H.Laruelle, "Representation and Control in Ixtet, a Temporal Planner", AIPS, 1994.

[Kautz, Selman, 1999] H.Kautz, B.Selman, "Unifying SAT-based and Graph-based Planning", IJCAI, 1999.

[Laborie, Ghallab, 1995] P.Laborie, M.Ghallab, "Planning with Sharable Resource Constraints", IJCAI, 1995.

[Long, Fox, 2003] D.Long, M.Fox, "Exploiting a graphplan framework in temporal planning", ICAPS, 2003.

[Mausam, Weld, 2006] Mausam, D.S.Weld., "Probabilistic temporal planning with uncertain durations", AAAI, 2006.

[Miller, Firby, Dean, 1985] D.Miller, J.Firby, T.Dean, "Deadlines, travel time and robot problem solving", IJCAI, 1985.

[Oddi, Cesta, 2000] A.Oddi, A.Cesta, "Incremental forward checking for the disjunctive temporal constraint problem", ECAI, 2000.

[Penberthy, Weld, 1992] J.S.Penberthy, D.Weld, "UCPOP: A Sound, Complete, Partial-Order Planner for ADL", KR, 1992.

[Rintanen, 2007] J.Rintanen, "Complexity of Concurrent Temporal Planning", ICAPS, 2007.

[Smith, Frank, Jonsson, 2000] D.E.Smith, J.Frank, A.K.Jonsson, "Bridging the gap beween planning and scheduling", Knowledge Engineering Review, 15(1), 2000.

[Smith, Weld, 1999] D.E.Smith, D.Weld, "Temporal planning with mutual exclusion reasoning", IJCAI, 1999.

[Stergiou, Koubarakis, 1998] K.Stergiou, M.Koubarakis, "Backtracking Algorithms for Disjunctions of Temporal Constraints", AAAI, 1998.

[Tsamardinos, Pollack, 2003] I.Tsamardinos, M.Pollack, "Efficient solution techniques for disjunctive temporal reasoning problems", AI, 151, pp. 43-89, 2003.

[Vere, 1983] S.A.Vere, "Planning in time: windows and durations for activities and goals", IEEE Transactions on Pattern Analysis and Machine Intelligence, 5(3), 1983.

[Vidal, Geffner, 2006] V.Vidal, H.Geffner, "Branching and Pruning: An Optimal Temporal POCL Planner based on Constraint Programming", AI, 170 (3), 2006.

[Younes, Simmons, 2003] H.L.S.Younes, R.G.Simmons, "VHPOP: Versatile Heuristic Partial Order Planner", Journal of AI Research, 20, 2003.

[6] http://tlpgp.free.fr