Time-Optimal Planning in Temporal Problems

Antonio Garrido and **Eva Onaindia** and **Federico Barber**
Dpto. Sistemas Informaticos y Computacion
Universitat Politecnica de Valencia
Camino de Vera s/n, 46022, Valencia (Spain)

Abstract

This paper presents TPSYS, a *Temporal Planning SYStem*, which arises as an attempt to combine the ideas of Graphplan and TGP to solve temporal planning problems more efficiently. TPSYS is based on a three-stage process. The first stage, a preprocessing stage, facilitates the management of constraints on duration of actions. The second stage expands a temporal graph and obtains the set of temporal levels at which propositions and actions appear. The third stage, the plan extraction, obtains the plan of minimal duration by finding a proper flow of actions.

In real world planning problems which deal with time, it is necessary to discard the assumption that actions have the same duration. For instance, it is clear that in a logistics domain the action `fly plane(London, Moscow)` is longer than `fly plane(London, Paris)`. Hence, dealing with temporal planning problems requires to handle more complex constraints because it is necessary to select the right execution times for actions. Consequently, an important issue in temporal planning is to guarantee the plan which minimizes the global duration.

This paper builds on the work of Smith and Weld (the *Temporal Graphplan* algorithm, TGP, presented in (Smith and Weld 1999)) and examines the general question of including temporality on actions in a Graphplan-based approach (Blum and Furst 1997) by guaranteeing the plan of minimal duration. We present a *Temporal Planning SYStem* (TPSYS) which consists of three stages: a *preprocessing* stage, a temporal graph expansion stage and a plan extraction stage. The main features of TPSYS are:

- It is able to handle overlapping actions of different duration and guarantees the optimal plan, i.e. the plan of minimal duration.
- It defines a new classification of mutual exclusion relations: *static* mutexes which are time independent and *dynamic* mutexes which are time dependent.
- It expands a relaxed temporal graph (from now on *TG*), without maintaining `no_op` actions nor delete-edges, through temporal levels. Then, it performs a plan extraction (from now on *PE*) stage by selecting the appropriate actions in the *TG* to achieve the problem goals.

Related Work

Although temporal features in planning are not usually managed by classical planners, one of the first temporal planners on the last decade was O-Plan (Currie and Tate 1991) which integrates both planning and scheduling processes into a single framework. Other planners, such as IxTeT (Ghallab and Laruelle 1994), deal with resource availability and temporal constraints to represent constraints on time points. An attempt to integrate planning and scheduling is performed in HSTS (*Heuristic Scheduling Testbed System* (Muscettola 1994)) which defines an integrated framework to solve planning and scheduling tasks. This system uses multi-level heuristic techniques to manage resources under the constraints imposed by the action schedule. The parcPLAN approach (El-Kholy and Richards 1996) manages multiple capacity resources with actions which may overlap, instantiating time points in a similar way to our approach.

TGP (Smith and Weld 1999) introduces a complex mutual exclusion reasoning to handle actions of differing duration in a Graphplan context. TPSYS combines features of both Graphplan and TGP and introduces new aspects to improve performance. The reasoning on *conditional* mutex (involving time mutex) between actions, propositions and between actions and propositions is managed in TGP by means of inequalities which get complex in some problems and may imply an intractable reasoning on large problems (Smith and Weld 1999). On the contrary, the reasoning process in TPSYS is simplified thanks to the incorporation of several improvements:

- Static mutex relations between actions and between actions and propositions are calculated in a preprocessing stage because they only depend on the definition of the actions.
- TPSYS uses a multi-level temporal planning graph as Graphplan where each level represents an instant of time. While in TGP actions and propositions are only annotated with the first level at which they appear in the planning graph, TPSYS annotates all different instances of actions and propositions produced along time. The compact en-

Action	Dur	Precs	Effs

`ld(B1,BC,H)`	5	`at(B1,H)`
		`at(BC,H)`
		`free(BC)`
			`in(B1,BC)`
			`¬at(B1,H)`
			`¬free(BC)`

`mv(BC,H,U)`	5	`at(BC,H)`	`at(BC,U)`
			`¬at(BC,H)`

`uld(B1,BC,U)`	2	`in(B1,BC)`
		`at(BC,U)`
			`at(B1,U)`
			`free(BC)`
			`¬in(B1,BC)`

Table 1: Simplified *Briefcase* domain: necessary actions to achieve the goal `at(B1,U)`

coding of TGP reduces vastly the space costs but it increases the complexity of the search process, which may traverse cycles in the planning graph. However, the *PE* in TPSYS is straightforward because it merely consists of obtaining the plan as an acyclic *flow* of actions throughout the *TG*.

Our Temporal Planning SYStem

In TPSYS, a temporal planning problem is specified as a 4-tuple $\{\mathcal{I}_s, \mathcal{A}, \mathcal{F}_s, \mathcal{D}_{\max}\}$, where $\mathcal{I}_s$ and $\mathcal{F}_s$ represent the initial and final situation respectively, $\mathcal{A}$ represents the set of actions (with positive duration), and $\mathcal{D}_{\max}$ stands for the maximal duration of the plan required by the user. Time is modelled by $\mathbb{R}^+$ and their chronological order. A temporal proposition is represented by $\langle \mathtt{p}, \mathtt{t} \rangle$ where p denotes the proposition and $\mathtt{t} \in \mathbb{R}^+$ represents the time at which p is produced. Hence, $\mathcal{I}_s$ and $\mathcal{F}_s$ are formed by two set of temporal propositions $\{\langle \mathtt{p_i}, \mathtt{t_i} \rangle / \mathtt{t_i} \leq \mathcal{D}_{\max}\}$.

We will make use of the action domain defined in Table 1, which presents a description of the actions of the *Briefcase* domain, to show the behaviour of our system. Only three actions are defined, those which are necessary to transport a book (B1) from home (H) to university (U) by using a briefcase (BC).

First Stage: Preprocessing

TPSYS calculates the static mutual exclusions which will allow us to speed up the following two stages. A mutex relationship between actions is defined as in Graphplan (Blum and Furst 1997). Mutex between propositions appears as a consequence of mutex between actions. Thus, two propositions p and q are mutex if all actions that achieve p are mutex with all actions that achieve q.

Definition. ***Static mutex between actions.*** Actions a and b are statically mutex if they cannot be executed in parallel (Graphplan's interference). For instance, in Table 1, actions `ld(B1,BC,H)` and `uld(B1,BC,U)` are statically mutex because of the conflicting effect `in(B1,BC)`.

Definition. ***Static ap-mutex (static action/proposition mutex).*** One action a is statically *ap-mutex* with a proposition p iff $\mathtt{p} \in \mathtt{del\_effs(a)}$. For instance, `ld(B1,BC,H)` is *ap-mutex* with `at(B1,H)` and `free(BC)` in Table 1.

Second Stage: Temporal Graph Expansion

Definition. ***Temporal graph (TG).*** A *TG* is a directed, layered graph with proposition and action nodes, and precondition- and add-edges following the same structure as Graphplan. Each level is labelled with a number representing the instant of time at which propositions are present and actions start their execution. Levels are ordered by their instant of time.

Definition. Instance of an action. We define an instance of an action a as the triple $\langle \mathtt{a,s,e} \rangle$ where a denotes the action and $\mathtt{s,e} \in \mathbb{R}^+$ represent the time when the instance starts and ends executing, respectively ($\mathtt{e} = \mathtt{s} + duration(\mathtt{a})$).

Definition. ***Proposition level.*** A proposition level $P_{[t]}$ is formed by the set of temporal propositions $\{\langle \mathtt{p_i}, \mathtt{t_i} \rangle / \mathtt{t_i} \leq \mathtt{t}\}$ present at time t which verify $\langle \mathtt{p_i}, \mathtt{t_i} \rangle \in \mathcal{I}_s \vee \exists \langle \mathtt{a_i}, \mathtt{s_i}, \mathtt{e_i} \rangle / \mathtt{p_i} \in \mathtt{add\_effs(a_i)}, \mathtt{e_i} = \mathtt{t_i}$.

Definition. ***Dynamic mutex between temporal propositions*** **at** $P_{[t]}$**.** Let $\{\langle \mathtt{a_i}, \mathtt{s_i}, \mathtt{t_i} \rangle\}$ and $\{\langle \mathtt{b_j}, \mathtt{s_j}, \mathtt{t_j} \rangle\}$ be two sets of instances of actions which achieve $\langle \mathtt{p}, \mathtt{t_i} \rangle$,$\langle \mathtt{q}, \mathtt{t_j} \rangle \in P_{[t]}$ respectively. Temporal propositions $\langle \mathtt{p}, \mathtt{t_i} \rangle$ and $\langle \mathtt{q}, \mathtt{t_j} \rangle$ are dynamically mutex at $P_{[t]}$ iff i) $\forall \alpha, \beta / \alpha \in \{\langle \mathtt{a_i}, \mathtt{s_i}, \mathtt{t_i} \rangle\}, \beta \in \{\langle \mathtt{b_j}, \mathtt{s_j}, \mathtt{t_j} \rangle\}$, α and β overlap and ii) $\mathtt{a_i}$ and $\mathtt{b_j}$ are statically mutex. A dynamic mutex expires as new levels are expanded further in the *TG*.

Definition. ***Action level.*** An action level $A_{[t]}$ is formed by the set of instances of actions $\{\langle \mathtt{a_i}, \mathtt{t}, \mathtt{e_i} \rangle\}$ which start their execution at time t.

Proposition. Let $P_{[t]}$ ($t \leq \mathcal{D}_{\max}$) be the earliest proposition level at which all temporal propositions in $\mathcal{F}_s$ are not pairwise dynamically mutex. Under this assumption, no correct plan can be found before time t.

TPSYS adopts the same conservative model of action as TGP (Smith and Weld 1999). The second stage expands the *TG* by alternating proposition and action levels through a forward-chaining process. Starting at $P_{[0]}$, the algorithm moves incrementally in time throughout the *TG* generating new action and proposition levels. At each action level $A_{[t]}$, the algorithm generates the entire set of instances of actions which start their execution at t because their preconditions are not dynamically mutex at $P_{[t]}$. After generating each instance of an action, the propositions in `add_effs` are added into the proper proposition level (according to the duration of each action). The *TG* expansion terminates once all temporal propositions in the final situation are present in $P_{[t]}$ and none are pairwise dynamically mutex (i.e. $\mathcal{F}_s$ is satisfied in $P_{[t]}$). If $t > \mathcal{D}_{\max}$ the algorithm outputs '*Failure*' because no feasible plan can be found earlier than $\mathcal{D}_{\max}$.

The resulting *TG* for the domain defined in Table 1 is shown in Figure 1. Action `uld(B1,BC,U)` cannot start at $A_{[5]}$ because its preconditions `in(B1,BC)` and `at(BC,U)` are dynamically mutex at $P_{[5]}$ and they cannot be simultaneously available until $P_{[10]}$. At $A_{[10]}$, `uld(B1,BC,U)` is applicable thus obtaining the goal `at(B1,U)` at $P_{[12]}$ (terminating the second stage).

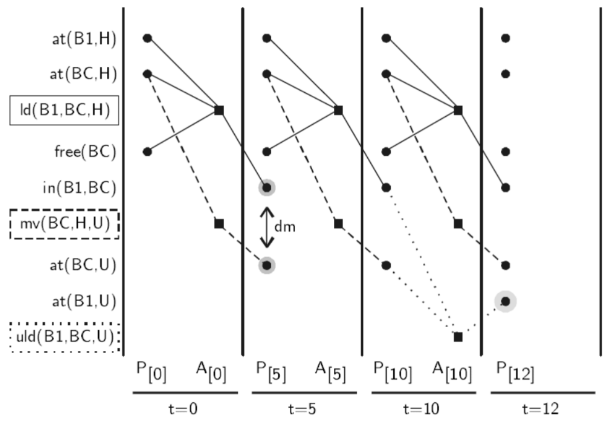

Figure 1: *Temporal Graph* for the *Briefcase* problem defined in Table 1

Third Stage: Plan Extraction

This stage is a backward search process throughout the *TG* to extract a feasible plan. Two data structures `PlannedActs` and `GoalsToSatisfy`, which are indexed by a level, are used. `PlannedActs`, which is initialized empty, stores the instances of actions planned at each action level. `GoalsToSatisfy` stores the temporal propositions to be satisfied at each proposition level, and it is initialized by inserting all the temporal propositions in $\mathcal{F}_s$.

Assuming the *PE* process starts from the proposition level $P_{[t]}$ (that is, the search starts from time t in the *TG*), where all temporal goals in $\mathcal{F}_s$ are not dynamically mutex, the algorithm proceeds in the following way:

1. If $t = 0$ and $\texttt{GoalsToSatisfy}[t] \nsubseteq \mathcal{I}_s$, then fail (backtrack) –this is the base case for the recursive process.
2. If $\texttt{GoalsToSatisfy}[t] = \phi$ then move backwards in time (t =previous level in the *TG*) and go to step 1 to satisfy the goals at t.
3. Extract a temporal proposition $\langle \texttt{p}, t\rangle$ from $\texttt{GoalsToSatisfy}[t]$.
4. Select an instance of an action $\alpha = \langle \texttt{a}_\texttt{i}, \texttt{s}_\texttt{i}, \texttt{e}_\texttt{i}\rangle / \texttt{p} \in \texttt{add\_effs}(\texttt{a}_\texttt{i})$, $\texttt{e}_\texttt{i} \leq t$ (*backtracking point* to guarantee completeness). In order to guarantee the correctness of the plan, α is discarded (selecting another instance of an action by *backtracking* to step 4) if at least one of the following conditions holds; i) $\exists\beta = \langle \texttt{b}_\texttt{j}, \texttt{s}_\texttt{j}, \texttt{e}_\texttt{j}\rangle \in \texttt{PlannedActs} / \alpha$ and β overlap and $\texttt{a}_\texttt{i}$ and $\texttt{b}_\texttt{j}$ are statically mutex, or ii) $\exists\langle \texttt{q}, \texttt{e}_\texttt{i}\rangle \in \texttt{GoalsToSatisfy} / \texttt{a}_\texttt{i}$ is statically *ap-mutex* with q. Otherwise, p is satisfied and the structures $\texttt{PlannedActs}[\texttt{s}_\texttt{i}]$ and $\texttt{GoalsToSatisfy}[\texttt{s}_\texttt{i}]$ are updated with α and $\texttt{precs}(\texttt{a}_\texttt{i})$ respectively. Then, the algorithm goes to step 2 to satisfy another (sub)goal.

Proposition. **TPSYS is complete and optimal.** In TPSYS, all levels at which propositions and actions appear are all generated during the *TG* expansion. Therefore, if a plan exists for the problem, it will be found in the *TG*. Additionally, since all instances of actions are considered in the *PE* process and the *TG* is expanded through time, the first solution TPSYS finds is the plan of minimal duration.

Problem	TPSYS	TGP
tgp-AB-q	4	60
tgp-AB-pq	5	90
tgp-AC-r	4	80
tgp-AC-pr	5	80
tgp-ABDE-r	4	70

Table 2: Results of comparison between TPSYS and TGP (times are in milliseconds)

Some Experimental Results

Although comparison between our approach and other planning systems is quite difficult because they are based on different algorithms, we made a comparison between TPSYS and TGP on the examples provided by TGP. The experiments (Table 2) were performed in a Celeron 400 MHz with 64 Mb and show the performance of TPSYS is better than TGP for these problems. Consequently, TPSYS seems quite promising to deal with temporal planning problems.

Conclusions and Future Work

In this paper we have presented TPSYS, a system for dealing with temporal planning problems. TPSYS contributes on a classification into static and dynamic mutual exclusion relations. This allows to perform a preprocessing stage which calculates static mutexes between actions and between actions and propositions to speed up the following stages. The second stage expands a *TG* with features of both Graphplan and TGP planning graphs. The third stage guarantees that the first found plan has the minimal duration. From our experience and the obtained results we think TPSYS is promising to solve temporal planning problems.

The presented work constitutes a first step towards an integrated system for planning and scheduling. Such a system will be able to manage temporal constraints on actions and to reason on shared resource utilization. Additionally, the system will apply several optimization criteria to obtain the plan of minimal duration or the plan of minimal cost.

Acknowledgments

This work has been partially supported by the Project n. 20010017 - *Navigation of Autonomous Mobile Robots* of the Universitad Politecnica de Valencia.

References

Blum, A., and Furst, M. 1997. Fast planning through planning graph analysis. *Artificial Intelligence* 90:281–300.

Currie, K., and Tate, A. 1991. O-plan: the open planning architecture. *Artificial Intelligence* 52(1):49–86.

El-Kholy, A., and Richards, B. 1996. Temporal and resource reasoning in planning: the parcPLAN approach. In *Proc. 12th European Conference on Artificial Intelligence (ECAI-96)*, 614–618.

Ghallab, M., and Laruelle, H. 1994. Representation and control in IxTeT, a temporal planner. In *Proc. 2nd Int. Conf. on AI Planning Systems*, 61–67. Hammond.

Muscettola, N. 1994. HSTS: Integrating planning and scheduling. In Zweben, M., and Fox, M., eds., *Intelligent Scheduling*. San Mateo, CA: Morgan Kaufmann. 169–212.

Smith, D., and Weld, D. 1999. Temporal planning with mutual exclusion reasoning. In *Proc. 16th Int. Joint Conf. on AI (IJCAI-99)*, 326–337.