

# A Temporal Planning System for Time-Optimal Planning

Antonio Garrido, Eva Onaindía and Federico Barber

Dpto. Sistemas Informáticos y Computación  
Universidad Politécnica de Valencia  
Camino de Vera s/n, 46022 Valencia, Spain  
Fax: (+34) - 963877359  
{agarridot,onaindia,fbarber}@dsic.upv.es

**Abstract.** Dealing with temporality on actions presents an important challenge to AI planning. Unlike **Graphplan**-based planners which alternate levels of propositions and actions in a regular way, introducing temporality on actions unbalance this symmetry. 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 flow of actions through the temporal graph. The experiments show the utility of our system for dealing with temporal planning problems.

**Key words:** planning, temporal planning, planning and reasoning about actions

## 1 Introduction

In many real world planning problems, time plays a crucial role. In these problems it is necessary to discard the assumption that actions have the same duration. For instance, it is clear that in a logistics domain the operator **fly plane** may be longer than **load plane**, and the action **fly plane(London,Moscow)** may be longer than **fly plane(London,Paris)**. Hence, dealing with temporal planning problems requires to handle a set of more complex constraints since the difficulty does not only lie in the process of action selection [11], but also in the process of selecting the right execution times for actions. In addition, the criterion for optimization changes because in temporal contexts the interest lies in obtaining a plan of minimal duration rather than a plan of minimal number of actions. Therefore, finding the plan which minimizes the global duration becomes an important issue in temporal planning.

Dealing with temporality on planning has also been tackled by introducing independent scheduling techniques in charge of doing all the reasoning on

time and resources. Under this approach of two separated processes, the planner obtains a plan, which is validated afterwards by the scheduler. In case the scheduler determines this plan is unfeasible, the planner must obtain a new plan and the same procedure is repeated again. Obviously, this approach entails a large overhead in the overall process. For that reason, a few attempts at integrating planning and scheduling are being carried out [5]. TPSYS follows the guidelines of these latter approaches by achieving a total integration of time and actions in the same framework.

This paper aims to solve the above drawbacks. It builds on the work of Smith and Weld (the *Temporal Graphplan* algorithm, TGP, presented in [12]) and examines the general question of including temporality on actions in a Graphplan-based approach [1] 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 and transitory.
- It expands a relaxed temporal graph (from now on *TG*) without maintaining `no-op` actions nor delete-edges. The *TG* is incrementally expanded through temporal levels defined by the instants of time at which propositions appear.
- It performs a plan extraction (from now on *PE*) stage by selecting the appropriate actions in the *TG* to achieve the problem goals and guaranteeing the plan of minimal duration. Consequently, the algorithm can also solve problems of the type ‘*obtain a plan of duration shorter than  $\mathcal{D}_{\max}$* ’.

## 2 Related Work

Despite the great effort to introduce temporal constraints and resources in planning, none of the existing works have been largely used. Presumably, the reason is they do not exhibit a good performance when dealing with temporal problems due to the complexity these problems entail. If we focus on the last decade, one of the first temporal planners was *O-Plan* [2] which integrates both planning and scheduling processes into a single framework. Then, other planners such as *ZENO* [10] or *lXTeT* [6] appeared in the literature. Although *lXTeT* does not manage disjunctive constraints, it deals with resource availability and temporal constraints by using a *TCN* (*Temporal Constraint Network* [3]) to represent constraints on time points. An alternative system for integrating planning and scheduling is performed in *HSTS* (*Heuristic Scheduling Testbed System* [9]) which defines an integrated framework to solve planning and scheduling tasks in specific domains of spatial missions. This system uses multi-level heuristic techniques to manage resources under the constraints imposed by the action schedule. The

parcPLAN approach [4] manages multiple capacity resources with actions which may overlap, instantiating time points in a similar way to our approach. parcPLAN behaves efficiently if there are enough resources but with stricter resource limitations it becomes more inefficient. Köehler deals with planning under resource constraints by means of time producer/consumer actions but without incorporating an explicit model of time in [8].

TGP [12] 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. While TGP aims to demonstrate that its mutual exclusion reasoning remains valuable in a temporal setting, TPSYS aims to guarantee the optimal plan for any temporal planning problem.

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. In fact, the application of conditional mutexes imposes a set of sophisticated formulae (even with binary disjunctions) which may imply an intractable reasoning on large problems [12]. 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. This allows us to speed up the process of calculating the dynamic mutex relations between propositions while generating the *TG*, obtaining better execution times than TGP.
- TPSYS uses a multi-level temporal planning graph as Graphplan where each level represents an instant of time. Both TPSYS and TGP build a relaxed graph by ignoring `no-op` actions and delete-edges which allows an extremely fast graph expansion<sup>1</sup> [7]. 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 encoding 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*. The experiments show that using a much more informed *TG* vastly reduces the overhead during the *PE*, thus obtaining better global execution times than TGP.

In addition to the temporally goals which TGP is able to manage, TPSYS can also manage propositions of the initial situation which hold at other times than  $t = 0$ .

---

<sup>1</sup> The *TG* expansion represents a small percentage of the execution time in TPSYS. In general, hundreds of temporal levels can be generated in few seconds for many classical domains.

### 3 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, 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. Each action is assigned a nondisjunctive positive duration, which may be different. A temporal proposition is represented by  $\langle p, \tau \rangle$  where  $p$  denotes the proposition and  $\tau \in \mathbb{R}^+$  represents the time at which  $p$  is produced. Hence,  $\mathcal{I}_s$  and  $\mathcal{F}_s$  are formed by two set of temporal propositions  $\{\langle p_i, \tau_i \rangle / \tau_i \leq \mathcal{D}_{\max}\}$ . It must be noticed that propositions in  $\mathcal{I}_s$  ( $\mathcal{F}_s$ ) can be placed (required) at any time  $\tau_i$  through the execution of the plan.

TPSYS follows three consecutive stages, as shown in Fig. 1. After the first stage, the second and the third stage are executed in an interleaved way until a plan is found or the duration of the plan exceeds  $\mathcal{D}_{\max}$ .

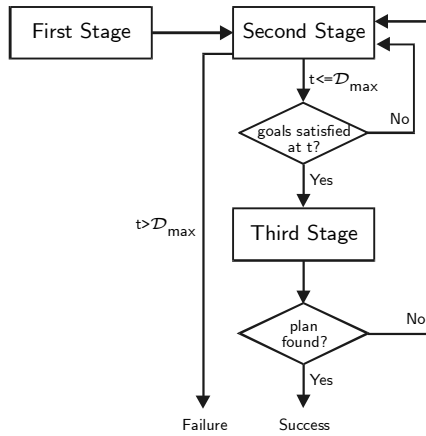


Fig. 1. Flow chart of the three stages of TPSYS

In this section, we will make use of the action domain defined in Table 1 to show the behaviour of our system. Table 1 presents a description of the actions (with duration) of the well-known *Briefcase* domain. In order to simplify the domain 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).

#### 3.1 First Stage: Preprocessing

At this preprocessing stage, TPSYS calculates the static mutual exclusions which will facilitate the computation of dynamic mutex relations during the second stage. A mutex relationship between actions is defined as follows [1]: two actions

Action	Duration	Precs.	Effects
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: only three actions are defined to achieve the goal  $\text{at}(B1,U)$

$\mathbf{a}$  and  $\mathbf{b}$  are mutex if  $\mathbf{a}$  deletes a precondition of  $\mathbf{b}$ , or  $\mathbf{a}$  and  $\mathbf{b}$  have conflicting effects, in whose case they cannot be executed in parallel. Mutex between propositions appears as a consequence of mutex between actions. Thus, two propositions  $\mathbf{p}$  and  $\mathbf{q}$  are mutex if all actions for obtaining  $\mathbf{p}$  are mutex with all actions for obtaining  $\mathbf{q}$ .

Let  $\mathbf{a}$  and  $\mathbf{b}$  be two actions, and let  $S_1, S_2 \in \{\text{precs}(\mathbf{a}), \text{add} - \text{effs}(\mathbf{a}), \text{del} - \text{effs}(\mathbf{a}), \text{precs}(\mathbf{b}), \text{add} - \text{effs}(\mathbf{b}), \text{del} - \text{effs}(\mathbf{b})\} / S_1 \neq S_2$ . We define the function  $\text{coincidences}(S_1, S_2)$  as a boolean function which holds iff  $\exists \mathbf{p} / \mathbf{p} \in S_1 \wedge \mathbf{p} \in S_2$ . According to this, we introduce the following definitions:

**Definition 1. *Static mutex between actions.*** Actions  $\mathbf{a}$  and  $\mathbf{b}$  are statically mutex if they cannot be executed in parallel. Although this relationship is equivalent to *Graphplan*'s interference, we break it down into two different types of mutex relationships. Hence, if two statically mutex actions are given at the same instant of time, these actions will have to be executed in one of the following ways:

a) *Consecutive actions in any order, iff*

$$\text{coincidences}(\text{add} - \text{effs}(\mathbf{a}), \text{del} - \text{effs}(\mathbf{b})) \vee \text{coincidences}(\text{add} - \text{effs}(\mathbf{b}), \text{del} - \text{effs}(\mathbf{a})).$$

Clearly, if  $\mathbf{a}$  and  $\mathbf{b}$  have conflicting effects, the correct execution order is 'a after b' or 'b after a'. In Table 1, actions  $\text{ld}(B1,BC,H)$  and  $\text{uld}(B1,BC,U)$  can be executed in any order because their only interdependency is the conflicting effect  $\text{in}(B1,BC)$ .

b) *Consecutive actions in an after order, iff*

$$\text{coincidences}(\text{del} - \text{effs}(\mathbf{a}), \text{precs}(\mathbf{b})) \wedge \neg \text{coincidences}(\text{precs}(\mathbf{a}), \text{del} - \text{effs}(\mathbf{b})).$$

Clearly, if  $\mathbf{a}$  deletes a precondition of  $\mathbf{b}$  but  $\mathbf{b}$  does not delete any precondition of  $\mathbf{a}$ , the correct execution order is 'a after b'. In Table 1, action  $\text{mv}(BC,H,U)$  must be executed after  $\text{ld}(B1,BC,H)$  because of the proposition  $\text{at}(BC,H)$ .

**Definition 2. Static ap-mutex (static action/proposition mutex).** One action  $\mathbf{a}$  is statically ap-mutex with a proposition  $\mathbf{p}$  iff  $\mathbf{p} \in \text{del} - \text{effs}(\mathbf{a})$ . For instance,  $\text{ld}(\mathbf{B1}, \mathbf{BC}, \mathbf{H})$  is ap-mutex with  $\text{at}(\mathbf{B1}, \mathbf{H})$  and  $\text{free}(\mathbf{BC})$  in Table 1.

### 3.2 Second Stage: Temporal Graph Expansion

We adopt the same conservative model of action as in TGP, in which i) all preconditions must hold at the start of the execution of the action, ii) preconditions *not deleted* by the action itself must hold during the entire execution of the action, iii) effects are undefined during the execution and only guaranteed to hold at the end of the action, and additionally, iv) actions can start their execution as soon as all their preconditions hold.

**Definition 3. Temporal graph (TG).** A TG is a directed, layered graph with two kind of nodes (proposition and action nodes) and two kinds of edges (precondition-edges and add-edges). The TG alternates proposition and action levels like *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. This way, the algorithm can easily move from a level  $t$  to the next level  $t' > t$  during the TG expansion, and to the previous level  $t'' < t$  during the plan extraction. The TG is expanded by a forward-chaining process which simply adds the add-effects of actions (delete-effects are ignored) at the proper level according to the duration of actions.

**Definition 4. Instance of an action.** We define an instance of an action  $\mathbf{a}$  as the triple  $\langle \mathbf{a}, \mathbf{s}, \mathbf{e} \rangle$  where  $\mathbf{a}$  denotes the action and  $\mathbf{s}, \mathbf{e} \in \mathbb{R}^+$  represent the time when the instance starts and ends executing, respectively ( $\mathbf{e} = \mathbf{s} + \text{duration}(\mathbf{a})$ ).

**Definition 5. Proposition level.** A proposition level  $P_{[t]}$  is formed by the set of temporal propositions  $\{ \langle \mathbf{p}_i, \mathbf{t}_i \rangle / \mathbf{t}_i \leq \mathbf{t} \}$  present at time  $\mathbf{t}$  which verify  $\langle \mathbf{p}_i, \mathbf{t}_i \rangle \in \mathcal{I}_s \vee \exists \langle \mathbf{a}_i, \mathbf{s}_i, \mathbf{e}_i \rangle / \mathbf{p}_i \in \text{add} - \text{effs}(\mathbf{a}_i), \mathbf{e}_i = \mathbf{t}_i$ . It must be noticed that ' $<$ ' between  $\mathbf{t}_i$  and  $\mathbf{t}$  is used to denote  $\mathbf{p}_i$  persists in time. Consequently, if a proposition is present at  $P_{[t]}$ , it will appear at all  $P_{[t']}$  such that  $\mathbf{t}' > \mathbf{t}$ .

**Definition 6. Dynamic mutex between temporal propositions at  $P_{[t]}$ .** Let  $\{ \langle \mathbf{a}_i, \mathbf{s}_i, \mathbf{t}_i \rangle \}$  and  $\{ \langle \mathbf{b}_j, \mathbf{s}_j, \mathbf{t}_j \rangle \}$  be two sets of instances of actions which achieve  $\langle \mathbf{p}, \mathbf{t}_i \rangle, \langle \mathbf{q}, \mathbf{t}_j \rangle \in P_{[t]}$  respectively (i.e.,  $\mathbf{p} \in \text{add} - \text{effs}(\mathbf{a}_i)$  and  $\mathbf{q} \in \text{add} - \text{effs}(\mathbf{b}_j)$ ). Temporal propositions  $\langle \mathbf{p}, \mathbf{t}_i \rangle$  and  $\langle \mathbf{q}, \mathbf{t}_j \rangle$  are dynamically mutex at  $P_{[t]}$  iff i)  $\forall \alpha, \beta / \alpha \in \{ \langle \mathbf{a}_i, \mathbf{s}_i, \mathbf{t}_i \rangle \}, \beta \in \{ \langle \mathbf{b}_j, \mathbf{s}_j, \mathbf{t}_j \rangle \}, \alpha$  and  $\beta$  overlap and ii)  $\mathbf{a}_i$  and  $\mathbf{b}_j$  are statically mutex. Otherwise, temporal propositions are nondynamically mutex at  $P_{[t]}$ . Loosely speaking, two propositions are dynamically mutex at  $P_{[t]}$  if they cannot be simultaneously available at time  $\mathbf{t}$ . Obviously, a dynamic mutex may expire as new levels are expanded further in the TG, and the involved statically mutex actions are ordered in sequence.

**Definition 7. Action level.** An action level  $A_{[t]}$  is formed by the set of instances of actions  $\{ \langle \mathbf{a}_i, \mathbf{t}, \mathbf{e}_i \rangle \}$  which start their execution at time  $\mathbf{t}$  because  $\forall \langle \mathbf{p}, \mathbf{t}_i \rangle, \langle \mathbf{q}, \mathbf{t}_j \rangle \in P_{[t]} / \mathbf{p}, \mathbf{q} \in \text{precs}(\mathbf{a}_i), \langle \mathbf{p}, \mathbf{t}_i \rangle, \langle \mathbf{q}, \mathbf{t}_j \rangle$  are nondynamically mutex at  $P_{[t]}$ .

**Proposition 1.** *Let  $P_{[t]}$  ( $t \leq \mathcal{D}_{\max}$ ) be the earliest proposition level at which all temporal propositions in  $\mathcal{F}_s$  are pairwise nondynamically mutex. Under this assumption, no correct plan can be found before time  $t$  (i.e., the duration of a correct plan will never be shorter than  $t$ ).*

*Proof.* According to Definition 6 the proof is simple: if each pair of goal propositions are not simultaneously available until time  $t$ , the minimal duration as possible of a plan will be  $t$ . Notice, however, that this statement does not guarantee that the minimal duration of a correct plan is always  $t$  because delete-effects have been ignored during the *TG* expansion and, consequently, some goal propositions might have been deleted.

The second stage of TPSYS expands the *TG* (see Fig. 2) by alternating proposition and action levels through a fast forward-chaining process. The *TG* expansion differs from *Graphplan* in the following points:

- No `op` actions are never generated because propositions persist in time through proposition levels.
- Each instance of an action  $\langle \mathbf{a}, \mathbf{s}, \mathbf{e} \rangle$  in  $A_{[s]}$  does not add its add-effects into  $P_{[s+1]}$  but in  $P_{[e]}$ . Thus, different proposition levels are now generated from a given action level.
- The only mutual exclusion relationship checked during the *TG* expansion is the dynamic mutex relationship between temporal propositions.

Starting at  $P_{[0]}$  (we will suppose there exists at least one temporal proposition in  $\mathcal{I}_s$  which belongs to  $P_{[0]}$ ), the algorithm moves incrementally in time (from one level to the next) 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 because their preconditions are nondynamically mutex at  $P_{[t]}$ . After generating each instance of an action, the propositions in add-effects are added into the proper proposition level. 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  $P_{[t]}$ ). Moreover, if  $t > \mathcal{D}_{\max}$  the algorithm outputs ‘*Failure*’ because no feasible plan can be found earlier than  $\mathcal{D}_{\max}$ .

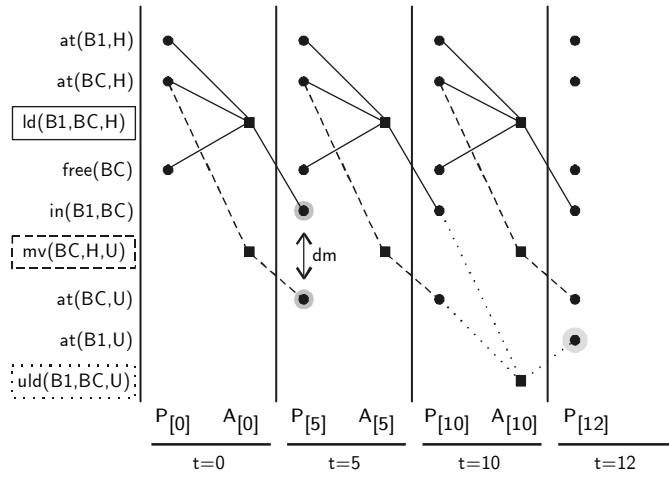
The resulting *TG* for the domain defined in Table 1 is shown in Fig. 3. Action `uld(B1,BC,U)` cannot start at  $A_{[5]}$  because its preconditions `in(B1,BC)` and `at(BC,U)` are dynamically mutex at  $P_{[5]}$  and they cannot be simultaneously available until  $P_{[10]}$ . Goal proposition `at(B1,U)` is achieved at  $P_{[12]}$  and, therefore, the *TG* expansion terminates at that level.

### 3.3 Third Stage: Plan Extraction

The third stage is a backward search process throughout the *TG* to extract a feasible plan. The algorithm uses two data structures `PlannedActs` and `GoalsToSatisfy` which are indexed by a level. `PlannedActs`, which is initialized empty, stores the instances of actions planned at each action level. `GoalsToSatisfy` stores the

Algorithm *TG* expansion  
 $\forall \langle \mathbf{p}_i, \mathbf{t}_i \rangle \in \mathcal{I}_s$  do initialize  $P_{[\mathbf{t}_i]}$   
 $t = 0$   
while  $(t \leq \mathcal{D}_{\max}) \wedge (\mathcal{F}_s$  is not satisfied in  $P_{[t]})$  do  
 $\forall \alpha = \langle \mathbf{a}_i, t, \mathbf{e}_i \rangle$  which starts at  $A_{[t]}$  do  
 $A_{[t]} = A_{[t]} \cup \alpha$   
 $P_{[\mathbf{e}_i]} = P_{[\mathbf{e}_i]} \cup \mathbf{add} - \mathbf{effs}(\mathbf{a}_i)$   
 $t =$  next level in the *TG*  
endwhile

**Fig. 2.** Algorithm for the *TG* expansion



**Fig. 3.** Temporal Graph for the *Briefcase* problem defined in Table 1

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 nondynamically mutex, the algorithm proceeds in the following way:

1. If  $t = 0$  and  $\text{GoalsToSatisfy}[t] \not\subseteq \mathcal{I}_s$ , then fail (backtrack) –this is the base case for the recursive process.
2. If  $\text{GoalsToSatisfy}[t] = \phi$  then move backwards in time ( $t = \text{previous level in the } TG$ ) and go to step 1 to satisfy the goals at  $t$ .
3. Extract a temporal proposition  $\langle \mathbf{p}, t \rangle$  from  $\text{GoalsToSatisfy}[t]$ .
4. Select an instance of an action<sup>2</sup>  $\alpha = \langle \mathbf{a}_i, \mathbf{s}_i, \mathbf{e}_i \rangle / \mathbf{p} \in \text{add} - \text{effs}(\mathbf{a}_i), \mathbf{e}_i \leq t$  (*backtracking point* to guarantee completeness). In order to guarantee the correctness of the plan,  $\alpha$  is discarded if at least one of the following conditions holds:
  - $\exists \beta = \langle \mathbf{b}_j, \mathbf{s}_j, \mathbf{e}_j \rangle \in \text{PlannedActs} / \alpha$  and  $\beta$  overlap and  $\mathbf{a}_i$  and  $\mathbf{b}_j$  are statically mutex.
  - $\exists \langle \mathbf{q}, \mathbf{e}_i \rangle \in \text{GoalsToSatisfy} / \mathbf{a}_i$  is statically *ap-mutex* with  $\mathbf{q}$  (i.e.  $\alpha$  deletes  $\mathbf{q}$ ).

If  $\alpha$  is discarded, another instance of an action is selected by *backtracking* to step 4. Otherwise,  $\mathbf{p}$  is satisfied and the structures  $\text{PlannedActs}[\mathbf{s}_i]$  and  $\text{GoalsToSatisfy}[\mathbf{s}_i]$  are updated with  $\alpha$  and  $\text{precs}(\mathbf{a}_i)$  respectively. Then, the algorithm goes to step 2 to satisfy another (sub)goal.

Since delete-effects have been ignored during the *TG* expansion, it may be not possible to achieve the problem goals from the proposition level  $P_{[t]}$ . Thus, if the *PE* process does not find a feasible plan from the proposition level  $P_{[t]}$ , TPSYS backs to the second stage to continue extending more levels before executing the third stage again (see Fig. 1).

**Proposition 2.** *TPSYS is complete and optimal.*

Informally, we can show TPSYS is complete and optimal. In TPSYS, all levels at which propositions and actions may appear are generated during the *TG* expansion. It is clear that, if a solution plan exists for the problem, this plan will be found in the *TG* comprised from  $P_{[0]}$  to one of the generated proposition levels and it cannot be present in a proposition level which does not appear in the *TG*. Additionally, when the *PE* process searches for a correct plan, all instances of actions which achieve the (sub)goals are considered. Therefore, if a solution exists, TPSYS finds it.

On the other hand, let us suppose the *PE* process finds a correct plan of duration  $t$  (represented by  $\mathcal{P}_t$ ) which is not optimal. In that case, there would exist an optimal correct plan  $\mathcal{P}'_{t'}$  (of duration  $t' < t$ ) not found by the *PE* process. Since  $\mathcal{P}'_{t'}$  is a correct plan, all the temporal goals in  $\mathcal{F}_s$  would be nondynamically

<sup>2</sup> In TPSYS, all instances of actions  $\alpha$  are executed as late as possible, unless this leads to an unfeasible plan.

mutex at proposition level  $P_{[t']}$  and, therefore, the  $TG$  expansion would have terminated at that level. After terminating the  $TG$  expansion at proposition level  $P_{[t']}$ , the  $PE$  process would have found the plan  $\mathcal{P}'_{t'}$ , and the proposition level  $P_{[t]}$  would have never been studied.

### 3.4 Application Example

Here, we study a simple application example to illustrate the behaviour of our system. This abstract example shows how the proposition and action levels of the  $TG$  are generated. The description of the actions and their static mutex are shown in Table 2.  $\mathcal{I}_s = \{ \langle p, 0 \rangle, \langle q, 10 \rangle \}$ , so proposition  $q$  is not available until time 10.  $\mathcal{F}_s = \{ \langle s, 50 \rangle, \langle t, 50 \rangle, \langle u, 50 \rangle \}$  which implies all the goals must be satisfied no later than 50 ( $\mathcal{D}_{\max} = 50$ ).

Action	Duration	Precs.	Effects	Static mutex
a	10	p, q	r	d
b	5	r	s	c
c	20	p	$\neg s, t$	b, d
d	15	r	$\neg p, u$	a, c

**Table 2.** Domain of the actions in the application example

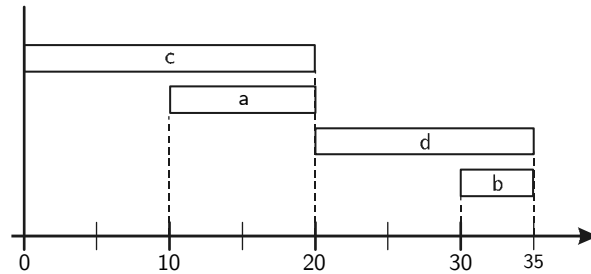
An outline with the basic information of the  $TG$  is shown in Table 3. In this table, we can see the proposition and action levels generated. Although proposition levels have been generated until  $P_{[50]}$ , the algorithm terminates the  $TG$  expansion at  $P_{[35]}$  because all temporal goals in  $\mathcal{F}_s$  are nondynamically mutex at that level. During the  $PE$  stage, the algorithm selects the instances of actions which obtain these goals, then the preconditions of these actions and so on. The plan obtained by TPSYS is shown in Fig. 4. The optimal duration of the plan is 35 and therefore, if the user constrains the maximal duration to  $\mathcal{D}_{\max} < 35$ , the algorithm will output ‘*Failure*’ because no plan exists.

## 4 Some Experimental Results

To date, our work has been mainly focused on the validation of our system, rather than on code optimization. Based on our experience, TPSYS seems quite promising to deal with temporal planning problems. 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. The experiments were performed in a 64 Mb. memory Celeron 400 MHz. We solved each problem (some simple, typical *benchmarks* used in AIPS 1998 competition

Temporal level $t$	$P_{[t]}$	$A_{[t]}$
0	p	c
10	p,q	a,c
20	p,q,r,t	a,b,c,d
25	p,q,r,s,t	a,b,c,d
30	p,q,r,s,t	a,b,c,d
35	p,q,r,s,t,u	-
40	p,q,r,s,t,u	-
45	p,q,r,s,t,u	-
50	p,q,r,s,t,u	-

**Table 3.** Outline of the *TG* generated by the algorithm



**Fig. 4.** Optimal plan found by TPSYS for the application example

and some examples provided by TGP<sup>3</sup>) with TPSYS and TGP. We can see the results in Table 4 and how the performance of TPSYS is better than TGP in these problems (even in all the examples provided by TGP). Moreover, TPSYS was able to solve problems which TGP was unable, such as **sussman-anomaly** and **briefcase-2b-1c** (which must transport two books with only one container).

As in **parcPLAN** [4], the main drawback of TPSYS appears in problems with limited resources, such as **briefcase-2b-1c**. In this case it is necessary to plan additional actions to release the used resources and to make them available again. In these problems, the goals are nondynamically mutex at a proposition level  $P_{[t]}$ , but a valid plan is not found at time  $t$ . Unfortunately, dynamic mutex relationships are not always enough to determine a proper proposition level from which a feasible plan can be found.

<sup>3</sup> The source code of TGP and the examples it provides have been obtained at <ftp://ftp.cs.washington.edu/pub/ai/tgp.tgz>

Problem	TPSYS	TGP
tower2	4	40
tower3	8	241
gripper2	4	40
sussman-anomaly	339	-
att-log0	39	40
att-log1	45	200
briefcase-2b-2c	5	251
briefcase-2b-1c	998	-
tgp-AB-p	4	50
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 4.** Results of comparison between TPSYS and TGP (times are in milliseconds)

## 5 Conclusions and Future Work

In this paper we have presented TPSYS, a system for dealing with temporal planning problems. TPSYS arises as a combination of the ideas of **Graphplan** and TGP to solve temporal planning problems, finding optimal plans and avoiding the complex reasoning performed in TGP.

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. Our experiences and the obtained results show the appropriateness of TPSYS for temporal planning problems.

As commented above, TGP presents a mutual exclusion reasoning very valuable for dealing with actions and propositions on a **Graphplan**-based approach, but unfortunately this complex process may make large problems become intractable. In fact, in our tests TGP was unable to solve a simple problem on the *Briefcase* domain with two books to transport and only one briefcase (which TPSYS was able to solve). Unfortunately, performance of TPSYS degrades in problems with limited resources.

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.

Another objective of our current research is the inclusion of resource abstraction as proposed by Srivastava in [13]. Under this proposal, the *TG* will

be generated assuming abstract resources, which are always available, avoiding instantiation over particular resources and consequently reducing the size of the *TG*. A special module for reasoning about resources will be included into the *PE* stage to select the specific resource for each action to be planned.

## References

1. Blum, A.L. and M.L. Furst. "Fast Planning through Planning Graph Analysis," *Artificial Intelligence*, 90:281–300 (1997).
2. Currie, K. and A. Tate. "O-Plan: the Open Planning Architecture," *Artificial Intelligence*, 52(1):49–86 (1991).
3. Dechter, R., et al. "Temporal Constraint Networks," *Artificial Intelligence*, 49:61–95 (1991).
4. El-Kholy, A. and B. Richards. "Temporal and Resource Reasoning in Planning: the parcPLAN Approach." *Proc. 12th European Conference on Artificial Intelligence (ECAI-96)*. 614–618. 1996.
5. Garrido, A. and F. Barber. "Integrating Planning and Scheduling," *Applied Artificial Intelligence*, 15(5):471–491 (2001).
6. Ghallab, M. and H. Laruelle. "Representation and Control in IxTeT, a Temporal Planner." *Proc. 2nd Int. Conf. on AI Planning Systems*. 61–67. Hammond, 1994.
7. Hoffmann, J. and B. Nebel. "The FF Planning System: Fast Plan Generation Through Heuristic Search," *Journal of Artificial Intelligence Research (to appear)* (2001).
8. Köehler, J. "Planning under Resource Constraints." *Proc. 15th European Conf. on AI (ECAI-98)*. 489–493. 1998.
9. Muscettola, N. "HSTS: Integrating Planning and Scheduling." *Intelligent Scheduling* edited by M. Zweben and M.S. Fox, 169–212, San Mateo, CA: Morgan Kaufmann, 1994.
10. Penberthy, J. and D. Weld. "Temporal Planning with Continuous Change," *Proc. 12th Nat. Conf. on AI* (1994).
11. Smith, D.E., et al. "Bridging the Gap Between Planning and Scheduling," *Knowledge Engineering Review*, 15(1) (2000).
12. Smith, D.E. and D.S. Weld. "Temporal Planning with Mutual Exclusion Reasoning." *Proc. 16th Int. Joint Conf. on AI (IJCAI-99)*. 326–337. 1999.
13. Srivastava, B. and S. Kambhampati. "Scaling up Planning by Teasing out Resource Scheduling." *Proc. European Conference of Planning (ECP-99)*, edited by S. Biundo and M. Fox. 172–186. Springer, 1999.