On Managing Temporal Information for Handling Durative Actions in LPG

Alessandro Saetti
Dipartimento di Elettronica per l'Automazione, Università degli Studi di Brescia
Via Branze 38, I-25123 Brescia, Italy
saetti@ing.unibs.it

Abstract

This paper presents how LPG manages ordering constraints for full handling of durative actions introduced by the recent standard language PDDL2.1. LPG is a domain-independent planner that took part in the third International Planning Competition (Toulouse, 2002) showing excellent performance.

The current version of our planner is based on a graph-based representation called "Temporal Durative Action Graphs" (TDA-graphs). TDA-graphs are an extension of a plan representation used in the competition version of LPG, and they improve the management of temporal information required to handle PDDL2.1 durative actions.

Introduction

Local search is emerging as a powerful method to address domain-independent planning. A first version of LPG, presented in (Gerevini & Serina 1999; 2002) that handled only STRIPS version, uses local search in the space of *action graphs* (A-graphs), particular subgraphs of the planning graph representation (Blum & Furst 1997). In this paper we present some extensions that were implemented in the last version of LPG to handle the domains specified in the recent PDDL2.1 language (Fox & Long 2001) supporting "durative actions". In the 3rd IPC, our planner showed excellent performance on a large set of problems in terms of both speed to compute the first solution and quality of the best solution that can be computed by the incremental process.

In this paper we focus on two extensions of LPG concerning the use of *temporal durative action graphs* (TDA-graphs), instead of simple TA-graphs introduced in (Gerevini, et al 2003), and the temporal management of domains involving durative actions. TDA-graphs are an extension of TA-graphs and they allow a better representation of temporal information to handle durative actions. Like in a TA-graph, the TDA-graph action nodes are marked with temporal values estimating the earliest time when the corresponding action terminates. Similarly, a fact node is marked with a temporal value estimating the earliest time when the corresponding fact becomes true. A set of ordering constraints is maintained during search to handle mutually exclusive actions, and to take into account the "causal" relations in the current plan. In the previous version of LPG all mutex actions were treated in the same way by imposing an appropriate ordering constraint. In the new version TDA-graphs can handle different type of mutex actions (depending on the type of precondition/effect that make a pair of action mutex). Different type of mutex are handled by imposing different types of ordering constraints, which can lead to plans of better quality, i.e., of shorter makespan.

The second section presents the plan representation used by the last version of our planner (LPG1.1) for PDDL2.1 domains involving durative actions; the third section describes temporal management, and in particular we describe different types of ordering constraints between durative actions and how these orderings affect the temporal values associated with the facts and actions of the TDA-graph; the fourth section gives the results of an experimental analysis using domains of the 3rd IPC that involve durative actions; finally, the fifth section gives conclusions and mentions further work.

Plan Representation

Our graph-based representation for temporal plans with durative actions is based on *action graphs* (Gerevini & Serina 1999; 2002), i.e. particular subgraphs of the planning graph representation. In the following we will assume that the reader is familiar with the planning graph representation (Blum & Furst 1997) and with the related terminology. Given a planning graph $\mathcal{G}$ for a planning problem, it's possible to assume that the goal nodes of $\mathcal{G}$ in the last level represent the preconditions of a special action a_{end}, which is the last action in any valid plan, while the fact nodes of the first level represent the effects of a special action a_{start}, which is the first action in any valid plan.

An *action graph* (A-graph) for $\mathcal{G}$ is a subgraph $\mathcal{A}$ of $\mathcal{G}$ such that, if a is an action node of $\mathcal{G}$ in $\mathcal{A}$, then also the fact nodes of $\mathcal{G}$ corresponding to the preconditions and positive effects of a are in $\mathcal{A}$, together with the edges connecting them to a. An action graph can contain some *inconsistencies*, i.e., an action with precondition nodes that are not *supported*, or a pair of action nodes involved in a *mutex relation*.[1] In general, a goal g or a precondition node q at a level i is supported in an action graph $\mathcal{A}$ of $\mathcal{G}$ if either (i) in $\mathcal{A}$ there is an action node at level $i-1$ representing an action with (positive) effect q, or (ii) $i=1$ (i.e., q is a proposition of the initial state). An action graph without inconsistencies represents a valid plan and is called *solution graph*. A *solution graph* for $\mathcal{G}$ is an action graph $\mathcal{A}_s$ of $\mathcal{G}$ where all precondition nodes of its action nodes are supported, and there is no mutex relation between its action nodes.

The version of LPG, that took part in the 3rd IPC, uses a

[1] LPG considers only pairs of actions that are *globally mutex* (Gerevini & Serina 2003), i.e. that hold at every level of $\mathcal{G}$.

particular subset of the action graphs, called *linear action graphs with propagation*. A *linear action graph with propagation (LA-graph)* of $\mathcal{G}$ is an A-graph of $\mathcal{G}$ in which each action level contains at most one action node and any number of no-op nodes, and such that if a is an action node of $\mathcal{A}$ at level l, then, for any positive effect e of a and any level $l' > l$ of $\mathcal{A}$, the no-op of e at level l' is in $\mathcal{A}$, unless there is another action node at a level l'' ($l < l'' \leq l'$) which is mutex with the no-op.[2]

The current version of LPG can handle levels 2 and 3 of PDDL2.1. Level 2 introduces numerical quantities and level 3 a new model of actions, called *durative actions*, that allow a higher parallelism among the actions in the plan (Fox & Long 2001). This paper focuses on the representation of durative actions. We indicate with $Cond_S(a)$, $Cond_O(a)$ and $Cond_E(a)$ the `at start`, `over all` and `at end` conditions, respectively, of an action a; with $Eff_S(a)$ and $Eff_E(a)$ the `at start` and `at end` effects, respectively, of a; with $Add_S(a)/Del_S(a)$ the `at start` additive/delete effects of a, and with $Add_E(a)/\ Del_E(a)$ the `at end` additive/delete effects of a.

In simple STRIPS domains, the additive effects of an action a at a level l are represented by fact nodes at the level $l+1$, and its preconditions by fact nodes at the level l. For PDDL2.1 domains involving durative actions, in order to represent the facts that become true after the beginning of the action a, we could introduce a third level between l and $l+1$. Instead, LPG uses no-op nodes to represent the state of the world during the execution of an action. The `at start` additive/delete effects of a at level l, $Add_S(a)/Del_S(a)$, are achieved after the beginning of a, and so LPG introduces/removes the corresponding no-op nodes at level l of $\mathcal{A}$. The `at end` additive/delete effects of a, $Add_E(a)/Del_E(a)$, are achieved at the end of a, and so they do not affect any no-op node at level l; LPG introduces/removes the fact nodes of $Add_E(a)/Del_E(a)$ at level $l+1$ of $\mathcal{A}$. The `at start` conditions of a, $Cond_S(a)$, must be achieved at the beginning of a, and so LPG verifies that the corresponding fact nodes at level l are supported in $\mathcal{A}$. The `over all` conditions of a, $Cond_O(a)$, must be achieved during the full duration of the execution of a, and so LPG verifies that the corresponding no-op nodes are supported at level l. The `at end` conditions of a, $Cond_E(a)$, must be true at the end of a; more precisely, they must be achieved after the `at start` effects become true and before the `at end` effects become true. Therefore, as for the `over all` conditions, LPG checks that the no-op nodes corresponding to the `at end` conditions are supported at level l. The difference between the `over all` and `at end` conditions consists in a different temporal management. This additional way of using the no-op nodes in action graphs leads to the definition of a new class of action graphs called *durative action graphs*.

Definition 1 *A* **durative action graph** *(DA-graph) for $\mathcal{G}$ is a linear subgraph $\mathcal{A}$ of $\mathcal{G}$ such that, if a is an action node of $\mathcal{G}$ in $\mathcal{A}$, then also the fact nodes of $\mathcal{G}$ corresponding to the* `at start` *conditions and* `at end` *effects and the no-op nodes of $\mathcal{G}$ corresponding to the* `over all`, `at end` *conditions and* `at start` *effects of a are in $\mathcal{A}$, together with the edges connecting them to a.*

[2]As noted in (Gerevini, et al 2003), having only one action in each level of a LA-graph does not prevent the generation of parallel (partially ordered) plans.

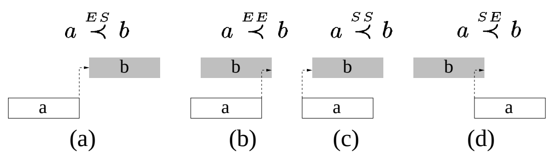

$a \overset{ES}{\prec} b$
$a \overset{EE}{\prec} b$
$a \overset{SS}{\prec} b$
$a \overset{SE}{\prec} b$
b
a
(a)
(b)
(c)
(d)

Figure 1: Types of ordering constraints between durative actions.

We note that the newer version of LPG represents durative actions by modifying the original structure of the planning graph $\mathcal{G}$, i.e., by introducing an edge from an action node to a no-op node at the same level of the graph to represent an `at start` effect, and by introducing an edge from a no-op node to an action node to represent an `at end` and `over all` condition.

In order to represent the temporal information associated with the end points of an action, our planner (i) assigns real values to action, fact and no-op nodes of the DA-graph, and (ii) uses a set Ω of ordering constraints between action nodes. The value associated with a fact or no-op node f represents the (estimated) earliest time at which f becomes true, while the value associated with an action node a represents the (estimated) earliest time when the execution of a can terminate. Obviously, the value associated with a_{start} is zero, while the value associated with a_{end} represents the makespan of the current plan. This assignment of real values to the durative action graph nodes leads to the representation used by the newer version of LPG to handle durative actions called *temporal durative action graph*.

Definition 2 *A* **temporal durative action graph** *(TDA-graph) of $\mathcal{G}$ is a triple $\langle \mathcal{A}, \mathcal{T}, \Omega \rangle$ where $\mathcal{A}$ is a durative action graph with propagation; $\mathcal{T}$ is an assignment of real values to the fact, no-op and action nodes of $\mathcal{A}$; Ω is a set of ordering constraints between action nodes of $\mathcal{A}$.*

Temporal Management for Durative Actions

In this session we discuss which types of ordering constraints are stored in Ω and how, in conformity with Ω, the temporal values assigned by $\mathcal{T}$ are computed.

Ordering Constraints

The original planning graph representation (Blum & Furst 1997) imposes the global constraint that, for any action a at any level l of the graph, every action at the following level starts after the end of a. If we remove this constraint the order of the graph levels should not imply by itself any ordering between actions; the orderings between actions are only those into set Ω. The TA-graphs (Gerevini, et al 2003) respect this assumption. So, it's easy to see that TA-graphs allow to improve the parallelism of the plan with respect to the original planning graph representation.

The ordering constraints in the TA-graphs are of two types: constraints between actions that are implicitly ordered by the causal structure of the plan ($\prec_C$*-constraints*), and constraints imposed by the planner to deal with mutually exclusive actions ($\prec_E$*-constraints*). $a \prec_C b$ belongs to Ω if and only if a is used to achieve a condition node of b in $\mathcal{A}$, while $a \prec_E b$ (or $b \prec_E a$) belongs to Ω only if a and b are mutually exclusive in $\mathcal{A}$. If a and b are exclusive, the planner appropriately imposes either $a \prec_E b$ or $b \prec_E a$. LPG chooses $a \prec_E b$ if the level of a precedes the level of b, $b \prec_E a$ otherwise. Under this assumption on the "direction"

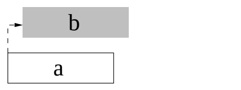
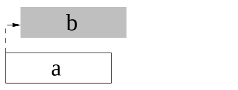
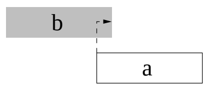
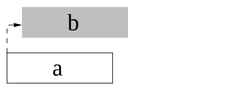
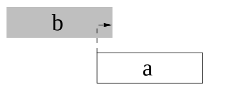
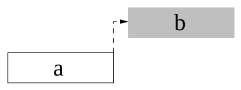
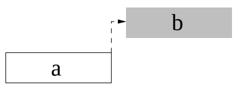
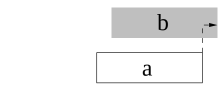
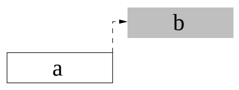
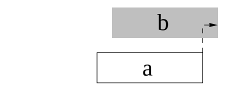
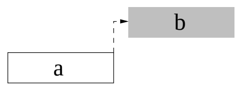
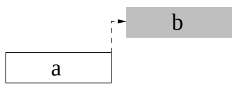
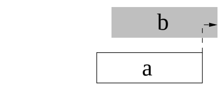
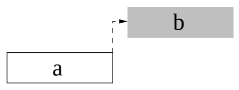
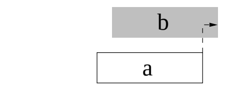
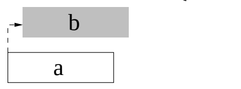
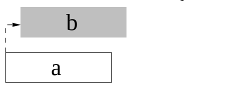
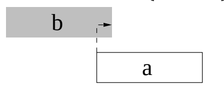
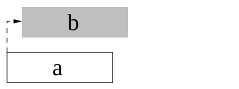
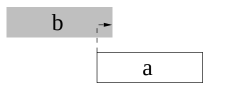
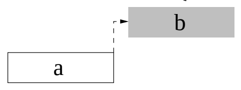
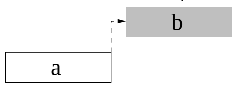
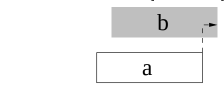
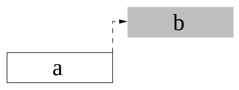
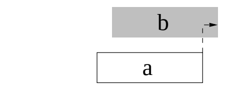
$a \prec b$	$Cond_S(b)$	$Cond_O(b)$	$Cond_E(b)$	$Eff_S(b)$	$Eff_E(b)$
$Cond_S(a)$	1 $\prec_E$ 	2 $\prec_E$ 	3 $\prec_E$ 	4 $\prec_E$ 	5 $\prec_E$ 
$Cond_O(a)$	6 $\prec_E$ 	7 $\prec_E$ 	8 $\prec_E$ 	9 $\prec_E$ 	10 $\prec_E$ 
$Cond_E(a)$	11 $\prec_E$ 	12 $\prec_E$ 	13 $\prec_E$ 	14 $\prec_E$ 	15 $\prec_E$ 
$Eff_S(a)$	**16** $\{\prec_C, \prec_E\}$ 	17 $\{\prec_C, \prec_E\}$ 	18 $\{\prec_C, \prec_E\}$ 	19 $\prec_E$ 	20 $\prec_E$ 
$Eff_E(a)$	21 $\{\prec_C, \prec_E\}$ 	22 $\{\prec_C, \prec_E\}$ 	23 $\{\prec_C, \prec_E\}$ 	24 $\prec_E$ 	25 $\prec_E$ 

Table 1: Ordering constraints between the durative actions a and b, according to the possible casual relations ($\prec_C$) and to the *mutex* between conditions and effects of the durative actions ($\prec_E$). The label $\prec_E$, marking an entry of the table, indicates that at least a proposition of the set associated with the raw of the entry is *mutex* with at least a proposition of the set associated with the column of the entry. The label $\prec_C$, marking an entry of the table, indicates that at least one proposition of the set associated with the raw of the entry supports a proposition of the set associated with the column of the entry.

in which $\prec_E$-constraints are imposed, it is easy to see that the levels of a TA-graph correspond to a topological order of the actions in the represented plan satisfying every ordering constraint in Ω.[3] In the TA-graphs, an ordering constraint $a \prec b$ in Ω (where "$\prec$" stands for $\prec_C$ or $\prec_E$) states that the beginning of b comes after the end of a. Our planner schedules actions in a way that the execution of an action is anticipated as soon as possible; and so $a \prec b$, means that b starts immediately after the end of a. Note that LPG generates partial order plans; so, the ordering strategy illustrated in this section is only one possible policy of scheduling the actions.

In PDDL2.1 domains with durative actions, LPG uses TDA-graphs. In the TDA-graphs, if $\Omega \models a \prec b$, it's possible that a and b overlaps; so, we can distinguish other types of constraints in accordance with the possible casual relations ($\prec_C$) and with the *mutex* between conditions and effects ($\prec_E$). Two ordered actions a and b can overlap in four different ways. Any way represents a different ordering constraint; so, we distinguish between the following four ordering constraints (see figure 1):[4]

a. b can't start before the end of a (denoted by $a \overset{ES}{\prec} b$);

b. b can't end before the end of a (denoted by $a \overset{EE}{\prec} b$);

c. b can't start before the beginning of a (denoted by $a \overset{SS}{\prec} b$);

d. b can't end before the beginning of a (denoted by $a \overset{SE}{\prec} b$).

Instead, we note that in the TA-graphs the ordering between a and b is only of type $a \overset{ES}{\prec}_C b$ or $a \overset{ES}{\prec}_E b$.

Table 1 shows all possible situations that generate in Ω one of the ordering constraints of figure 1. For example, the sixteenth entry of the table 1 shows that $a \overset{SS}{\prec} b \in \Omega$ if (i) at least an effect of type `at start` of a supports a condition of type `at start` of b ($\overset{SS}{\prec}_C$*-constraints*), or (ii) at least an effect of type `at start` of a is mutex with a condition of type `at start` of b ($\overset{SS}{\prec}_E$*-constraints*).

In general, if $a \prec b$, there is at least an ordering constraint in Ω between a and b of type $\overset{EE}{\prec}$, $\overset{ES}{\prec}$, $\overset{SE}{\prec}$, or $\overset{SS}{\prec}$. If there are more than one ordering constraints between a and b, it's possible to simplify Ω by removing all ordering constraints between a and b except the strongest one, i.e. the constraint for which the execution of b is most delayed. The strongest constraint is $\overset{ES}{\prec}$, because $a \overset{ES}{\prec} b$ imposes that the execution of a can not overlap the execution of b, i.e. b can not start before the end of a. The weakest constraint is $\overset{SE}{\prec}$, because $a \overset{SE}{\prec} b$ imposes that b ends after the beginning of a; so b, can start before the beginning of a. Note that the strongest constraint between $\overset{SS}{\prec}$ and $\overset{EE}{\prec}$ depends on the durations of the actions involved. In particular, if the duration of b is longer than the duration of a, the constraint $\overset{SS}{\prec}$ is stronger than $\overset{EE}{\prec}$; if the duration of b is shorter than the duration of a, then the constraint $\overset{EE}{\prec}$ is stronger than $\overset{SS}{\prec}$ (see figure 2). If a

[3] The $\prec_C$ and $\prec_E$*-constraints* are distinguished only for clarity of presentation; but LPG manages both in the same way.

[4] There is a fifth possible ordering constraint between a and b; e.g., if $dur(b) > dur(a)$, it is possible that a supports an `at end` condition of b and b supports or deletes an `at start` condition of a, but actually LPG does not consider it.

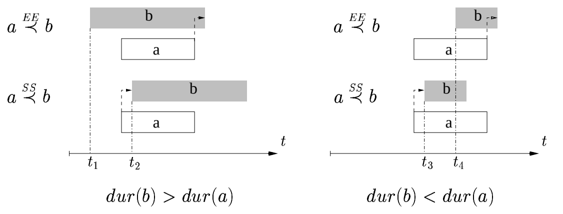


Figure 2: The strongest ordering constraints between $\overset{SS}{\prec}$ and $\overset{EE}{\prec}$ depending on the durations of the actions a and b.

and b have the same duration, the constraint $\overset{SS}{\prec}$ is as strong as $\overset{EE}{\prec}$. Since the duration of an action can depend on the state of the world in which the action is applied, if $a \overset{ES}{\prec} b \notin \Omega$ but $a \{\overset{SS}{\prec}, \overset{EE}{\prec}\} b \in \Omega$, then both $a \overset{SS}{\prec} b$ and $a \overset{EE}{\prec} b$ must be kept in Ω and evaluated at "runtime".

Temporal Values of TDA-graph Nodes

The constraints stored in Ω are useful to compute the temporal values of the TDA-graph action nodes. We denote with $Time(x)$ the temporal value assigned by $\mathcal{T}$ to a node x. In the TA-graphs, LPG computes the temporal value of an action b by simply examining the maximal values over the temporal values of the actions a in $\mathcal{A}$ that must precede b according to Ω:

$$Time(b) = \max_{a \prec b \in \Omega} \{Time(a),\, 0\} + dur(b) + \epsilon.^5$$

If there is no action node that must precede a according to Ω, then b can not start before zero; so, $Time(b)$ is set to the duration of b.[6]

In the TDA-graphs, LPG considers all types of ordering constraints previously introduced, and so the temporal value of an action node b is computed according to the following definition of $Time(b)$:

$$Time(b) = \Bigg[\max \Bigg\{ \max_{a \overset{SE}{\prec} b \in \Omega} \{Time(a) - dur(a)\} - dur(b), \max_{a \overset{ES}{\prec} b \in \Omega} \{Time(a)\}, \max_{a \overset{EE}{\prec} b \in \Omega} \{Time(a)\} - dur(b), \max_{a \overset{SS}{\prec} b \in \Omega} \{Time(a) - dur(a)\},\, 0 \Bigg\} \Bigg] + dur(b) + \epsilon.$$

The term into square brackets represents the earliest temporal value at which the execution of b can start, in accordance with the ordering constraints of type $\overset{SE}{\prec}$, $\overset{ES}{\prec}$, $\overset{EE}{\prec}$ and $\overset{SS}{\prec}$ involving b that are present in the current TDA-graph.

The temporal values of TDA-graph action nodes are used to compute temporal values of TDA-graph fact and no-op nodes. If a fact f is supported by more actions, LPG considers the temporal value of the action that supports f earlier. In TA-graphs, LPG computes the temporal value of a fact node f by simply examining the minimum values over the temporal values of the actions a in $\mathcal{A}$ that support f:

$$Time(f) = \min_{a \in \Lambda(f)} \{Time(a)\},$$

where $\Lambda(f)$ is the set of TA-graph action nodes that support the fact node f. In TDA-graphs, LPG distinguishes the cases in which f is supported at the beginning or at the end of an action, and so the temporal value of a fact node f is computed according to the following definition of $Time(f)$:

$$Time(f) = \min \Bigg\{ \min_{a \in \Lambda_E(f)} \{Time(a)\}, \min_{a \in \Lambda_S(f)} \{Time(a) - dur(a)\} \Bigg\},$$

where $\Lambda_E(f)$ and $\Lambda_S(f)$ are the sets of the TDA-graph action nodes that support f at the end and at the beginning, respectively. In planning problems for which it is important to minimize the makespan of the plan, LPG uses these temporal values to guide the search toward a direction that improves the quality of the plan; in particular they are used to estimate a temporal value at which a condition not supported could become supported (Gerevini, et al 2003).

Figure 3 gives an example of a portion of a TDA-graph containing four action nodes ($a_{1\ldots3,\,start}$) and several fact

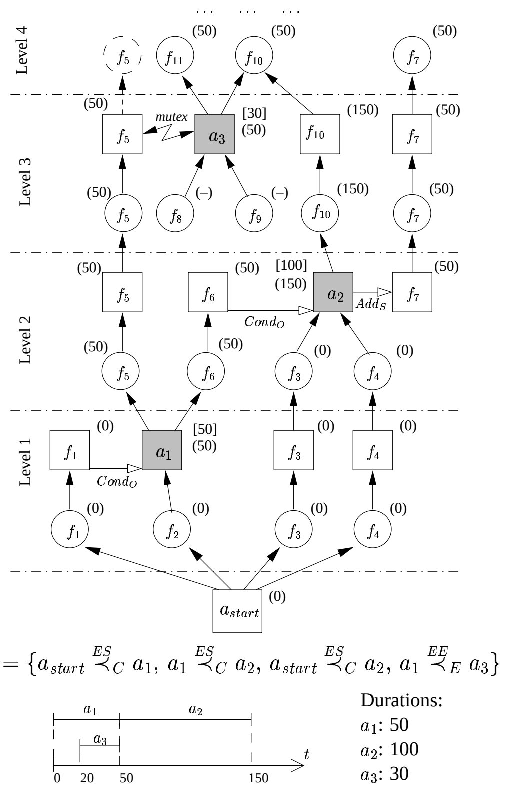


Figure 3: A portion of a TDA-graph. The edges from no-op nodes to action nodes and from action nodes to no-op nodes represent the `over all` and `at start` conditions, respectively. Round brackets contain temporal values assigned by $\mathcal{T}$ to the fact nodes (circles) and the action nodes (squares). The Square-nodes marked with facts are no-ops. The numbers in square brackets represent action durations. "(–)" indicates that the corresponding fact node is not supported.

[5]In order to respect the ordering constraint, LPG introduces an epsilon ($\epsilon > 0$) between the actions involved.

[6]In order to give the better estimate of the temporal value at which an action terminates, if a condition is not supported, instead of zero, LPG estimates the earliest temporal value at which the corresponding proposition becomes true, as described in (Gerevini, et al 2003).

Domain	Problems solved	LPG-speed better	LPG-quality better	LPG-speed worse	LPG-quality worse
Simple-time					
`Depots`	22 (11)	19 (86.4%)	10 (90.9%)	3 (13.6%)	1 (9.1%)
`DriverLog`	20 (16)	18 (90%)	15 (93.8%)	2 (10%)	1 (6.2%)
`Rovers`	20 (10)	16 (80%)	10 (100%)	4 (20%)	0 (0%)
`Satellite`	20 (19)	17 (85%)	19 (100%)	2 (10%)	0 (0%)
`ZenoTravel`	18 (16)	17 (85%)	15 (93.8%)	1 (5%)	1 (6.2%)
`Total`	98(73.5)%	85.2%	95.8%	11.7%	4.2%
Time					
`Depots`	21 (11)	14 (63.6%)	10 (90.9%)	7 (31.8%)	1 (9.1%)
`DriverLog`	20 (16)	19 (95%)	16 (100%)	1 (5%)	0 (0%)
`Rovers`	20 (12)	19 (95%)	12 (100%)	1 (5%)	0 (0%)
`Satellite`	20 (20)	18 (90%)	20 (100%)	1 (5%)	0 (0%)
`ZenoTravel`	20 (20)	20 (100%)	15 (75%)	0 (0%)	5 (25%)
`Total`	99(77.5)%	88.2%	92.4%	9.8%	7.6%
Complex					
`Satellite`	20 (17)	19 (95%)	17 (100%)	1 (5%)	0 (0%)
`Total`	98.7(75)%	87.5%	94.6%	10.2%	5.4%

Table 2: Summary of the comparison of LPG1.1 and SuperPlanner in terms of: number of problems solved by LPG (2nd column) and the SuperPlanner (in brackets); problems in which LPG-speed is faster/slower (3rd/5th columns); problems in which LPG-quality computes better/worse solutions (4th/6th columns).

and no-op nodes representing eleven facts ($f_{1...11}$). Since a_1 supports an `over all` condition of a_2, $a_1 \overset{ES}{\prec}_C a_2$ belongs to Ω. $a_1 \overset{EE}{\prec}_E a_3$ belongs to Ω because an `at end` effect of a_1 is mutex with an `at end` effect of a_3. $a_{start} \overset{ES}{\prec}_C a_1 \in \Omega$ because f_1, that is an `over all` condition of a_1, and f_2, that is an `at start` condition of a_1, belong to the initial state. Similarly, $a_{start} \overset{ES}{\prec}_C a_2 \in \Omega$ because f_3 and f_4, that are `at start` conditions of a_2, belong to the initial state. The temporal value assigned to the facts $f_{1...4}$ at the first level is zero, because they belong to the initial state. Since $a_{start} \overset{ES}{\prec}_C a_1$, $Time(a_1)$ is the sum of $Time(a_{start})$ and of the duration of a_1. $Time(a_2)$ is given by the sum of the duration of a_2 and of the maximum over $Time(a_{start})$ and $Time(a_1)$, because $\{a_1, a_{start}\} \overset{ES}{\prec}_C a_2 \in \Omega$. f_{10} is an `at end` effect of a_2; so, the time assigned to f_{10} at level 3 is equal to $Time(a_2)$. f_7 is an `at start` effect of a_2; so, the time assigned to the no-op node f_7 at level 2 is equal to $Time(a_2) - dur(a_2)$ (the beginning of a_2). Since $a_1 \overset{EE}{\prec}_E a_3 \in \Omega$, $Time(a_3)$ is given by the sum of the duration of a_3 and the maximum between zero (because condition f_8 and f_9 are not supported) and $Time(a_1) - dur(a_3)$. f_{11} at level 4 is supported only by a_3 at the end of it; therefore the temporal value associated with f_{11} is equal to $Time(a_3)$. f_{10} at level 4 is supported at the end of a_2 and a_3; since $Time(a_2) > Time(a_3)$, we have that $Time(f_{10})$ at level 4 is equal to $Time(a_3)$.

Experimental Results

In this section we present some experimental results illustrating the efficiency of LPG using the variants of the domains of the 3rd IPC that involve durative actions ("SimpleTime", "Time" and "Complex").[7]

With respect to the previous temporal management of LPG, the newer management (that full handles durative actions) gives some improvements on the quality of the plans. For example, in the "Satellite" domain the duration of the plans generated by LPG using TDA-graphs is on average 10% shorter than the duration of the plans generated using TA-graphs.

In order to derive some general results on the performance of our planner with respect to all the other planners of the 3rd IPC, we have compared the last version of our planner (LPG1.1) with the best results over all the other fully automated planners in terms of CPU-time and plan quality. We will indicate these results as if they were produced by an hypothetical "SuperPlanner" (note, however, that such a planner does not exist). The tests of "SuperPlanner" were conducted on the official machine of the competition, an AMD Athlon(tm) MP 1800+ (1500Mhz) with 1 Gbytes of RAM, while the tests concerning the last version of LPG on a PIII Intel 866 Mhz with 512 Mbytes of RAM, that is slightly slower than the previous machine. The results of LPG correspond to median values over five runs for each problem considered. The CPU-time limit for each run was 5 minutes, after which the termination was forced.

The performance of LPG was tested in terms of both CPU-time required to find a solution (LPG-speed) and quality of the best plan computed, using at most 5 minutes of CPU-time (LPG-quality). The overall results are showed in table 2. LPG-speed is generally faster than the SuperPlanner, and it always solves a larger number of problems. Overall, the percentage of the problems solved by LPG is 98.7%, while those solved by the SuperPlanner is 75%. The percentage of the problems in which our planner is faster is 87.5%, while this percentage for the SuperPlanner is 10.2%. Concerning LPG-quality, the percentage of the problems for which our planner produced a better quality solution is 94.6%, while this percentage for the SuperPlanner is only 5.4%. In particular LPG finds a solution with quality considerably better (at least 50%) in 35.7% of the problems for which both LPG and SuperPlanner find a solution (with some significant differences in `Satellite`), while the SuperPlanner never finds a solution with quality considerably better than LPG.

Conclusions

We have presented a new plan representation and the technique to manage ordering constraints for handling durative actions in PDDL2.1. These techniques are fully implemented and integrated in the last version of LPG. This work has been carried out in collaboration with Alfonso Gerevini and Ivan Serina.

In my Ph.D. studies, that I am conducting as a first year student at the University of Brescia, I intend to continue this work along several directions. One of the most interesting concerns the management of domains with uncertainty and incomplete information.

References

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

Fox, M., and Long, D. 2001. PDDL2.1: An extension to PDDL for expressing temporal planning domain. http://www.dur.ac.uk/d.p.long/competition.html.

Gerevini, A., and Serina, I. 1999. Fast planning through greedy action graphs. In *Proc. of AAAI-99*.

Gerevini, A., and Serina, I. 2002. LPG: A planner based on local search for planning graphs with action costs. In *Proc. of AIPS-02*.

Gerevini, A., Serina, I., Saetti A., Spinoni S. 2003. Local Search for Temporal Planning in LPG. In *Proc. of ICAPS-03*.

Gerevini, A., and Serina, I. 2003. Planning through Stochastic Local Search and Temporal Action Graph s. In *JAIR* (to appear).

[7]For a description of these domains and of the relative variants the reader may see the official web site of the 3rd IPC (`www.dur.ac.uk/d.p.long/competition.html`).