On the Use of Landmarks in LPG

Francesco Benzi, Alfonso E. Gerevini, Alessandro Saetti, and Ivan Serina
University of Brescia, Italy
{f.benzi,alfonso.gerevini,alessandro.saetti,ivan.serina}@unibs.it

Abstract. Domain-Independent planning is notoriously a very hard search problem. In the literature, several techniques for search control have been proposed in the context of various planning formalisms. In particular, Landmark techniques have been widely used in the planning community in order to guide the search process or to define heuristic functions. A Landmark can be defined as a logical expression, consisting of facts or actions, that certainly becomes true in any solution plan for that problem. In this work, we propose the use of Landmarks for the LPG planner, considering different design choices and analysing empirically its impact on the performance of the planner. Preliminary results show that these techniques can effectively improve the performance of LPG, obtaining results comparable with the state-of-the-art planner LAMA.

Introduction

In the last two decades a number of different techniques have been proposed in the planning community in order to find a good quality solution plan for a given planning problem using a reasonable amount of CPU time. In particular, *planning through local search and action graphs* [13, 10, 12, 6, 8] has shown extremely good performances in the context of fully-automated domain-independent planning. This approach is implemented in the well-known LPG planner, which was awarded at two planning competitions [18, 14] and has been widely used by the planning community, both as a reference planner in many experimental studies comparing planner performance and as a module incorporated into other reasoning systems or applications, e.g. [5, 16, 17, 19, 20, 24, 26]. Landmark techniques have been widely used in the planning community in order to guide the search process or to define heuristic functions [25]. A *landmark* is a fact (a condition or property of the world state) or an action that certainly becomes true in any solution plan for the input planning problem. Different kinds of landmarks and orders between landmarks have been proposed and studied in literature. Finding all possible landmarks and relative orders for a planning problem is computationally very hard [15]. This has made researchers conceive only approximate solutions [15], such as methods that find only a subset of the landmarks and orders. Once landmarks have been computed, there are different ways of using them. The two main families of methods proposed in the literature consist in the use of landmarks to produce a heuristic function and the use of landmarks to decompose the planning process in sub planning tasks [25]. In this paper, we investigate the use of landmark techniques in the context of planning through local search and action graphs, and we evaluate experimentally the impact of their use on the performance of LPG. After a description of the search process in LPG we give some basic definitions regarding landmarks and their computation. Then

we describe the use of landmarks in LPG, and we present the experimental analysis we conducted with different options of our implementation of landmark techniques in LPG. Finally we give conclusions and mention future work.

Local Search in LPG

Our framework is based on a local search in the context of the "planning through planning graph analysis", an approach introduced by Blum and Furst [1]. The problem of generating a plan is a search problem where the elements of the search space are particular subgraphs of the planning graph representing partial plans. The local search method of LPG for a planning graph $\mathcal{G}$ [1] of a given problem $\mathcal{P}$ is a process that, starting from an initial subgraph $\mathcal{G}'$ of $\mathcal{G}$ (a partial plan for $\mathcal{P}$), transforms $\mathcal{G}'$ into a solution of $\mathcal{P}$ through the iterative application of some graph modifications that greedily improve the quality of the current partial plan. Each modification is either an extension of the subgraph to include a new action node of $\mathcal{G}$, or a reduction of the subgraph to remove an action node (and the relevant edges).

Adding an action node to the subgraph corresponds to adding an action to the partial plan represented by the subgraph (analogously to remove an action node). At any step of the search process the set of actions that can be added or removed is determined by the **constraint violations** that are present in the current subgraph of $\mathcal{G}$. More precisely, the search space is formed by the *action subgraphs* of the planning graph $\mathcal{G}$, where an action subgraph of $\mathcal{G}$ is defined in the following way:

Definition 1. *An* **action subgraph** $\mathcal{A}$ *of a planning graph* $\mathcal{G}$ *is a subgraph 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 of* $\mathcal{G}$ *connecting them to* a*.*

A *solution subgraph* (a final state of the search space) is defined in the following way:

Definition 2. *A* **solution subgraph** *of a planning graph* $\mathcal{G}$ *is an action subgraph* $\mathcal{A}_s$ *containing the goal nodes of* $\mathcal{G}$ *and such that*

- *all the goal nodes and fact nodes corresponding to preconditions of actions in* $\mathcal{A}_s$ *are supported;*
- *there is no mutually exclusive relation between action nodes.*

The first version of LPG [10, 11] was based on action graphs where each level may contain an arbitrary number of action nodes, as in the usual definition of planning graphs. The following versions of the system [6, 12, 8, 3, 9, 21] used a restricted class of action graphs, called *linear action graphs* (or extensions of them in order to support temporal and numeric information), combined with some additional data structures supporting a more expressive action and plan representation according to the language features of PDDL 2.1 and 2.2 [4, 2].

Definition 3. *A* **linear action graph** *(LA-graph) of* $\mathcal{G}$ *is an action graph of* $\mathcal{G}$ *in which each level of actions contains at most one action node representing a domain action and any number of "no-ops".*

As shown in [6], having only one action in each level of an LA-graph does not prevent the generation of parallel (partially ordered) plans.

The initial LA-graph contains only two special actions a_{start} and a_{end}, where a_{end} is the last action in any valid plan and its preconditions correspond to the goals of the planning problem under consideration; similarly the initial facts represent the effects of the special action a_{start}, which is the first action in any valid plan. Each search step identifies the neighborhood $N(\mathcal{G})$ (successor states) of the current LA-graph $\mathcal{G}$ (search state), which is a set of LA-graphs obtained from $\mathcal{G}$ by adding an action node to $\mathcal{A}$ or removing an action node from $\mathcal{A}$ in an attempt to repair the *earliest* flawed level of $\mathcal{G}$.[1]

The elements in $N(\mathcal{G})$ are evaluated using a *heuristic evaluation function* E [6, 8] consisting of three weighed terms, estimating their additional *search cost*, *execution cost* and *temporal cost*, i.e., the number of search steps required to repair the new flaws introduced, their contribution to the plan quality and their contribution to the makespan of the represented plan, respectively. An element of $N(\mathcal{G})$ with the lowest combined cost is then selected using a "noise parameter" randomizing the search to escape from local minima [6].

Landmark Techniques

Although single fact (atomic formulae) landmarks are the most studied and used kind of landmarks, also single action landmarks are receiving considerable attention nowadays. More complex kinds of landmarks (i.e. conjunction of facts) are not used because deriving them for a planning problem does not give any advantage, in terms of CPU times: the benefits do not outweigh the efforts [23].

Let us concentrate on single fact landmarks. For a given planning problem, initial and goal facts are, by definition, landmarks for that problem. Our commitment is the computation of *causal landmarks*: facts that, for any solution plan, appear as a precondition of an action in the plan [27]. Goal facts are returned too: we can see them as preconditions of a_{end}.

If we want to use landmarks in a planning problem, we do not only need to know which facts are landmarks, we also need to order the landmarks. This order is fundamental to guide the creation of a solution plan: some facts must become true before others in any solution plan. This is exactly why landmarks are really useful. The needed information is thus a graph of landmarks, called *Landmarks Graph* (LG) or *Landmarks Generation Graph* (LGG), where the nodes are the facts and the (directed) arcs are the orders between facts [15]. Depending on the convention used, arcs can go from goal facts to initial facts or vice versa. Any cycle that is eventually generated must be removed before planning starts [15]. The presence of cycles is related, for example, to the recurrence of the "arm free" fact in a "blocks world" planning domain, since "arm free" switches between TRUE and FALSE many times in any solution plan. The use of the information given by cycles in planning is still a research topic [15].

[1] LPG can use several flaw selection strategies that are described and experimentally evaluated in [7]. The strategy preferring flaws at the earliest level of the graph tends to perform better than the others, and so it is used as the default strategy in LPG. More details and a discussion about this strategy are given in the aforementioned paper.

Several different types of landmark orders are defined in literature [15]:

- *Natural* order: in every solution plan Landmark A appears before Landmark B. So A is ordered before B.

- *Necessary* order: in every solution plan, if B is true at a given step, A is true at the previous step.

- *Greedy Necessary* order: in every solution plan, if B is true for the first time in the plan at a given step, then A is true at the previous step. For every successive step where B is true, we know nothing about A.

The previous 3 orders are mandatory, in the sense that they must be satisfied in every solution plan. Instead the next 2 orders are only "suggested": they may help obtain a better solution plan, but it may be that the only way to compute a solution plan is to violate them.

- *Reasonable* order: landmark B is ordered after landmark A if, from a state in which B is true, to make A true it is necessary to destroy B, but after that B will be needed again to reach the goals. So this means that it is reasonable to make A true before B, and not vice versa.

- *Obedient Reasonable* order: if we decide to satisfy reasonable orders, i.e. treat them as mandatory, new "reasonable" orders may arise. These are called obedient reasonable orders.

Regarding the computational complexity of the problem of finding landmarks and relative orders, it has been proved that both the decision problems "Is the fact L a landmark for the given planning problem?" and "Is there an order (of any kind) O between two landmarks A and B of the given planning problem?" are PSPACE-Complete problems [15]. Since every method commonly used for finding Landmarks and orders has polynomial time complexity, these methods are often incomplete or approximate.

There are two families of methods to compute the Landmarks Graph of a given problem. The first includes the method proposed by Hoffmann [15] and its evolutions (for example LAMA). The second is the method proposed by Zhu & Givan. The method proposed by Hoffmann starts from the goal facts, which are landmarks by definition, and for each of them it looks at the first appearance of that fact in the relaxed planning graph. Then, the intersection of the preconditions of all the actions that support that fact in the relaxed planning graph is computed. The facts resulting from the intersection are landmarks, and these new landmarks are ordered before the previous fact landmark. The method continues until no more Landmarks are generated or the initial facts are reached.

The method proposed by Zhu & Givan [27], instead, starts from the initial facts and goes forward on the relaxed planning graph. The key elements of this method are labels associated to facts and actions and represent the list of facts necessary to reach those facts or actionss. The labels are propagated forward along the planning graph. The rule is: for facts, compute the intersection of the content of all the labels that are applied to the actions reaching that fact, then to the resulting list the fact itself must be added; for actions, compute the union of the content of all the labels that are applied to the facts preconditions of that action. Finally, the content of the labels applied to the goal facts is the list of landmarks for the given planning problem. The advantage of Zhu & Givan's method is that it computes all the (causal) landmarks. Furthermore, if not only

the facts but also the actions are propagated, we can compute the Action Landmarks for the given planning problem. The disadvantage is that this method does not specify how to order the computed landmarks. A naive way to order them is to look at the order of appearance of the facts in the labels. However, many superfluous orders are produced in this way. This excess of landmark ordering can in principle increase computation time, however in practice this increase is very limited.

Landmarks in LPG

We implemented two methods for computing the Landmarks Graph. The first is the method proposed by Hoffmann et al., for which we simply imported in LPG the code written by the authors. The second is the method proposed by Zhu & Givan, which we implemented by ourselves through an extension of the existing LPG code.

In the planning process, we used the Landmarks Graph as described in [22]. They proposed a method where the planning problem is divided into sub planning problems, whose concatenated solution gives a global solution to the original problem. This method works by controlling a base planner. During the planning process, when a landmark is satisfied, it is removed (with its edges) from the Landmarks Graph (obviously the Initial Facts are removed from the Landmark Graph at the first step of the planning process). The set of nodes with no other nodes ordered before them is the frontier of the current planning process and corresponds to the set of facts that the system would currently make supported. At every planning step, the Landmarks in the frontier are selected and given to the base planner as a new set of (sub)goal. Then the subplan computed is added to the end of the current (incomplete) plan. When all the landmarks in the graph have been processed, if a solution plan has not been already generated, the base planner is run with the original goals of the problem and the resulting plan is appended to the subplan computed for the processed landmarks. Differently from the work of Hoffmann et al., we do not give the base planner a disjunction of landmarks at every step; instead, we select a single landmark in the previously computed disjunction. The selection is guided by an heuristic function provided by LPG, which is based on the computation of a relaxed plan [6]. Different approaches have been implemented and tested for the selection of a landmark in the frontier, specifically, (1) selection of a random landmark, (2) selection of the landmark with a maximum heuristic value, (3) with a minimum value, and (4) a variant of the third (minimum value) where we prefer landmarks that do not destroy the portion of the relaxed plan needed to reach another landmark in the disjunction. This is done in order to try to limit the destructive interaction between landmarks. Quite interestingly this variant performs extremely well in the logistics domains. This difference with respect to Hoffmann et al.'s method was motivated by the fact that LPG does not work with disjunctive goals, and even if we modified LPG to accept this kind of goals, in the end the planning process of LPG would select a single fact anyway, or a conjunction of facts taken from the disjunctive goal, to be reached. Similarly to Hoffmann et al., if during a planning step more than a single landmark (the selected one) are reached, all of them are removed from the Landmarks Graph.

As previously said, we have implemented and tested three methods to control the insertion of actions in the action graph. The first one allows the local search of LPG to insert actions only at the end of the current partial plan; the second one also allows to insert actions inside the partial plan, instead of only adding actions at the end of the partial plan. The third one allows the local search to both insert and remove actions at any point of the partial plan. This last variant is the most robust one because it allows the search process also to remove previously inserted actions, but is characterised by a larger search space that negatively influences the planner runtime, as observed in teh experimental results section.

Figure 1 gives the pseudo code of our implementation of the landmarks control loop in LPG. In this implementation, we explicitly impose to LPG the next subgoal (chosen among the landmarks in the LG) to be reached.

```
plan = NULL;

while(TRUE)
{
  update_landmarks_graph(initial_state, plan);

  current_subgoal = select_current_landmark();

  plan = LocalSearch(initial_state, plan, current_subgoal);

  if(all_goals_reached(initial_state, plan)) break;
}

return TRUE;
```

Fig. 1. Pseudo code of the landmarks control implementation.

Every run through the loop is a search step handling the creation of the subplan needed to reach a landmark in the Landmarks Graph. First we update the Landmarks Graph to remove those landmarks that have already been reached. Then we select a landmark in the updated Landmarks Graph as the current subgoal. The control loop terminates when all the goals of the problem have been reached: all landmarks that are goals of the problem have been reached and removed from the Landmarks Graph. In this implementation, we simply run the "LocalSearch" routine of LPG to reach the selected landmark. The initial state for LocalSearch is always the initial state of the planning problem. We do this because every call to LocalSearch uses the precomputed partial plan, beginning from the initial state and reaching the previously selected landmark, as a starting point for the local search.

Experimental Analysis

Tests were performed on an Intel(R) Xeon(R) CPU E5-2620 (with an effective 2.00 GHz rating) with 8 GB of RAM. Our tests have been conducted on a series of problems mainly from IPC competitions. Problem domains tested are "Logistics" (IPC2),

Comparative Results			
Planner/Domain	% Sol.	Time (score)	Quality (score)
LLPG ZhuGivan fs3			
logistics IPC2	100.0 %	83.45 (52.17)	602.683 (50.96)
openstacks IPC5	80.0 %	261.59 (12.15)	105.208 (22.52)
storage IPC5	96.6 %	118.09 (25.07)	231.034 (24.01)
TPP IPC5	83.3 %	124.51 (21.01)	125.960 (22.75)
elevators IPC6	100.0 %	3.03 (27.81)	744.833 (23.24)
transport IPC6	83.3 %	266.73 (18.36)	3140.720 (19.61)
elevators IPC7	100.0 %	44.15 (17.66)	326.050 (17.02)
transport IPC7	90.0 %	625.58 (13.36)	423.611 (15.19)
barman IPC7	0 %	900.88 (1.00)	75.000 (0.39)
nomystery IPC7	0 %	1800.00 (0.00)	-1.0 (0.00)
visitall IPC7	25.0 %	13.24 (3.80)	365.200 (3.33)
Total	75.8 %	430.51 (220.06)	565.296 (202.41)
LLPG ZhuGivan fs2			
logistics IPC2	100.0 %	67.41 (57.77)	520.067 (58.99)
openstacks IPC5	90.0 %	49.31 (18.78)	121.407 (25.82)
storage IPC5	93.3 %	126.02 (22.45)	242.893 (22.27)
TPP IPC5	76.6 %	124.01 (16.80)	116.304 (21.20)
elevators IPC6	100.0 %	14.05 (23.79)	658.800 (27.46)
transport IPC6	93.3 %	308.57 (19.01)	3985.786 (22.68)
elevators IPC7	100.0 %	185.91 (13.25)	308.900 (18.23)
transport IPC7	90.0 %	690.72 (13.57)	411.778 (15.34)
barman IPC7	9.5 %	944.86 (1.79)	276.333 (2.11)
nomystery IPC7	10.0 %	3.09 (1.00)	21.000 (0.91)
visitall IPC7	25.0 %	16.08 (3.43)	299.400 (4.00)
Total	78.1 %	487.76 (199.23)	612.357 (227.92)
LLPG ZhuGivan fs1			
logistics IPC2	100.0 %	68.68 (56.97)	557.150 (55.03)
openstacks IPC5	100.0 %	114.47 (28.64)	159.167 (28.84)
storage IPC5	93.3 %	146.59 (21.80)	244.321 (22.09)
TPP IPC5	76.6 %	158.62 (16.31)	116.304 (21.20)
elevators IPC6	100.0 %	6.60 (26.60)	706.833 (25.40)
transport IPC6	93.3 %	301.80 (19.18)	3985.786 (22.68)
elevators IPC7	100.0 %	89.84 (15.94)	313.000 (18.06)
transport IPC7	80.0 %	605.88 (11.91)	396.500 (13.43)
barman IPC7	9.5 %	1041.76 (2.50)	178.000 (3.00)
nomystery IPC7	40.0 %	600.45 (6.46)	28.500 (6.61)
visitall IPC7	25.0 %	11.65 (3.80)	299.400 (4.00)
Total	80.3 %	447.27 (218.22)	625.469 (229.21)
LLPG ZhuGivan fs0			
logistics IPC2	100.0 %	82.10 (51.66)	625.683 (49.12)
openstacks IPC5	80.0 %	250.82 (12.37)	99.542 (23.00)
storage IPC5	93.3 %	133.75 (23.49)	202.964 (24.45)
TPP IPC5	80.0 %	120.35 (18.63)	119.375 (22.03)
elevators IPC6	100.0 %	3.37 (27.71)	703.333 (24.71)
transport IPC6	100.0 %	175.20 (24.76)	4345.367 (25.26)
elevators IPC7	100.0 %	79.29 (16.77)	378.150 (15.00)
transport IPC7	70.0 %	721.08 (10.78)	422.500 (10.17)
barman IPC7	0 %	1800.00 (0.00)	-1.0 (0.00)
nomystery IPC7	15.0 %	663.99 (1.08)	20.667 (1.95)
visitall IPC7	25.0 %	84.14 (2.18)	737.200 (1.62)
Total	76.5 %	424.08 (219.12)	745.541 (204.16)

Table 1. Comparison between landmark selection rules: average values for Coverage, Time and Quality results. IPC score in brackets.

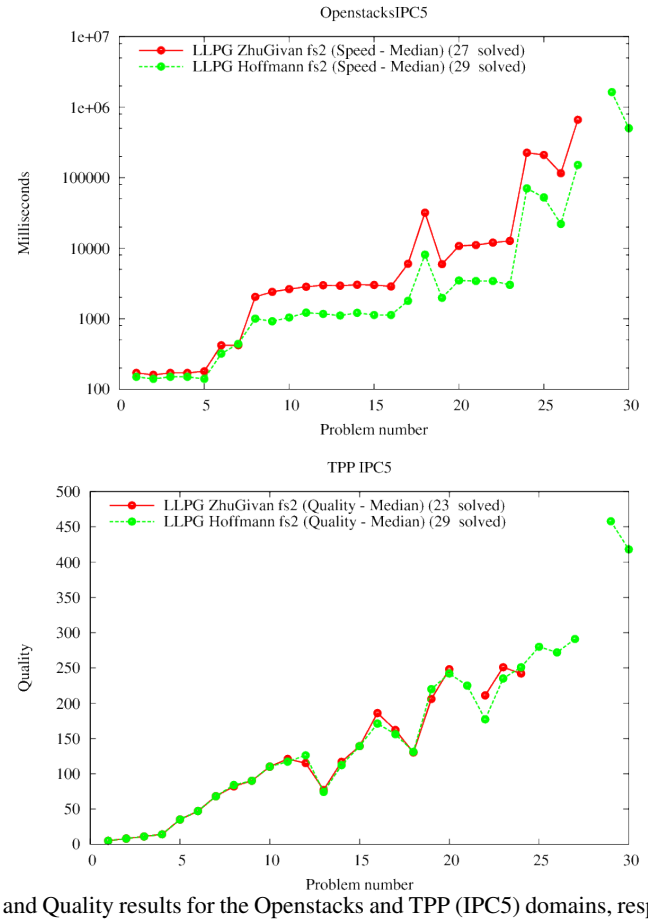

Fig. 2. Speed and Quality results for the Openstacks and TPP (IPC5) domains, respectively. Comparison of using Zhu & Givan's landmarks and Hoffmann's landmarks.

"Transport" (IPC6 and IPC7), "Elevators" (IPC6 and IPC7), "Storage" (IPC5), "Barman" (IPC7), "NoMystery" (IPC7), "Visitall" (IPC7), "Openstacks" (IPC5) and "TPP" (IPC5).

The results are shown in the tables and figures below. For each domain we show runtime (speed) and plan quality results in two separate plots. For each plot, on the x axis we have the different problems from that domain, on the y axis the results. For the "speed" results we used a log10 scale.

In the following comparison tests we will show the results of LPG without landmarks (labelled: LPG) and LPG with landmarks (labelled: LLPG). We will use the label "fs2" to indicate that the selection of a landmark in the disjunction is done using our variant of the "select the one with the minimum heuristic value" rule. The label "test" will indicate that if the landmarks control of LPG fails, the program will retry again and again, until a solution plan is found or the time available expires. Also, ex-

Comparative Results				
Planner/Domain	% Sol.	Time (score)	Quality (score)	LM
LLPG Z&G fs2				
logistics IPC2	100.0 %	67.41 (58.47)	520.1 (59.00)	224.6
openstacks IPC5	90.0 %	49.31 (19.13)	121.4 (25.91)	98.3
storage IPC5	93.3 %	126.02 (26.70)	242.9 (27.00)	0.9
TPP IPC5	76.6 %	124.01 (17.56)	116.3 (21.52)	11.6
elevators IPC6	100.0 %	14.05 (25.92)	658.8 (28.31)	30.7
transport IPC6	93.3 %	308.57 (26.48)	3985.8 (27.00)	0
elevators IPC7	100.0 %	185.91 (16.45)	308.9 (17.88)	68.1
transport IPC7	90.0 %	690.72 (16.70)	411.8 (17.00)	0
barman IPC7	9.5 %	944.86 (3.00)	276.3 (2.58)	14.7
nomystery IPC7	10.0 %	3.09 (1.00)	21.0 (1.00)	14.5
visitall IPC7	25.0 %	16.08 (4.00)	299.4 (3.94)	0
Total	78.1 %	487.76 (223.93)	612.4 (240.02)	463.4
LLPG Hoff. fs2				
logistics IPC2	100.0 %	74.85 (55.21)	520.1 (59.00)	224.6
openstacks IPC5	96.6 %	142.40 (27.98)	147.4 (28.00)	91.3
storage IPC5	93.3 %	134.15 (25.81)	244.3 (26.79)	0.9
TPP IPC5	96.6 %	151.07 (25.93)	157.5 (27.78)	14.1
elevators IPC6	100.0 %	6.87 (28.44)	717.6 (26.39)	24.3
transport IPC6	93.3 %	279.21 (26.59)	3997.6 (26.97)	0
elevators IPC7	100.0 %	121.45 (17.61)	303.4 (18.27)	49.8
transport IPC7	85.0 %	673.96 (15.96)	409.2 (16.00)	0
barman IPC7	0 %	1171.31 (0.51)	73.0 (1.00)	13.6
nomystery IPC7	25.0 %	751.96 (3.43)	25.8 (4.00)	14.8
visitall IPC7	25.0 %	19.91 (3.59)	302.0 (3.90)	26
Total	80.7 %	441.33 (239.59)	624.6 (246.98)	459.4

Table 2. Comparison between Hoffmann's landmarks and Zhu & Givan's landmarks: average values for Coverage, Time, Quality and number of landmarks. IPC score in brackets.

cept differently indicated, the control variant used is the default one: only adding new actions at the end of the partial plan.

In Table 1 we compare LPG with landmarks for all variants of the Landmarks Graph frontier selection methods: selection of a random landmark (labelled: fs0); the selection of the landmark with minimum heuristic value (labelled: fs1); our variant of fs1 designed to avoid destructive interaction between landmarks in the frontier (labelled: fs2); and selection of the landmark with maximum heuristic value (labelled: fs3). The results show that "fs1" is the best selection rule for quality and coverage results. Concerning speed the best rule is "fs3", however the advantage with respect to "fs1" is small. Moreover, "fs2" is particularly effective in Logistics.

In Figure 2 we show speed and quality plots for the comparison between the use of Zhu & Givan's landmarks and the use of Hoffmann's landmarks. As we can see, landmarks computed by Hoffmann's method give better coverage, better time results and better quality results.

Figure 3 plots speed and quality results for the comparison of the landmark selection rules we implemented. The Speed plot shows better coverage and time results for the "fs1" rule, while the Quality plot shows that our variant of "fs1", labelled "fs2", gives, as expected, better quality results in Logistics.

In Table 2 we compare LPG with landmarks in the two variants: using landmarks computed by Hoffmann's method and using landmarks computed using Zhu & Givan's method. The last column contains the average number of Landmarks for every do-

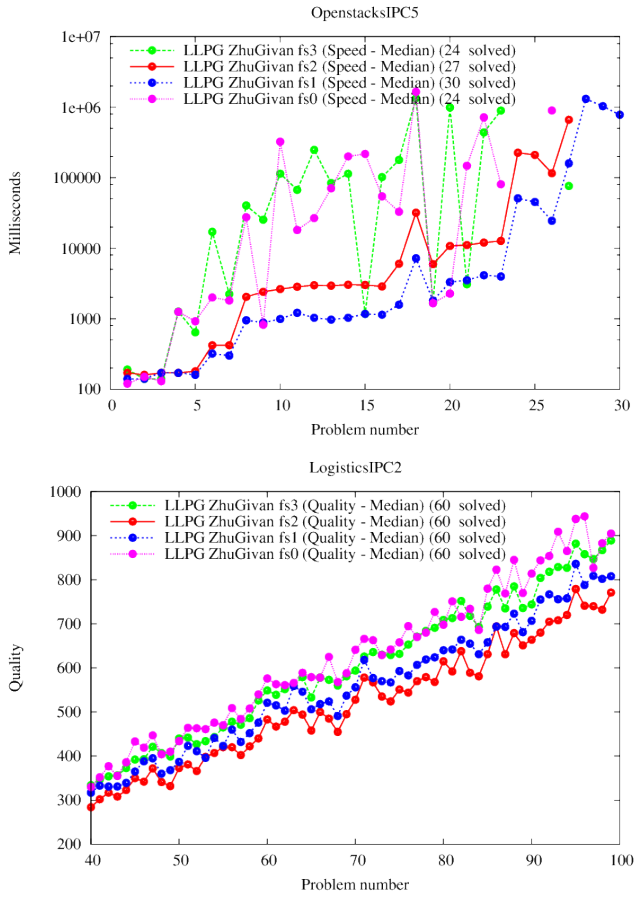

Fig. 3. Speed and Quality results for the Openstacks (IPC5) and Logistics (IPC2) domains, respectively. Comparison of different landmark selection rules.

main (initial and goal Landmarks were not counted); here we can observe that the two approaches produce a similar number of landmarks in the different domains. Unfortunately in Transport and Visitall (Zhu&Givan approach) the system cannot find new landmarks. In general, the use of landmarks computed by Hoffmann's method gives better results; however the difference is usually small.

Table 3 compares LPG with landmarks using three control variants: the one that only adds new actions at the end of the current partial plan (labelled: control 2), the one that can add new actions at any point of the current partial plan (labelled: control 1), and the one that can add as well as remove actions at any point of the partial plan (labelled: control 0). The results show that the best solution is "control 2". However, "control 1" gives best results in IPC5 "TPP" domain, and "control 0" in IPC7 "barman" domain.

Comparative Results			
Planner/Domain	% Sol.	Time (score)	Quality (score)
LLPG ZhuGivan fs2 control 2			
logistics IPC2	100.0 %	67.41 (53.41)	520.067 (56.92)
openstacks IPC5	90.0 %	49.31 (23.91)	121.407 (26.00)
storage IPC5	93.3 %	126.02 (23.00)	242.893 (21.12)
TPP IPC5	76.6 %	124.01 (15.22)	116.304 (19.34)
elevators IPC6	100.0 %	14.05 (27.13)	658.800 (28.11)
transport IPC6	93.3 %	308.57 (22.57)	3985.786 (21.68)
elevators IPC7	100.0 %	185.91 (19.00)	308.900 (19.00)
transport IPC7	90.0 %	690.72 (15.92)	411.778 (17.00)
barman IPC7	9.5 %	944.86 (1.80)	276.333 (2.07)
nomystery IPC7	10.0 %	3.09 (1.00)	21.000 (0.91)
visitall IPC7	25.0 %	16.08 (2.06)	299.400 (4.00)
Total	78.1 %	487.76 (212.33)	612.357 (225.15)
LLPG ZhuGivan fs2 control 1			
logistics IPC2	100.0 %	70.68 (53.40)	508.650 (58.17)
openstacks IPC5	90.0 %	47.75 (23.90)	121.407 (26.00)
storage IPC5	93.3 %	135.79 (22.78)	216.000 (22.98)
TPP IPC5	80.0 %	242.46 (18.30)	113.167 (22.49)
elevators IPC6	86.6 %	295.70 (14.54)	681.231 (20.61)
transport IPC6	80.0 %	385.47 (17.62)	3383.167 (18.39)
elevators IPC7	50.0 %	746.24 (4.01)	480.600 (4.14)
transport IPC7	40.0 %	950.66 (5.07)	686.500 (3.73)
barman IPC7	0 %	1800.00 (0.00)	-1.0 (0.00)
nomystery IPC7	15.0 %	453.40 (1.72)	22.333 (2.00)
visitall IPC7	25.0 %	31.98 (1.77)	428.800 (2.78)
Total	69.1 %	563.73 (192.11)	528.238 (187.22)
LLPG ZhuGivan fs2 control 0			
logistics IPC2	100.0 %	67.01 (57.90)	516.533 (57.30)
openstacks IPC5	16.6 %	.04 (4.00)	25.000 (4.00)
storage IPC5	26.6 %	490.92 (7.00)	17.250 (5.85)
TPP IPC5	56.6 %	359.72 (14.11)	103.647 (11.90)
elevators IPC6	73.3 %	489.85 (14.18)	918.455 (14.49)
transport IPC6	80.0 %	374.24 (17.04)	2811.708 (18.31)
elevators IPC7	20.0 %	666.83 (1.93)	387.250 (1.75)
transport IPC7	45.0 %	841.74 (5.66)	697.556 (3.51)
barman IPC7	14.2 %	779.49 (4.00)	193.250 (4.00)
nomystery IPC7	5.0 %	231.68 (0.00)	20.000 (0.00)
visitall IPC7	20.0 %	361.48 (4.00)	347.500 (3.80)
Total	50.4 %	941.28 (136.59)	417.421 (131.48)

Table 3. Comparison of different control methods: average values for Coverage, Time and Quality results. IPC score in brackets.

The results for domain Elevators (IPC7) in Figure 4 are shown to compare different control methods. As shown in the plots, "control 2" is the best solution, giving best coverage, speed and quality.

In Table 4 we compare LPG with landmarks (LLPG ZhuGivan fs2), LPG without landmarks (LPG) and the state-of-the-art planner LAMA2011 [23]. The experimental tests show that our planner gives the best speed and coverage, while LAMA gives the best quality. However, in assessing the results we must take into consideration that LAMA2011 in some domains (Elevators IPC6 and Transport IPC6) does not solve any problem.

The results for domain Logistics (IPC2) in Figure 5 compare LPG with and without landmarks and LAMA2011. As we can see, only LPG (with and without Landmarks)

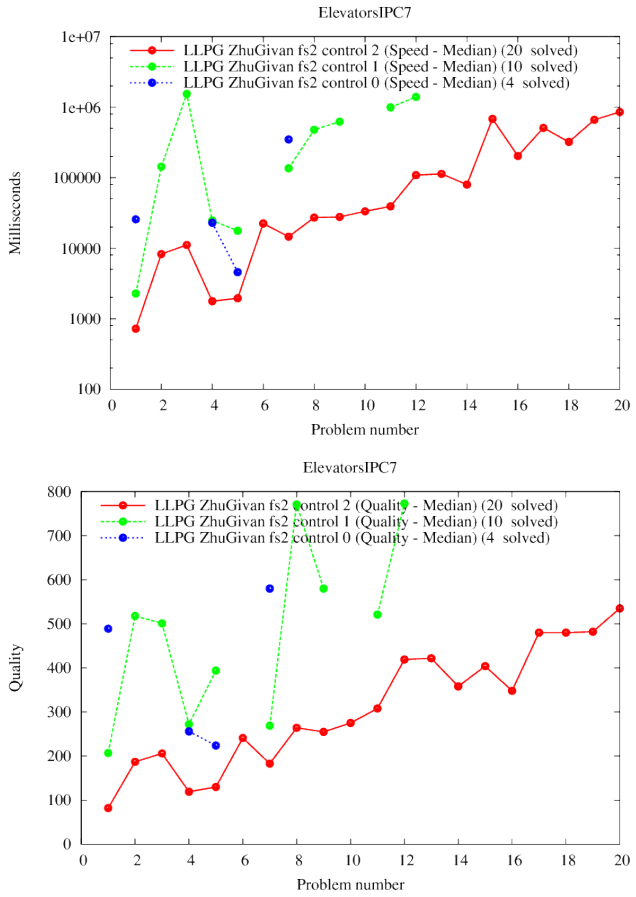

Fig. 4. Speed and Quality results for domain Elevators (IPC7): comparison of different control methods.

solved all the problems. If we consider speed, quality and coverage altogether, the best planner is LPG with landmarks.

Table 5 compares the performances of LLPG versus LPG in terms of delta values of the coverage, the IPC Speed and the IPC Quality scores. The LM column reports the average number of landmarks for every domain (initial and goal landmarks were not counted), while the "# goals" column reports the average number of goals in the different domains. We can see that a high number of landmarks is usually associated with higher performances of LLPG w.r.t. LPG, see for example the `logistics IPC2` domain (where on average we can find 224 landmarks + initial facts + the goal facts) and the `elevators IPC7` domain (where on average we can find 50 landmarks + initial facts + the goal facts). Furthermore, we can also observe performance improvements in the `transport` domains in which the LM value is equal to 0; this is related to the

Comparative Results			
Planner/Domain	% Sol.	Time (score)	Quality (score)
LPG			
logistics IPC2	100.0 %	179.33 (46.27)	666.267 (43.03)
openstacks IPC5	90.0 %	105.73 (17.38)	120.630 (24.98)
storage IPC5	96.6 %	110.74 (26.54)	189.276 (17.91)
TPP IPC5	70.0 %	177.97 (17.44)	123.952 (13.50)
elevators IPC6	90.0 %	442.08 (14.18)	509.296 (23.64)
transport IPC6	53.3 %	448.90 (12.13)	2055.062 (13.19)
elevators IPC7	50.0 %	1005.54 (2.90)	216.300 (7.69)
transport IPC7	5.0 %	1018.84 (0.48)	246.000 (0.48)
barman IPC7	0 %	900.88 (1.00)	74.000 (0.43)
nomystery IPC7	20.0 %	621.51 (1.13)	20.250 (3.00)
visitall IPC7	10.0 %	.52 (1.00)	293.000 (0.64)
Total	63.3 %	780.25 (147.17)	321.550 (155.10)
LLPG			
logistics IPC2	100.0 %	74.85 (56.84)	520.067 (54.81)
openstacks IPC5	96.6 %	142.40 (23.38)	147.379 (26.91)
storage IPC5	93.3 %	134.15 (21.25)	244.321 (13.96)
TPP IPC5	96.6 %	151.07 (18.72)	157.483 (20.46)
elevators IPC6	100.0 %	6.87 (27.54)	717.633 (25.34)
transport IPC6	93.3 %	279.21 (22.36)	3997.607 (21.74)
elevators IPC7	100.0 %	121.45 (16.33)	303.400 (14.36)
transport IPC7	85.0 %	673.96 (11.65)	409.235 (11.12)
barman IPC7	0 %	1171.31 (0.29)	73.000 (0.44)
nomystery IPC7	25.0 %	751.96 (2.42)	25.800 (3.10)
visitall IPC7	25.0 %	19.91 (2.35)	302.000 (4.00)
Total	80.7 %	441.33 (209.72)	624.611 (204.83)
LAMA2011			
logistics IPC2	83.3 %	311.93 (29.53)	441.360 (50.00)
openstacks IPC5	100.0 %	7.85 (22.76)	154.600 (28.99)
storage IPC5	63.3 %	125.06 (8.46)	21.526 (17.92)
TPP IPC5	100.0 %	14.97 (21.69)	116.900 (29.00)
elevators IPC6	0 %	1800.00 (0.00)	-1.0 (0.00)
transport IPC6	0 %	1800.00 (0.00)	-1.0 (0.00)
elevators IPC7	100.0 %	61.10 (17.91)	231.200 (18.29)
transport IPC7	70.0 %	359.93 (12.57)	212.000 (13.00)
barman IPC7	95.2 %	90.85 (20.00)	188.571 (20.00)
nomystery IPC7	60.0 %	144.41 (10.42)	31.417 (9.54)
visitall IPC7	100.0 %	94.02 (18.61)	1483.050 (18.37)
Total	69.1 %	324.14 (205.28)	286.888 (200.61)

Table 4. Percentage of problem solved, CPU time in seconds and Plan Quality (IPC scores in brackets) of LPG, LLPG and Lama 2011.

fact that the LM value does not count the goals as landmarks, although indeed they are landmarks and they are effectively used by LPG with landmarks (LLPG).

Conclusions

In this paper, we have presented some new techniques for planning with landmarks that have been implemented in LPG; the experimental results show significant improvements in terms of both number of problems solved and CPU time. In particular, the use of landmarks for dividing the planning problem into sub planning problems, whose concatenated solution gives a global solution to the original problem, gives extremely interesting results.

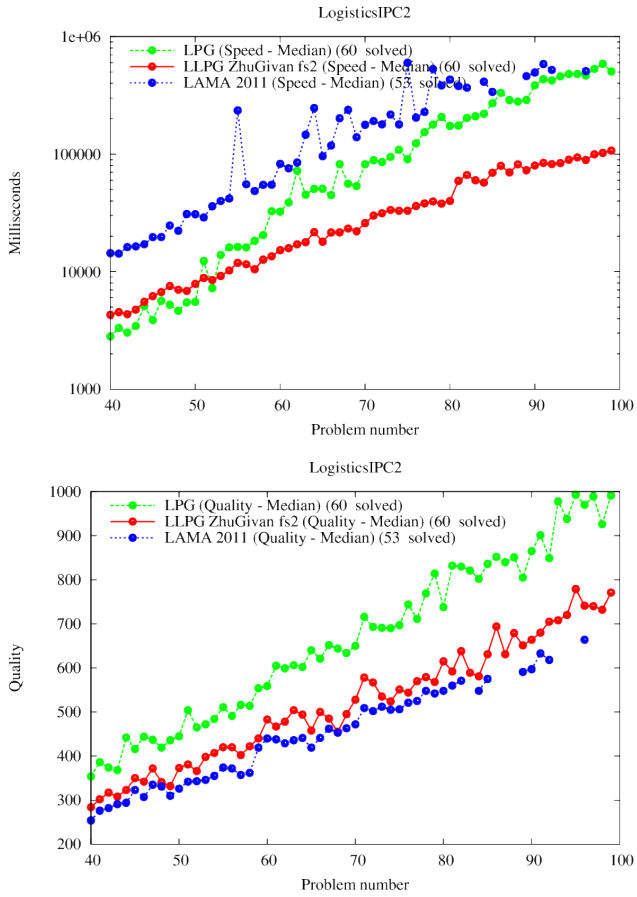

Fig. 5. Speed and Quality results for domain Logistics (IPC2): comparison of different planners.

As future work, we plan to extend LLPG in order to compute landmarks for temporal and metric domains. Moreover, we plan to compute also Action Landmarks and use them to effectively initialize the search process. Exploiting action landmarks seems to be very natural and promising in the context of LPG. Finally, we are developing a new idea about *quasi-landmarks*: facts that appear in almost every solution plan. We expect quasi-landmarks to be useful in domains where the only landmarks that are computed by the existing methods are the initial and goal facts. These domains include, for example, the two domains Transport IPC6 and Transport IPC7 that we used in our experiments.

References

1. Blum, A., Furst, M.: Fast planning through planning graph analysis. Artificial Intelligence 90, 281–300 (1997)

LLPG vs LPG					
Planner/Domain	LM	# goals	△ % Sol.	△ Time score	△ Quality score
logistics IPC2	224.6	69.5	+ 0.0 %	+ 10.6	+ 11.8
openstacks IPC5	91.3	31	+ 6.6 %	+6	+ 1.9
storage IPC5	0.9	7.7	-3.3 %	- 5.3	-4
TPP IPC5	14.1	8.7	+ 26.6 %	+1.3	+7
elevators IPC6	24.3	17	+ 10.0 %	+13.4	+1.7
transport IPC6	0	10.4	+ 40.0 %	+10.2	+8.6
elevators IPC7	49.8	37.6	+ 50.0 %	+13.4	+6.7
transport IPC7	0	18.8	+ 80.0 %	+11.2	+10.6
barman IPC7	13.6	9.3	0 %	-0.7	0
nomystery IPC7	14.8	8.4	+5 %	+1.3	+0.1
visitall IPC7	26	263	+ 15.0 %	+1.3	+3.4
Total	459.4	481.4	+ 17.4 %	+62.6	+49.7

Table 5. LLPG vs LPG in terms on delta values for the coverage, IPC Speed and Quality scores.

2. Edelkamp, S., Hoffmann, J.: PDDL2.2: The language for the classic part of the 4th international planning competition. Technical Report 195, Institut für Informatik, Freiburg, Germany (2004)
3. Fox, M., Gerevini, A., Long, D., Serina, I.: Plan stability: Replanning versus plan repair. In: Proceedings of the 16th International Conference on Automated Planning and Scheduling. AAAI Press, Cumbria, UK (2006)
4. Fox, M., Long, D.: PDDL2.1: An extension to PDDL for expressing temporal planning domains. Journal of Artificial Intelligence Research (JAIR) 20, pp. 61–124 (2003)
5. Gerevini, A., Kuter, U., Nau, D., S., A., S., Waisbrot, N.: Combining domain-independent planning and HTN planning: The Duet planner. In: Proceedings of the Eighteenth European Conference on Artificial Intelligence (ECAI-08) (2008)
6. Gerevini, A., Saetti, A., Serina, I.: Planning through stochastic local search and temporal action graphs. Journal of Artificial Intelligence Research (JAIR) 20, 239–290 (2003)
7. Gerevini, A., Saetti, A., Serina, I.: An empirical analysis of some heuristic features for local search in LPG. In: Proceedings of the Fourteenth International Conference on Automated Planning and Scheduling (ICAPS-04). pp. 171–180. AAAI Press, Menlo Park, CA, USA (2004)
8. Gerevini, A., Saetti, A., Serina, I.: An approach to temporal planning and scheduling in domains with predictable exogenous events. Journal of Artificial Intelligence Research (JAIR) 25, 187–231 (2006)
9. Gerevini, A., Saetti, A., Serina, I.: An approach to efficient planning with numerical fluents and multi-criteria plan quality. Artificial Intelligence 172(8-9), 899–944 (2008)
10. Gerevini, A., Serina, I.: LPG: A planner based on local search for planning graphs with action costs. In: Proceedings of the 6th International Conference on Artificial Intelligence Planning and Scheduling (AIPS-02). pp. 281–290. AAAI Press/MIT Press (2002)
11. Gerevini, A., Serina, I.: Planning as propositional CSP: from Walksat to local search for action graphs. CONSTRAINTS 8(4) (October 2003)
12. Gerevini, A., Serina, I., Saetti, A., Spinoni, S.: Local search techniques for temporal planning in LPG. In: Proceedings of the 13th International Conference on Automated Planning & Scheduling (ICAPS03). pp. 62–71. AAAI Press (2003)
13. Gerevini, A., Serina, I.: Fast plan adaptation through planning graphs: Local and systematic search techniques. In: Proceedings of the 5th International Conference on Artificial Intelligence Planning Systems. AAAI Press, Breckenridge, CO (2000)
14. Hoffmann, J., Edelkamp, S.: The deterministic part of IPC-4: An overview. Journal of Artificial Intelligence Research (JAIR) 24, 519–579 (2005)

15. Hoffmann, J., Porteous, J., Sebastia, L.: Ordered landmarks in planning. J. Artif. Int. Res. 22(1), 215–278 (Nov 2004), http://dl.acm.org/citation.cfm?id=1622487.1622495
16. Jimẽnez, S., Fernãndez, F., Borrajo, D.: The PELA architecture: integrating planning and learning to improve execution. In: Proceedings of the Twenty-Third National Conference on Artificial Intelligence (AAAI-08) (2008)
17. Kolobov, A., Mausam, Weld, D., S.: Determinize, solve, and generalize: Classical planning for MDP heuristics. In: ICAPS-09 Workshop on Heuristics for Domain-independent Planning (2009)
18. Long, D., Fox, M.: The 3rd International Planning Competition: Results and analysis. Journal of Artificial Intelligence Research (JAIR) 20, 1–59 (2003)
19. Morales, L., Castillo, L., Fernandez-Olivares, J., Gonzalez-Ferrer, A.: Automatic generation of user adapted learning designs: An AI-planning proposal. In: Proceedings of the Fifth International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH-08) (2008)
20. Nguyen, T., A., Do, M., B., Kambhampati, D., Srivasta, B.: Planning with partial preference models. In: Twenty-first International Joint Conference on Artificial Intelligence (IJCAI-09) (2009)
21. Nguyen, T.A., Do, M.B., Gerevini, A., Serina, I., Srivastava, B., Kambhampati, S.: Generating diverse plans to handle unknown and partially known user preferences. Artificial Intelligence 190, 1–31 (2012)
22. Porteous, J., Sebastia, L., Hoffmann, J.: On the extraction, ordering and usage of landmarks in planning. In: European Conference of Planning (ECP'01). pp. 37–48 (2001)
23. Richter, S., Westphal, M.: The LAMA planner: Guiding cost-based anytime planning with landmarks. CoRR abs/1401.3839 (2014), http://arxiv.org/abs/1401.3839
24. Srivastava, B., Nguyen, T.A., Gerevini, A., Kambhampati, S., Do, M.B., Serina, I.: Domain independent approaches for finding diverse plans. In: Proceedings of the 20th International Joint Conference on Artificial Intelligence (2007)
25. Vernhes, S., Infantes, G., Vidal, V.: Problem splitting using heuristic search in landmark orderings. In: IJCAI'13. pp. –1–1 (2013)
26. Vrakas, D., Hatzi, O., Bassiliades, N.and Anagnostopoulos, D., Vlahavas, I.: A visual programming tool for designing planning problems for semantic web service composition. In: Visual Languages for Interactive Computing: Definitions and Formalizations (2008)
27. Zhu, L., Givan, R.: Landmark Extraction via Planning Graph Propagation. In *Printed Notes of ICAPS'03 Doctoral Consortium* (June 2003), trento, Italy