STeLLa v2.0: Planning with Intermediate Goals

L. Sebastia, E. Onaindia, E. Marzal

Dpto. Sistemas Informaticos y Computacion
Universidad Politecnica de Valencia
e-mail: {lstarin, onaindia, emarzal}@dsic.upv.es

Abstract. In the last few years, AI planning techniques have experimented a great advance. One of the reasons for this big expansion is the International Planning Competition (IPC), which enforces the definition of language standards as PDDL+ and new benchmarks. In this paper, we present the new features of STeLLa, a planner that participated in the last IPC, held in Toulouse last April. STeLLa is a forward search planner that builds intermediate goals to ease the resolution of the planning problem.

Acknowledgement.This work has been partially funded by the Spanish Government CICYT project DPI2001-2094-C03-03 and by the Universidad Politecnica de Valencia projects UPV-2001-0017 and UPV-2001-0980.

1 Introduction

The field of planning in AI has experimented great advances over the last few years. Problems that seemed unsolvable, can be solved now in a few seconds. Obviously, this motivates the researchers to increase the complexity of the problems to be solved by their planners. This way, some of the domains tested in the last International Planning Competition 2002 (IPC2002) came out as a combination of the domains used in the previous planning competition (AIPS'00), like blocksworld, logistics, etc. thus increasing the difficulty of the problems[1]. Moreover, other domains exhibiting new features such as the use of numerical values or durative actions were also defined.

In this paper, we present STeLLa v2.0, a planner which participated in the STRIPS track at the IPC2002. This version uses a new problem-solving approach that consists of computing the set of subgoals to be achieved at each time step. Solving a planning problem can be stated as successively decomposing the problem into a set of intermediate goals. STeLLa's goal is then to find the (parallel) actions to reach those intermediate goals. That is, STeLLa obtains parallel plans and tries to minimize the overall number of actions in the plan.

Under this new approach, efforts are simply concentrated in obtaining the literals in the next intermediate goal set and then finding the set of actions to

[1] The definition of these new domains and the results obtained by the participating planners in the IPC2002 can be found in www.dur.ac.uk/d.p.long/competition.html.

reach those literals from the previous state. The issue of how to build these intermediate goals is fully detailed in Section 3, but we can anticipate that STeLLa uses *landmarks* graphs (LG) to create those intermediate goals. The idea of using LGs was firstly introduced in [5] and was extended in [6]. In both papers, a landmark is defined as a literal that must be true in every solution plan. The process extracts a set of landmarks which are later ordered under the concept of "reasonable order", obtaining a LG.

This paper is organized as follows. Section 2 summarizes some basic concepts about landmarks and LGs. Our technique to build intermediate goals is explained in Section 3 and how STeLLa uses the obtained intermediate goals in order to find the solution is shown in Section 4. Section 5 gives the results obtained in the IPC2002 and Section 6 concludes by summarizing the strong and weak points of this framework and the future work.

2 Concepts about Landmarks and Orders

Definition 1. *A* **STRIPS action** o *is a triple* $o = \langle \mathsf{Pre}(o), \mathsf{Add}(o), \mathsf{Del}(o) \rangle$ *where* $\mathsf{Pre}(o)$ *are the preconditions of* o, $\mathsf{Add}(o)$ *is the Add list of* o *and* $\mathsf{Del}(o)$ *is the Delete list of the action, each being a set of atoms. The result of applying a single STRIPS action to a state* S *when* $\mathsf{Pre}(o) \subseteq S$ *is defined as follows:*

$$Result(S, \langle o \rangle) = (S \cup \mathsf{Add}(o)) \setminus \mathsf{Del}(o)$$

The result of applying a sequence of actions to a state is recursively defined as:

$$Result(S, \langle o_1, \ldots, o_n \rangle) = Result(Result(S, \langle o_1, \ldots, o_{n-1} \rangle), \langle o_n \rangle).$$

Definition 2. *A* **planning task** $\mathcal{P} = (\mathcal{O}, \mathcal{I}, \mathcal{G})$ *is a triple where* $\mathcal{O}$ *is the set of actions, and* $\mathcal{I}$ *(the initial state) and* $\mathcal{G}$ *(the goals) are sets of atoms.*

The process to build a LG consists of two steps [6]: (1) extracting the set of landmarks and (2) ordering the obtained set of landmarks.

Definition 3. *Given a planning task* $\mathcal{P} = (\mathcal{O}, \mathcal{I}, \mathcal{G})$*, a fact* l_i *is a* **landmark** *in* $\mathcal{P}$ *iff* l_i *is true at some point in all solution plans, i.e., iff for all* $P = \langle o_1, \ldots, o_n \rangle, \mathcal{G} \subseteq Result(\mathcal{I}, P) : l_i \in Result(\mathcal{I}, \langle o_1, \ldots, o_i \rangle)$ *for some* $0 \leq i \leq n$.

Definition 4. *The* **side-effects** *of a landmark* l_i *are defined as:*
$side\_effects(l_i) = \{\mathsf{Add}(o) - \{l_i\} \mid o \in \mathcal{O}, l_i \in \mathsf{Add}(o)\}$

The extraction process is straightforward. First, a Relaxed Planning Graph[2] (RPG) is built. Then, all top level goals are added to the LG and posted as goals in the first level at which they were added in the RPG. Each goal is solved in the RPG starting from the last level. For each goal g in a level, all actions achieving g are grouped into a set and the intersection I of their preconditions is

[2] A RPG is a Graphplan Planning Graph [1] where the Delete list of the actions is ignored.

computed. We also compute a set U formed by the union of these preconditions which do not belong to I. This set is called **disjunctive set**. For all facts p in I we post p as a goal in the first RPG level where p is achieved. When all goals in a level are achieved, we move on to the next lower level. The process stops when the first (initial) level is reached.

Definition 5. *Let l_i, l_j be two landmarks. l_i and l_j are* **consistent** *if there exists a possible state $S / l_i \in S \wedge l_j \in S$.*

We use the TIMinconsistent function [3], which returns whether two literals are consistent or not. Once the set of landmarks has been extracted, they are ordered according to the following orders, which in turn define an LG:

Definition 6. *A* **natural order** *is established between two landmarks l_i, l_j ($l_i <_n l_j$) when in every solution plan it is necessary to solve l_i to achieve l_j, that is, l_i is a precondition of all the actions that satisfy l_j.*

Definition 7. *A* **weakly reasonable order** *is established between two landmarks l_i, l_j ($l_i <_{wr} l_j$) in the following cases:*

- *if l_i and l_j are naturally ordered before the same node l_k and $\exists$ landmark x : $x <_n l_i \wedge$ TIMinconsistent$(x, l_j) =$ TRUE*
- *if there exists some other landmark x, and x and l_j are ordered before the same node; and there is an ordered sequence of $<_n$ orders that post l_i before x; and $\exists$ landmark $y : y <_n l_i \wedge$ TIMinconsistent$(y, l_j) =$ TRUE*
- *if $\exists$ landmark $x : l_i <_n x \wedge l_j <_{wr} x \wedge$ TIMinconsistent$(side_effects(l_j), l_i) =$ TRUE*

A weakly reasonable order between two landmarks $l_i <_{wr} l_j$ states that it is better to satisfy l_i before l_j because if l_j is achieved first it might be eventually deleted by l_i.

Definition 8. *A* **LG** *is a graph (N, E) with three types of nodes:*

- simple node: $l_i \in N$ *as a simple node if l_i is a landmark.*
- disjunctive node: $l_i \in N$ *as a disjunctive node if l_i is a disjunctive set.*
- conjunctive node: $[l_i, l_j] \in N$ *as a conjunctive node if l_i is a side-effect of l_j and viceversa*

The set of edges E for a **LG** *is built as follows. Let l_i, l_j be simple or disjunctive nodes and $[l_n, l_m]$ be a conjunctive node:*

- *If $l_i <_n l_j \vee l_i <_{wr} l_j \rightarrow E = E \cup (l_i, l_j)$*
- *$\forall l_k / (l_k, l_n) \in E \vee (l_k, l_m) \in E \rightarrow E = E \cup (l_k, [l_n, l_m])$*

We consider that a *landmark* is any of the elements in N and we will use the notation $l_i < l_j$ throughout the rest of the paper to refer an edge (l_i, l_j) in the LG.

It is important to remark the fact that, both the process for extracting landmarks and calculating the orders are approximate computations, that is, not every landmark in a problem or all the orders between the obtained landmarks are extracted. For this reason the information in the LG may not be complete.

3 Building Intermediate Goals

This section explains how to build the intermediate goals.

Definition 9. *An* **intermediate goal** $\mathcal{IG}$ *is the set of literals that should be achieved next from the current state.*

Definition 10. *A* **fringe** $\mathcal{F}$ *is an* $\mathcal{IG}$ *whose literals can be reached by applying only one action for each literal.*

Currently, STeLLa v2.0 focuses on building fringes instead of intermediate goals. The reason is that it is easier to build subplans to reach the corresponding fringe than the IG, although some more reasoning is required for building a fringe than for building an IG, as we will explain later on.

The first step for obtaining a fringe is creating the LG between the current state and the top level goals. Then, all the landmarks whose predecessors nodes in the LG belong to the current state are included in $\mathcal{F}$. More formally, let $LG(N, E)$ be the current LG and $l_i \in N$, $l_i \in \mathcal{F}$ if $\forall l_j \in N / l_j < l_i, l_j \in$ current state. Due to the incompleteness of the LG it might be the case that it is not possible to achieve all the landmarks in $\mathcal{F}$. This happens in any of the two following cases:

1. **One or more literals have to be postponed due to inconsistency or optimality criteria.** A literal $l_i \in \mathcal{F}$ can be delayed in the following cases:
   (a) $\exists l_j \in \mathcal{F}$ / TIMinconsistent(l_i, l_j) = TRUE
   (b) $\exists l_j \in \mathcal{F}$ / TIMinconsistent(l_i, l_j) = FALSE but the producer actions for each literal are mutually exclusive, that is, they cannot be executed at the same time from the current state
   (c) Even though l_i is consistent with all the other literals and its producer actions would not cause any conflict, it may be the case that the achievement of this literal at this moment would lead to a non-optimal plan.
2. **One or more literals need more than one action to be reached.** In this case, a regression process is performed to compute the new fringe.

3.1 Postponing Literals

In this section, we focus on case 1. In order to avoid inconsistency or to reduce the addition of redundant actions in the plan, the solution is to add new orders between the literals in $\mathcal{F}$:

Inconsistency Orders (Cases 1a and 1b)
By **inconsistency orders** we refer to those orders that must be established between two landmarks to solve cases 1a and 1b.

Definition 11. *A literal* l_i **can be delayed** *if for every landmark following* l_i *there is at least a precedessor landmark* l_k *which is neither in the fringe nor in the current state:* $\forall l_j / l_i < l_j, \exists l_k / l_k < l_j \wedge l_k \notin \mathcal{F} \wedge l_k \notin$ currentstate.

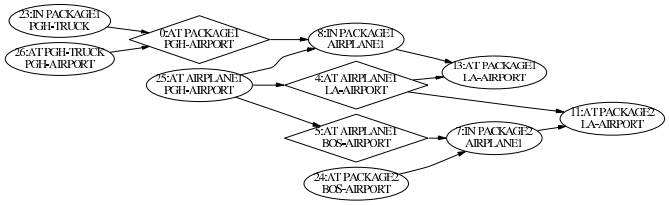

Fig. 1. Example of the order type 1 in the `logistics` problem

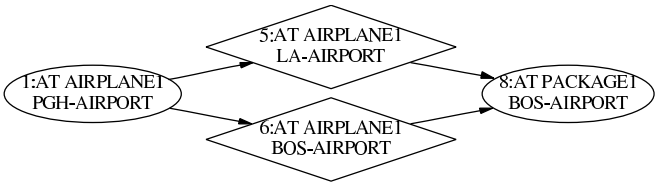

Fig. 2. Example of the order type 2 in the `logistics` problem

We now define three types of inconsistency orders. The first two types are aimed at solving case 1a and type 3 covers case 1b.

Type 1: If $\mathsf{TIMinconsistent}(l_i, l_j) = \mathsf{TRUE} \wedge$ can-be-delayed$(l_j) \wedge$ $\neg$can-be-delayed(l_i) then $l_i < l_j$.

Figure 1 shows an example where this order can be applied[3]. In this example, $\mathcal{F}$=$\{$(at package1 pgh-airport), (at airplane1 la-airport), (at airplane1 bos-airport)$\}$. Let's take l_i=(at airplane1 la-airport) and l_j=(at airplane1 bos-airport), as a pair of inconsistent literals. l_i can be delayed because (at package1 la-airport) has a predecessor landmark, (in package1 airplane1), which is neither in $\mathcal{F}$ nor in the current state. However, l_j cannot be delayed since the only previous landmark for (in package2 airplane1) belongs to the current state. Therefore, (at airplane1 bos-airport) should be achieved before (at airplane1 la-airport).

Type 2: If $\mathsf{TIMinconsistent}(l_i, l_j) = \mathsf{TRUE} \wedge \exists l_k / l_i <_{wr} l_k \wedge l_j <_n l_k$ then $l_i < l_j$.

The intuition behind this type of order is as follows. A natural order $l_j <_n l_k$ states that l_j is a precondition for l_k, that is, l_j must be true immediately before l_k. Therefore, if l_i is inconsistent with l_j, it seems reasonable to achieve l_i before so that l_j does not have to be reachieved after l_i.

Figure 2 shows an example of the order type 2. In this case, l_i=$\{$at airplane1 la-airport$\}$, l_j=$\{$at airplane1 bos-airport$\}$ and l_k=$\{$at package1 bos-airport$\}$, then l_i should be achieved in first place so that l_j is true just before satisfying l_k.

Type 3: If $\mathsf{TIMinconsistent}(l_i, l_j) = \mathsf{FALSE} \wedge \exists l_k / l_k < l_i \wedge l_k < l_j \wedge$ $\mathsf{TIMinconsistent}(l_j, l_k) = \mathsf{TRUE} \wedge \mathsf{TIMinconsistent}(l_i, l_k) = \mathsf{FALSE}$ then $l_i < l_j$.

[3] The nodes in diamond indicate the landmarks that belong to $\mathcal{F}$.

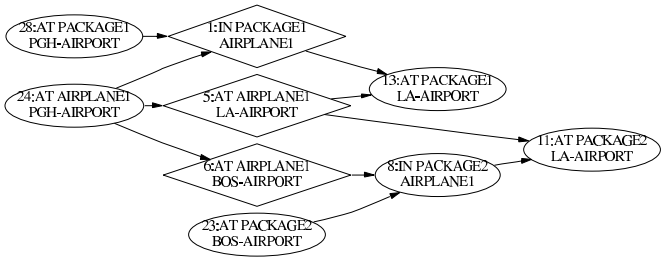

Fig. 3. Example of the order type 3 in the `logistics` problem

Figure 3 shows an example where this order is applied. In this example, $\mathcal{F}$={(in package1 airplane1), (at airplane1 la-airport), (at airplane1 bos-airport)}. If we take l_i=(in package1 airplane1), l_j=(at airplane1 la-airport) and l_k=(at airplane1 pgh-airport), since l_j and l_k are inconsistent and l_i and l_k are not, then l_j should be delayed. Otherwise, we could not achieve (in package1 airplane1) without reachieving (at airplane1 pgh-airport).

Optimality Orders (Case 1c)

The reasonable orders discovered when building the LG are aimed to reduce the number of actions in the final plan. Since not all these reasonable orders are found, we define some additional orders so as to improve the quality of the final plan. It is remarkable the fact that, while the inconsistency orders are defined between pairs of literals in $\mathcal{F}$, the optimality orders refer to a single literal.

Type 4: If the producer action of a landmark removes a literal which is required later on in the planning process, this landmark should be delayed. More formally, let l_i be a landmark in $\mathcal{F}$ and $P = \{l_j / l_j < l_i \wedge \mathsf{TIMinconsistent}(l_i, l_j) = \mathsf{TRUE}\}$. If $\exists p \in P / out-degree(p) > 1 \wedge \exists l_k \notin \mathcal{F} / p < l_k \wedge \mathsf{TIMinconsistent}(p, l_k) = \mathsf{FALSE}$ then l_i is delayed.

Figure 4 shows an example where this optimality order should be applied. Let's consider the node (at airplane1 la-airport) as l_i. Then, P ={(at airplane1 pgh-airport)}, and so is p. The out-degree of p is 2, setting l_k to (in package1 airplane1). l_k and p are consistent, so l_i has to be delayed. If we analyse this problem, we realise that if the literal (at airplane1 la-airport) is satisfied before (in package1 airplane1), we would have to achieve (at airplane1 pgh-airport) again for (in package1 airplane1), thus adding one more action.

Type 5: Let l_i be a landmark in $\mathcal{F}$. If $\exists l_j / l_i <_n l_j \wedge \exists l_k < l_j \wedge$ $\mathsf{TIMinconsistent}(l_i, l_k) = \mathsf{TRUE} \wedge l_k \notin$ current state then l_i is delayed.

In the example shown in Figure 5, if we consider (at airplane1 la-airport) as l_i, (at package2 la-airport) as l_j and (at airplane1 bos-airport) as l_k, the conditions above hold and, therefore, (at airplane1 la-airport) should be delayed. In this case, it is better going first to Boston to collect package2 and then transport both packages to la-airport, their final destination. Otherwise, if airplane1 headed la-

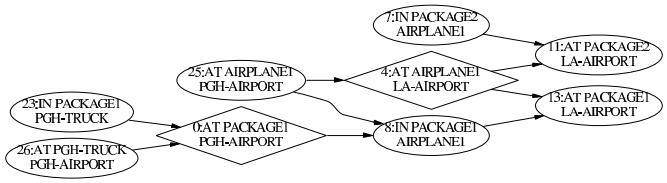

Fig. 4. Example of the order type 4 in the `logistics` problem

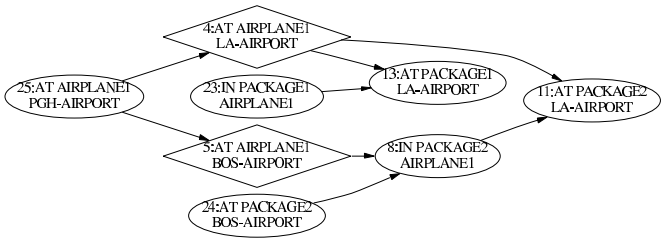

Fig. 5. Example of the order type 5 in the `logistics` problem

airport before bos-airport, then the plan would have one more action as it would be necessary to go to Boston from LA and then back again.

3.2 More-than-one-action Literals

As we said above, it might be the case that a literal in $\mathcal{F}$ cannot be reached by applying only one action from the current state. In this case, we perform a regression process that approaches the current fringe $\mathcal{F}$ to the current state. If we were working with $\mathcal{IG}$, we would not need to perform this regression process. This is done by finding the set of literals $\mathcal{F}'$ that should be reached before $\mathcal{F}$. This intermediate fringe is computed as follows:

$\mathcal{G}' = \emptyset$
`Forall the literals` $l_i \in \mathcal{F}$
    1. `Select an action` $\mathcal{A}$ `that achieves` l_i
    2. $\mathcal{G}' = \mathcal{G}' \cup \mathsf{Pre}(\mathcal{A})$
`Build a new fringe` $\mathcal{F}'$ `from` $\mathcal{I}$ `to` $\mathcal{G}'$
`If` $\mathcal{F}'$ `is a correct fringe, then return` $\mathcal{F}'$
`Else Regression` $(\mathcal{I}, \mathcal{F}')$

The selection of the action $\mathcal{A}$ is very important because a wrong selection may not lead to the best fringe. This can have a high impact in the final solution since it can cause the planner to add redundant actions, as the results in Section 5 will show.

4 Using the Intermediate Goals

In this section, we present the complete algorithm implemented in STeLLa .

Algorithm STeLLa $(\mathcal{I}, \mathcal{G}) \rightarrow$ plan $\mathcal{P}$ organized in time steps

---

$\mathcal{C} = \mathcal{I}$; *time* = 0
While $\mathcal{G} \not\subset \mathcal{C}$
    1. Compute fringe $\mathcal{F}$
    2. Compute the set of actions $\mathcal{A}$ that solve $\mathcal{F}$
    3. If $\mathcal{A} = \emptyset$ return $\mathcal{P} = \emptyset$
    4. Execute $\mathcal{A}$, obtaining the new $\mathcal{C}$
    5. $\mathcal{P}_{time} = \mathcal{A}$
    6. *time* = *time* + 1
return $\mathcal{P}$

While the planner has not reached the goals, it computes the next fringe $\mathcal{F}$. Then, the set of actions $\mathcal{A}$, necessary to solve the fringe, is computed. If $\mathcal{A}$ is empty the planner returns an empty plan. Otherwise, this set of actions is executed over $\mathcal{C}$, obtaining the new current state.

The most important difference between the current version and STeLLa v1.0 [7] is that in the latter the fringe after step 1 in the algorithm above, could contain inconsistent literals whereas in STeLLa v2.0 $\mathcal{F}$ is totally consistent. So, in the previous version it was necessary to create consistent subsets of goals from $\mathcal{F}$. Moreover, as no regression process was used the planner could reach a dead-end when the literals in the fringe could not be achieved in only one action.

5 Experiments

In this section, we summarize the experiments performed with STeLLa v2.0 for the IPC2002. We compare these results with the planners that were awarded a prize in this competition:

- The quality version of LPG planner [4] which exhibited a *distinguished performance of the first order* in the fully-automated track.
- MIPS [2] which exhibited a *distinguished performance* in the fully-automated track.
- VHPOP [8] which was awarded the *best newcomer* prize.

Table 1 shows only the results for the problems STeLLa was able to solve in the Depots and Driverlog domains out of 20 problems. For the Zeno and Satellite domains (Table 2), STeLLa was able to solve almost all of the problems.

We can see that in all domains the results obtained by STeLLa are comparable in terms of time steps and number of actions with the other planners. In terms of time[4], the performance of STeLLa varies along the domains. Although our main

[4] In the Driverlog we have considered only those problems solved by STeLLa, whereas in the Zeno and Satellite domains the time in average is computed over the problems solved by each planner.

Table 1. Depots and Driverlog problems (time steps / number of actions).

Depots					Driverlog				
	LPG	VHPOP	MIPS	STeLLa		LPG	VHPOP	MIPS	STeLLa
pfile1	8/10	8/10	12/13	6/11	pfile1	7/7	7/7	7/7	7/7
pfile2	12/15	12/15	9/16	9/16	pfile2	14/20	11/21	18/22	20/28
pfile3	21/27	/	21/34	17/33	pfile3	7/12	8/13	12/18	8/13
pfile8	21/35	/	27/48	30/56	pfile4	11/16	11/16	9/19	17/30
pfile10	13/24	/	22/34	25/30	pfile5	11/18	8/18	17/25	10/22
pfile13	19/25	/	18/25	11/30	pfile6	10/17	5/11	7/13	9/16
					pfile7	7/13	8/15	9/15	10/21
					pfile8	11/22	13/25	15/27	18/38
					pfile9	13/23	14/22	14/24	13/26
					pfile10	11/17	9/21	11/21	27/35
					Time in avg.	6986	1966	135	3426

concern is the plan quality, the performance in average is quite competitive for most of the domains.

One remarkable aspect is that STeLLa obtains sometimes better results for more complex instances than LPG. This is because the LG provides a more global view of the planning problem than local search planning.

It is also important to remark that STeLLa v2.0 obtains similar results to the previous version for those domains where regression is not needed. That is, both planners exhibit the same performance but the new version can solve a broader range of problems.

6 Conclusions and Further Work

In this paper we have presented the new version of STeLLa that participated in the IPC2002. STeLLa v2.0 offers a new planning approach consisting in building intermediate goals. However, the process to compute the intermediate goals is not reliable enough because of the regression step which may not lead to the best fringe.

Our ongoing work consists of building successive intermediate goals without being subject to be fringes. This way, if we can assure that these intermediate goals drive to an optimal plan, then solving a planning problem can be viewed as solving several subproblems each consisting in achieving the next intermediate goal.

References

1. A. Blum and M. Furst. Fast planning through planning graph analysis. *Artificial Intelligence*, 90(1-2):281–300, 1997.

Table 2. Zeno and Satellite problems (time steps / number of actions).

	Zeno				Satellite			
Problem	LPG	VHPOP	MIPS	STeLLa	LPG	VHPOP	MIPS	STeLLa
pfile1	1/1	1/1	1/1	1/1	8/9	8/9	8/9	8/9
pfile2	5/6	6/6	5/6	7/9	12/13	12/13	12/13	12/13
pfile3	5/6	5/6	7/9	5/6	10/11	10/11	10/11	10/11
pfile4	7/8	7/8	9/10	11/13	17/18	20/22	17/18	17/18
pfile5	7/11	9/12	11/16	11/15	14/16	15/16	14/16	15/19
pfile6	6/12	7/12	11/15	10/15	10/20	18/21	12/21	10/26
pfile7	8/15	7/16	6/16	6/16	12/21	13/25	14/23	19/26
pfile8	7/12	9/13	8/12	8/15	14/26	12/29	14/27	21/26
pfile9	7/23	10/21	10/25	16/32	10/30	/	10/35	13/31
pfile10	9/30	9/25	16/32	10/31	27/30	21/37	18/35	23/48
pfile11	6/15	6/16	8/18	9/18	27/32	13/33	13/35	23/40
pfile12	16/33	10/25	12/26	13/28	22/43	29/47	/	49/71
pfile13	16/38	10/27	9/33	11/35	22/61	24/64	22/60	43/84
pfile14	18/45	/	15/41	10/41	18/42	22/41	19/44	25/51
pfile15	11/54	/	14/54	23/61	16/48	19/50	17/50	19/56
pfile16	16/63	/	/	21/65	26/51	35/52	/	28/74
pfile17	45/96	/	/	46/131	28/43	22/43	/	27/70
pfile18	25/91	/	38/79	/	18/32	18/32	/	16/35
pfile19	59/113	/	/	27/117	25/75	/	/	/
pfile20	37/126	/	/	56/148	25/110	/	/	/
Time in avg.	164612	11380	43515	268315	13370	11528	34125	33981

2. S. Edelkamp. Symbolic pattern databases in heuristic search planning. In *Proceedings of the Sixth Int. Conference on AI Planning and Scheduling (AIPS'02)*. AAAI Press, 2002.
3. M. Fox and D. Long. The automatic inference of state invariants in TIM. *Journal of Artificial Intelligence Research*, 9:367–421, 1998.
4. A. Gerevini and I. Serina. Lpg: a planner based on local search for planning graphs. In *Proceedings of the Sixth Int. Conference on AI Planning and Scheduling (AIPS'02)*. AAAI Press, 2002.
5. J. Porteous and L. Sebastia. Extracting and ordering landmarks for planning. In *19th Workshop of the UK Planning and Scheduling Special Interest Group*, 2000.
6. J. Porteous, L. Sebastia, and J. Hoffmann. On the extraction, ordering, and usage of landmarks in planning. In *Recent Advances in AI Planning. 6th European Conference on Planning (ECP'01)*. Springer Verlag, 2001.
7. L. Sebastia, E. Onaindia, and E. Marzal. STeLLa: An optimal sequential and parallel planner. In *Proceedings of the 10th Portuguese Conference on Artificial Intelligence (EPIA'01)*. Springer Verlag, 2001.
8. H. Younes and R. Simmons. On the role of ground actions in refinement planning. In *Proceedings of the Sixth Int. Conference on AI Planning and Scheduling (AIPS'02)*. AAAI Press, 2002.