Landmark-Aware Strategies for Hierarchical Planning

Mohamed Elkawkagy and **Pascal Bercher** and **Bernd Schattenberg** and **Susanne Biundo**

Institute of Artificial Intelligence,
Ulm University, D-89069 Ulm, Germany,
email: *forename.surname*@uni-ulm.de

Abstract

In hierarchical planning, landmarks are abstract tasks the decomposition of which are mandatory when trying to find a solution to a given problem. In this paper, we present novel domain-independent strategies that exploit landmark information to speed up the planning process. The empirical evaluation shows that the landmark-aware strategies outperform established search strategies for hierarchical planning.

1 Introduction

While landmarks are widely used to improve the performance of classical planners, a different notion of landmarks has recently been developed for HTN-based approaches (Elkawkagy, Schattenberg, and Biundo 2010). Unlike the classical case where landmarks are facts that must hold in some intermediate state of any solution plan, hierarchical landmarks are mandatory tasks – tasks that have to be decomposed on any search path leading from the initial plan to a solution of the planning problem.

Hierarchical task network (HTN) planning relies on the concepts of tasks and methods (Erol, Hendler, and Nau 1994). While primitive tasks correspond to classical planning operators, abstract tasks are a means to represent complex activities. For each abstract task, a number of methods are available each of which provides a task network, i.e., a plan that specifies a predefined (abstract) solution of the task. Planning problems are (initial) task networks. They are solved by incrementally decomposing the abstract tasks until the network contains only primitive ones in executable order.

Strategies of HTN-based planners differ in the ways they select appropriate methods and interleave the decomposition of tasks with measures to resolve causal interactions between tasks. Systems of the SHOP family, like SHOP2, expand tasks in the order in which they are to be executed and consider causality only on primitive levels (Nau et al. 2003). Other strategies alternate task decomposition and causal conflict resolution (McCluskey 2000) or comply with the current state of the task network (Schattenberg, Bidot, and Biundo 2007).

In this paper, we describe how the exploitation of landmark information leads to novel domain-independent search strategies for HTN-based planning. A so-called landmark table is extracted from the current planning problem in a pre-processing step. It lists the landmark tasks and reveals the various options at hand. Options are tasks that are not mandatory, but may have to be decomposed depending on the method that is selected to implement the respective landmark. This information is used to compute the expansion effort of the problem – a heuristic to guide the selection of methods and with that reduce the effective branching factor of the search space. We implemented the landmark-aware planning strategies in our experimental setting and evaluated their performance on the well-established *Satellite* and *UM-Translog* benchmarks. It turned out that the novel strategies outperform their conventional counterparts on practically all problems, if the decomposition hierarchy of the underlying domain is of non-trivial depth.

The use of landmarks in hierarchical planning is quite novel. In classical state-based planning the concept of landmarks (Porteous, Sebastia, and Hoffmann 2001) enabled the development of strong heuristics (Helmert and Domshlak 2009; Bonet and Helmert 2010). LAMA, the currently best performing classical planner uses such a landmark heuristic (Richter and Westphal 2010). The work of Zhu and Givan (2004) generalized landmarks to so-called action landmarks. As for HTN-based planning, Marthi, Russell, and Wolfe (2008) introduce abstract landmark facts that are gained from effects of basic actions via incremental abstraction.

In the remainder of the paper, we give a brief introduction into the underlying planning framework and the concept of hierarchical landmarks. We then define the landmark-aware strategies and describe the experimental setting as well as the evaluation results.

2 Planning Framework

The planning framework is based on a hybrid formalization (Biundo and Schattenberg 2001) which fuses HTN planning with partial-order causal-link (POCL) planning. For the purpose of this paper, only the HTN shares of the framework are considered, however. A *task schema* $t(\overline{\tau}) = \langle \text{prec}, \text{eff} \rangle$ specifies the preconditions and effects of a task via conjunctions of literals over the task parameters $\bar{\tau} = \tau_1 \dots \tau_n$. *States* are sets of literals. Applicability of tasks and the state transformations caused by their execution are defined as usual. A *plan* $P = \langle S, \prec, V, C \rangle$ consists of a set S of *plan steps*, i.e., uniquely labeled, (partially) instantiated

tasks, a set $\prec$ of *ordering constraints* that impose a partial order on S, a set V of *variable constraints*, and a set C of *causal links*. V consists of (in)equations that associate variables with other variables or constants; it also reflects the (partial) instantiation of the plan steps in P. We denote by $Ground(S, V)$ the set of ground tasks obtained by equating all parameters of all tasks in P with constants, in a way compatible with V. The causal links are adopted from POCL planning: a causal link $l{:}t(\bar{\tau}) \rightarrow_{\varphi} l'{:}t'(\bar{\tau}')$ indicates that φ is implied by the precondition of plan step $l'{:}t'(\bar{\tau}')$ and at the same time is a consequence of the effects of plan step $l{:}t(\bar{\tau})$. Hence, φ is said to be *supported* this way. Methods $m = \langle t(\bar{\tau}), P\rangle$ relate an abstract task $t(\bar{\tau})$ to a plan P, which is called an *implementation* of $t(\bar{\tau})$. Multiple methods may be provided for each abstract task.

An HTN planning problem $\Pi = \langle D, s_{\text{init}}, P_{\text{init}}\rangle$ is composed of a domain model $D = \langle T, M\rangle$, where T and M denote sets of task schemata and decomposition methods, an initial state s_{init}, and an initial plan P_{init}. A plan $P = \langle S, \prec, V, C\rangle$ is a solution to Π if and only if:

1. P is a refinement of P_{init}, i.e., a successor of the initial plan in the induced search space (see Def. 1 below);
2. each precondition of a plan step in S is supported by a causal link in C and no such link is threatened, i.e., for each causal link $l{:}t(\bar{\tau}) \rightarrow_{\varphi} l'{:}t'(\bar{\tau}')$ the ordering constraints in $\prec$ ensure that no plan step $l''{:}t''(\bar{\tau}'')$ with an effect that implies $\neg\varphi$ can be placed between plan steps $l{:}t(\bar{\tau})$ and $l'{:}t'(\bar{\tau}')$;
3. the ordering and variable constraints are consistent, i.e., $\prec$ does not induce cycles on S and the (in)equations in V are not contradictory; and
4. all plan steps in S are primitive ground tasks.

$\mathcal{S}ol_{\Pi}$ denotes the set of all solutions of Π.

Please note that we encode the initial state via the effects of an artificial primitive "start" task, as it is usually done in POCL planning. In doing so, the second criterion guarantees that the solution is executable in the initial state.

In order to refine the initial plan into a solution, there are various *refinement steps* (or *plan modifications*) available; in HTN planning, these are: (1) the decomposition of abstract tasks using methods, (2) the insertion of causal links to support open preconditions of plan steps, (3) the insertion of ordering constraints, and (4) the insertion of variable constraints. Given an HTN planning problem we can define the induced search space as follows.

Definition 1 (Induced Search Space) *The directed graph $\mathcal{P}_{\Pi} = \langle \mathcal{V}_{\Pi}, \mathcal{E}_{\Pi}\rangle$ with vertices $\mathcal{V}_{\Pi}$ and edges $\mathcal{E}_{\Pi}$ is called the* induced search space of planning problem Π *if and only if (1) $P_{\text{init}} \in \mathcal{V}_{\Pi}$, (2) if there is a plan modification refining $P \in \mathcal{V}_{\Pi}$ into a plan P', then $P' \in \mathcal{V}_{\Pi}$ and $(P, P') \in \mathcal{E}_{\Pi}$, and (3) $\mathcal{P}_{\Pi}$ is minimal such that (1) and (2) hold.*

For $\mathcal{P}_{\Pi} = \langle \mathcal{V}_{\Pi}, \mathcal{E}_{\Pi}\rangle$, we write $P \in \mathcal{P}_{\Pi}$ instead of $P \in \mathcal{V}_{\Pi}$.

Note that $\mathcal{P}_{\Pi}$ is in general neither acyclic nor finite. For the former, consider a planning problem in which there are the abstract tasks $t(\bar{\tau})$, $t'(\bar{\tau}')$ as well as two methods, each of which transforms one task into the other. For the latter, consider a planning problem containing an abstract task $t(\bar{\tau})$ and a primitive task $t'(\bar{\tau}')$ as well as two methods for $t(\bar{\tau})$: one maps $t(\bar{\tau})$ to a plan containing only $t'(\bar{\tau}')$, the other maps $t(\bar{\tau})$ to a plan containing $t'(\bar{\tau}')$ and $t(\bar{\tau})$ thus enabling the construction of arbitrary long plans.

In order to search for solutions the induced search space is explored in a heuristically guided manner by the following standard refinement planning algorithm:

Algorithm 1: Refinement Planning Algorithm

Input : The sequence $\texttt{Fringe} = \langle P_{\text{init}}\rangle$.
Output : A solution or `fail`.

```
1 while Fringe = ⟨P_1 ... P_n⟩ ≠ ε do
2     F ← f^FlawDet(P_1)
3     if F = ∅ then return P_1
4     ⟨m_1 ... m_k⟩ ← f^ModOrd( ⋃_{f∈F} f^ModGen(f))
5     succ ← ⟨app(m_1, P_1) ... app(m_k, P_1)⟩
6     Fringe ← f^PlanOrd(succ ∘ ⟨P_2 ... P_n⟩)
7 return fail
```

The fringe $\langle P_1 \dots P_n\rangle$ is a sequence containing all unexplored plans that are direct successors of visited non-solution plans in $\mathcal{P}_{\Pi}$. It is ordered in a way such that a plan P_i is estimated to lead more quickly to a solution than plans P_j for $j > i$. The current plan is always the first plan of the fringe. The planning algorithm iterates on the fringe as long as no solution is found and there are still plans to refine (line 1). Hence, the flaw detection function f^{FlawDet} in line 2 calculates all flaws of the current plan. A flaw is a set of plan components that are involved in the violation of a solution criterion. The presence of an abstract task raises a flaw that consists of that task, a causal threat consists of a causal link and the threatening plan step, for example. If no flaws can be found, the plan is a solution and returned (line 3). In line 4, the modification generating function f^{ModGen} calculates all plan modifications that address the flaws of the current plan. Afterwards, the modification ordering function f^{ModOrd} orders these modifications according to a given strategy. The fringe is finally updated in two steps: first, the plans resulting from applying the modifications are computed (line 5) and put at the beginning of the fringe (line 6). Second, the plan ordering function f^{PlanOrd} orders the updated fringe. This step can also be used to discard plans, i.e., to delete plans permanently from the fringe. This is useful for plans that contain unresolvable flaws like an inconsistent ordering of tasks. If the fringe becomes empty, no solution exists and `fail` is returned.

In this setting, the search strategy appears as a combination of the plan modification and plan ordering functions. In order to perform a depth first search, for example, the plan ordering is the identity function ($f^{\text{PlanOrd}}(\overline{P}) = \overline{P}$ for any sequence $\overline{P}$), whereas the modification ordering f^{ModOrd} determines, which branch of the search space to visit first.

3 Landmarks

The landmark-aware planning strategies rely on hierarchical and local landmarks – ground tasks that occur in the plan sequences leading from a problem's initial plan to its solution.

Definition 2 (Solution Sequences) *Let $\langle \mathcal{V}_\Pi, \mathcal{E}_\Pi \rangle$ be the induced search space of planning problem Π. Then, for any plan $P \in \mathcal{V}_\Pi$, $\mathcal{S}ol\mathcal{S}eq_\Pi(P) := \{\langle P_1 \dots P_n \rangle \mid P_1 = P, (P_i, P_{i+1}) \in \mathcal{E}_\Pi$ for all $1 \leq i < n$, and $P_n \in \mathcal{S}ol_\Pi$ for $n \geq 1\}$.*

Definition 3 (Landmark) *A ground task $t(\overline{\tau})$ is called a* landmark *of planning problem Π, if and only if for each $\langle P_1 \dots P_n \rangle \in \mathcal{S}ol\mathcal{S}eq_\Pi(P_{\text{init}})$ there is an $1 \leq i \leq n$, such that $t(\overline{\tau}) \in Ground(S_i, V_n)$ for $P_i = \langle S_i, \prec_i, V_i, C_i \rangle$ and $P_n = \langle S_n, \prec_n, V_n, C_n \rangle$.*

While a landmark occurs in every plan sequence that is rooted in the initial plan and leads towards a solution, a *local* landmark occurs merely in each such sequence rooted in a plan containing a specific abstract ground task $t(\overline{\tau})$.

Definition 4 (Local Landmark of an Abstract Task) *For an abstract ground task $t(\overline{\tau})$ let $\mathcal{P}_\Pi(t(\overline{\tau})) := \{P \in \mathcal{P}_\Pi \mid P = \langle S, \prec, V, C \rangle$ and $t(\overline{\tau}) \in Ground(S, V)\}$.*
A ground task $t'(\overline{\tau}')$ is a local landmark *of $t(\overline{\tau})$, if and only if for all $P \in \mathcal{P}_\Pi(t(\overline{\tau}))$ and each $\langle P_1 \dots P_n \rangle \in \mathcal{S}ol\mathcal{S}eq_\Pi(P)$ there is an $1 \leq i \leq n$, such that $t'(\overline{\tau}') \in Ground(S_i, V_n)$ for $P_i = \langle S_i, \prec_i, V_i, C_i \rangle$ and $P_n = \langle S_n, \prec_n, V_n, C_n \rangle$.*

Since there are only finitely many task schemata and we assume only finitely many constants, there is only a finite number of (local) landmarks.

Given a planning problem Π, the relevant landmark information can be extracted in a pre-processing step. We use the extraction procedure introduced in previous work of the authors (Elkawkagy, Schattenberg, and Biundo 2010) and assume that the landmark information is already stored in a so-called *landmark table*. Its definition relies on a task decomposition graph, which is a relaxed representation of how the initial plan of a planning problem can be decomposed.

Definition 5 (Task Decomposition Graph) *The directed bipartite graph $\langle V_T, V_M, E \rangle$ with task vertices V_T, method vertices V_M, and edges E is called the task decomposition graph (*TDG*) of planning problem Π if and only if*

1. *$t(\overline{\tau}) \in V_T$ for all $t(\overline{\tau}) \in Ground(S, V)$, for $P_{init} = \langle S, \prec, V, C \rangle$,*
2. *if $t(\overline{\tau}) \in V_T$ and if there is a method $\langle t(\overline{\tau}'), \langle S, \prec, V, C \rangle \rangle \in M$, then*
   (a) *$\langle t(\overline{\tau}), \langle S, \prec, V', C \rangle \rangle \in V_M$ such that $V' \supseteq V$ binds all variables in S to a constant and*
   (b) *$(t(\overline{\tau}), \langle t(\overline{\tau}), \langle S, \prec, V', C \rangle \rangle) \in E$,*
3. *if $\langle t(\overline{\tau}), \langle S, \prec, V, C \rangle \rangle \in V_M$, then*
   (a) *$t'(\overline{\tau}') \in V_T$ for all $t'(\overline{\tau}') \in Ground(S, V)$ and*
   (b) *$(\langle t(\overline{\tau}), \langle S, \prec, V, C \rangle \rangle, t'(\overline{\tau}')) \in E$, and*
4. *$\langle V_T, V_M, E \rangle$ is minimal such that (1), (2), and (3) hold.*

Note that the TDG of a planning problem is always finite as there are only finitely many ground tasks.

Please also note that, due to the uninformed instantiation of unbound variables in a decomposition step in criterion 2.(a), the TDG of a planning problem becomes in general intractably large. We hence prune parts of the TDG which can provably be ignored due to a relaxed reachability analysis of primitive tasks. This pruning technique is described in our earlier work (Elkawkagy, Schattenberg, and Biundo 2010).

The *landmark table* is a data structure that represents a (possibly pruned) TDG plus additional information about local landmarks.

Definition 6 (Landmark Table) *Let $\langle V_T, V_M, E \rangle$ be a (possibly pruned)* TDG *of the planning problem Π. The* landmark table *of Π is the set $LT = \{\langle t(\overline{\tau}), M(t(\overline{\tau})), O(t(\overline{\tau})) \rangle \mid t(\overline{\tau}) \in V_T$ abstract ground task$\}$, where $M(t(\overline{\tau}))$ and $O(t(\overline{\tau}))$ are defined as follows:*

$$M(t(\overline{\tau})) := \{t'(\overline{\tau}') \in V_T \mid t'(\overline{\tau}') \in Ground(S, V) \text{ for all } \langle t(\overline{\tau}), \langle S, \prec, V, C \rangle \rangle \in V_M\}$$

$$O(t(\overline{\tau})) := \{\, Ground(S, V) \setminus M(t(\overline{\tau})) \mid \langle t(\overline{\tau}), \langle S, \prec, V, C \rangle \rangle \in V_M\}$$

Each landmark table entry partitions the tasks introduced by decompositions into two sets: mandatory tasks $M(t(\overline{\tau}))$ are those ground tasks that are contained in all plans introduced by some method which decomposes $t(\overline{\tau})$; hence, they are local landmarks of $t(\overline{\tau})$. The optional task set $O(t(\overline{\tau}))$ contains for each method decomposing $t(\overline{\tau})$ the set of ground tasks which are not in the mandatory set; it is hence a set of sets of tasks.

Please note that the landmark table encodes a possibly pruned TDG and is thus not unique. In fact, various local landmarks might only be detected after pruning. For instance, suppose an abstract task has three available methods, two of which have some tasks in their referenced plans in common. However, the plan referenced by the third method is disjunctive to the other two. Hence, the mandatory sets are empty. If the third method can be proven to be infeasible and is hence pruned from the TDG, the mandatory set will contain those tasks the plans referenced by the first two methods have in common.

Example

The following example will demonstrate how the TDG and a landmark table of a planning problem looks like.

Thus, let $\Pi = \langle D, s_{\text{init}}, P_{\text{init}} \rangle$ an HTN planning problem with $P_{\text{init}} = \langle \{l_1{:}t_1(\tau_1)\}, \{\tau_1 = c_1\} \rangle$[1], $D = \langle T, M \rangle$, $T = \{t_1(\tau_1), \dots, t_5(\tau_5)\}$, and $M = \{m_a, m'_a, m_b, m'_b\}$ with:

$$m_a := \langle t_1(\tau_1), \langle \{l_1{:}t_3(\tau_1), l_2{:}t_3(\tau_2), l_3{:}t_2(\tau_1)\}, \{\tau_1 \neq \tau_2\} \rangle \rangle$$
$$m'_a := \langle t_1(\tau_1), \langle \{l_1{:}t_2(\tau_1), l_2{:}t_1(\tau_1)\}, \emptyset \rangle \rangle$$
$$m_b := \langle t_3(\tau_1), \langle \{l_1{:}t_4(\tau_1), l_2{:}t_5(\tau_1)\}, \emptyset \rangle \rangle$$
$$m'_b := \langle t_3(\tau_1), \langle \{l_1{:}t_4(\tau_1)\}, \emptyset \rangle \rangle$$

The TDG for Π is given in Figure 1; the according landmark table is depicted in Table 1.

[1] As our example comes without ordering constraints and causal links, we give plans as 2-tuples $P = \langle S, V \rangle$

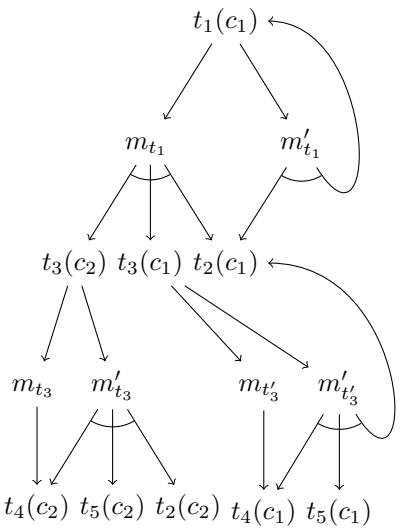

Figure 1: The TDG for the planning problem Π. The method vertices are given as follows:
$m_{t_1} = \langle t_1(c_1), m_{a|_{\tau_1=c_1,\tau_2=c_2}} \rangle$, $m'_{t_1} = \langle t_1(c_1), m'_{a|_{\tau_1=c_1}} \rangle$,
$m_{t_3} = \langle t_3(c_2), m_{b|_{\tau_1=c_2}} \rangle$, $m'_{t_3} = \langle t_3(c_2), m'_{b|_{\tau_1=c_2}} \rangle$,
$m_{t'_3} = \langle t_3(c_1), m_{b|_{\tau_1=c_1}} \rangle$, $m'_{t'_3} = \langle t_3(c_1), m'_{b|_{\tau_1=c_2}} \rangle$

Table 1: The landmark table for the TDG of Figure 1.

Abs. Task	Mandatory	Optional
$t_1(c_1)$	$\{t_2(c_1)\}$	$\{\{t_3(c_2), t_3(c_1)\}, \{t_1(c_1)\}\}$
$t_3(c_2)$	$\{t_4(c_2)\}$	$\{\emptyset, \{t_5(c_2), t_2(c_2)\}\}$
$t_3(c_1)$	$\{t_4(c_1)\}$	$\{\emptyset, \{t_5(c_1), t_2(c_1)\}\}$

4 Landmark-Aware Strategies

Exploiting landmarks during planning is based on the idea that identifying such landmarks along the refinement paths perfectly guides the search process because they are "inevitable" elements on the way to any solution. The mandatory sets in the landmark table do not contribute directly to the identification of a solution path. They do, however, allow to estimate upper and lower bounds for the number of expansions an abstract task requires before a solution is found. A landmark table entry $\langle t(\overline{\tau}), M(t(\overline{\tau})), O(t(\overline{\tau})) \rangle$ carries the following information: if the planning system decomposes the task $t(\overline{\tau})$, all tasks in the mandatory set $M(t(\overline{\tau}))$ are introduced into the refinement plan, no matter which method is used. With the optional tasks at hand we can now infer that in the most optimistic case a solution can be developed straight from the implementation of the method with the "smallest" remains according to $O(t(\overline{\tau}))$. Following a similar argument, the upper bound for the "expansion effort" can be obtained by adding the efforts for all implementations that are stored in the optional set.

From the above considerations, two essential properties of our landmark-aware strategies emerge: first, since the landmark exploitation will be defined in terms of measuring expansion alternatives, the resulting strategy component has to be a modification ordering function. Second, if we base the modification preference on the optional sets in the landmark table entries, we implement an abstract view on the method definition that realizes the least-commitment principle.

Concerning the first two strategies below, we interpret the term "expansion effort" literally and therefore define "smallest" method to be the one with the fewest abstract tasks in the implementing plan. To this end, we define the cardinality of a set of tasks in terms of the number of corresponding entries that a given landmark table does contain.

Definition 7 (Landmark Cardinality) *Given a landmark table LT, we define the* landmark cardinality *of a set of tasks $o = \{t_1(\overline{\tau}_1), \ldots, t_n(\overline{\tau}_n)\}$ to be*

$$|o|_{LT} := |\{t(\overline{\tau}) \in o \mid \langle t(\overline{\tau}), M(t(\overline{\tau})), O(t(\overline{\tau})) \rangle \in LT\}|$$

A heuristic based on this information is obviously a rough over-estimation of the search effort because the landmark table typically contains a number of tasks that turn out to be unachievable in the given problem. The strategy also does not take into account the refinement effort it takes to make an implementation operational on the primitive level by establishing causal links, resolving causal threats, and grounding tasks. For the time being, we assume that all methods deviate from a perfect heuristic estimate more or less to the same amount. We will see that this simplification actually yields a heuristic with good performance.

Definition 8 (Landmark-aware strategy lm_1) *Given a plan $P = \langle S, \prec, V, C \rangle$, let $t_i(\overline{\tau}_i)$ and $t_j(\overline{\tau}_j)$ be ground instances of two abstract tasks in S that are compatible with the (in)equations in V and that are referenced by two abstract task flaws $\mathtt{f}_i$ and $\mathtt{f}_j$, respectively, that are found in P. Let a given landmark table LT contain the corresponding entries $\langle t_i(\overline{\tau}_i), M(t_i(\overline{\tau}_i)), O(t_i(\overline{\tau}_i)) \rangle$ and $\langle t_j(\overline{\tau}_j), M(t_j(\overline{\tau}_j)), O(t_j(\overline{\tau}_j)) \rangle$.*

The modification ordering function lm_1 orders a plan modification $\mathtt{m}_i$ before $\mathtt{m}_j$ if and only if $\mathtt{m}_i$ addresses $\mathtt{f}_i$, $\mathtt{m}_j$ addresses $\mathtt{f}_j$, and

$$\sum_{o \in O(t_i(\overline{\tau}_i))} |o|_{LT} < \sum_{o \in O(t_j(\overline{\tau}_j))} |o|_{LT}$$

This strategy implements the least commitment principle, as it favors those decomposition plan refinements that impose less successor plans. It reduces the effective branching factor of the search space (cf. *fewest alternatives first* heuristic in HTN planning (Tsuneto, Nau, and Hendler 1997)). The proper choice of the ground task instances $t_i(\overline{\tau}_i)$ and $t_j(\overline{\tau}_j)$ in the above definition is crucial for the actual performance, however, because the plan modifications typically operate on the lifted abstract tasks and method definitions.

While the above heuristic focuses on the very next level of refinement, a strategy should also take estimates for subsequent refinement levels into account, thus minimizing the number of refinement choices until no more decompositions are necessary. To this end, for a given landmark table LT, let $O^*(t(\overline{\tau}))$ be the transitive closure of the optional sets on a recursive traversal of the table entries, beginning in $t(\overline{\tau})$.

Definition 9 (Closure of the Optional Set) *The* closure of the optional set *for a given ground task $t(\overline{\tau})$ and a landmark table LT is the smallest set $O^*(t(\overline{\tau}))$, such that $O^*(t(\overline{\tau})) = \emptyset$ for primitive $t(\overline{\tau})$, and otherwise:*

$$O^*(t(\overline{\tau})) = O(t(\overline{\tau})) \cup \bigcup_{o \in O(t(\overline{\tau}))} \Big(\bigcup_{t'(\overline{\tau}') \in o} O^*(t'(\overline{\tau}')) \Big)$$

$$\textit{with } \langle t(\overline{\tau}), M(t(\overline{\tau})), O(t(\overline{\tau})) \rangle \in LT$$

Note that $O^*(t(\overline{\tau}))$ is always finite due to the finiteness of the landmark table, even for cyclic method definitions.

Considering the the previous example (cf. Figure 1 and Table 1), the closures for the three abstract tasks of the planning problem Π are as follows: $O^*(t_1(c_1)) = O(t_1(c_1)) \cup O(t_3(c_2)) \cup O(t_3(c_1))$, $O^*(t_3(c_2)) = O(t_3(c_2))$, and $O^*(t_3(c_1)) = O(t_3(c_1))$.

Definition 10 (Landmark-aware strategy lm_1^*) *Given the prerequisites from Def. 8, the modification ordering function lm_1^* orders a plan modification $\mathtt{m}_i$ before $\mathtt{m}_j$ if and only if $\mathtt{m}_i$ addresses $\mathtt{f}_i$, $\mathtt{m}_j$ addresses $\mathtt{f}_j$, and*

$$\sum_{o \in O^*(t_i(\overline{\tau}_i))} |o|_{LT} < \sum_{o \in O^*(t_j(\overline{\tau}_j))} |o|_{LT}$$

So far, the "expansion effort" has been measured in terms of decompositions that have to be applied until a solution is obtained. The following strategies take into account that also primitive tasks in a decomposition contribute to the costs for developing the current plan into a solution. The cost measure is thereby a uniform one: solving the flaws affecting a primitive task is regarded as expensive as the expansion of an abstract task.

Definition 11 (Landmark-aware strategy lm_2) *Given the prerequisites from Def. 8, the modification ordering function lm_2 orders a plan modification $\mathtt{m}_i$ before $\mathtt{m}_j$ if and only if $\mathtt{m}_i$ addresses $\mathtt{f}_i$, $\mathtt{m}_j$ addresses $\mathtt{f}_j$, and*

$$\sum_{o \in O(t_i(\overline{\tau}_i))} |o| < \sum_{o \in O(t_j(\overline{\tau}_j))} |o|$$

Like we did for the landmark-aware strategy lm_1, we define a variant for strategy lm_2 that examines the transitive closure of the optional sets.

Definition 12 (Landmark-aware strategy lm_2^*) *Given the prerequisites from Def. 8, the modification ordering function lm_2^* orders a plan modification $\mathtt{m}_i$ before $\mathtt{m}_j$ if and only if $\mathtt{m}_i$ addresses $\mathtt{f}_i$, $\mathtt{m}_j$ addresses $\mathtt{f}_j$, and*

$$\sum_{o \in O^*(t_i(\overline{\tau}_i))} |o| < \sum_{o \in O^*(t_j(\overline{\tau}_j))} |o|$$

Since the landmark information can be extracted from any domain model and problem in an automated pre-processing step, the above strategies are conceptually domain- and problem-independent heuristics. In addition, they are independent from the actual plan generation procedure, hence their principles can be incorporated into any refinement-based hierarchical planning system.

5 Evaluation

We evaluated the performance of the landmark-aware strategies in a series of experiments in comparison to conventional hierarchical search strategies.

We base our evaluation on the same benchmark problems as in our previous work (Elkawkagy, Schattenberg, and Biundo 2010) including the domain reduction technique. In the new experiments, we compare our novel landmark-aware strategies with conventional ones and thereby show that even on the reduced domain models the landmark information can be used to improve the search efficiency.

Conventional Hierarchical Search Strategies

For the strategies *SHOP* and *UMCP*, we used plan and modification ordering functions that induce the search strategies of these planning systems: in the UMCP system (Erol, Hendler, and Nau 1994), plans are primarily developed into completely primitive plans in which causal interactions are dealt with afterwards. The SHOP strategy (Nau et al. 2003) prefers task expansion for the abstract tasks in the order in which they are to be executed.

In all other strategies the plan ordering function *Fewer Modifications First (fmf)* was used. It prefers plans for which a smaller number of refinement options is found, thereby implementing the least commitment principle on the plan ordering level. For the comparison to our landmark-aware modification ordering functions, we also conducted experiments with the following modification ordering functions:

The *Expand-Then-Make-Sound (ems)* procedure (McCluskey 2000) alternates task expansion with other modifications, which results in a "level-wise" concretion of all plan steps. We also included the well-established *Least Committing First (lcf)* paradigm, a generalization of POCL strategies, which prefers those modifications that address flaws for which the smallest number of alternative solutions is available. From more recent work (Schattenberg, Bidot, and Biundo 2007), two HotSpot-based strategies were deployed. HotSpots denote plan components that are affected by multiple flaws, thereby quantifying to which extent solving one deficiency may interfere with the solution options for coupled components. The *Direct Uniform HotSpot (du-HotSpot)* strategy strictly avoids to address flaws that refer to HotSpot plan components. While the du-HotSpot heuristic treats all flaws uniformly when calculating their interference potential, the *Direct Adaptive HotSpot (da)* strategy puts problem-specific weights on binary combinations of flaw types that occur in the plan. It adapts to a repeated occurrence of flaw type combinations by increasing their weights: if abstract task flaws happen to coincide with causal threats, their combined occurrence becomes more important for the current plan generation episode. As a generalization of singular HotSpots to commonly affected areas of plan components, the *HotZone* modification ordering function takes connections between HotSpots into account and tries to evade modifications that deal with these clusters.

Experimental Results

We conducted our experiments on two well-established planning domains (cf. Table 2). *Satellite* is a benchmark

for non-hierarchical planning. The hierarchical encoding of this domain regards the original primitive operators as implementations of abstract observation tasks. The domain model consists of 3 abstract and 5 primitive tasks, and includes 8 methods. *UM-Translog* is a hierarchical planning domain that supports transportation and logistics. It shows 21 abstract and 48 primitive tasks as well as 51 methods.

Please note that we performed our experiments on the reduced domain models. For the satellite domain, the domain model reduction did not have any effect on the number of tasks and/or methods; for the UM-Translog domain, the size of the reduced domain models depends on the given problem instance and is on average 63% as large as the unreduced domain model (Elkawkagy, Schattenberg, and Biundo 2010).

The strategies lm_1, lm_1^*, lm_2, and lm_2^* do outperform the other strategies on practically all problems in the UM-Translog domain (cf. Table 2a) in terms of both size of the explored search space and computation time. This is quite surprising because the landmark table does not reveal any information about causal dependencies on the primitive task level and the strategies hence cannot provide a focused guidance. An adequate selection of the decomposition refinements obviously pays off well enough to compensate for random choice on the causality issues. Another interesting facet is that the strategies lm_1^*/lm_2^* being the better informed heuristic while repeatedly performing worse than lm_1/lm_2. Furthermore, the same anomaly occurs when comparing lm_2/lm_2^* with the more abstract but also more successful lm_1/lm_1^*. We suppose these phenomena result from two sources: First, the random choice of ground candidates for the lifted task instances is relatively unreliable and this effect gets amplified by traversing along the landmark closures and into the primitive task level. Second, the most important choice points are on the early decomposition levels, i.e., once a method has been chosen for implementing the transport, this refinement puts more constraints on the remaining decisions than the strategy can infer from the feasibility analysis underlying the landmark table.

On the Satellite domain our landmark-aware strategies do not clearly dominate any other strategy (cf. Table 2b). This meets our expectations as there is hardly any landmark information available due to the shallow decomposition hierarchy of this domain and any landmark-centered strategy is bound to loose its strength given limited landmark information. However, none of the other strategies in this domain dominated any landmark-aware strategy; thus, all evaluated strategies can be regarded as equally good.

6 Conclusion

In this paper, we introduced four novel landmark-aware search strategies to improve hierarchical planning. In a number of experiments these strategies competed with a set of representative search procedures from the literature. The results showed that the new strategies outperformed the established ones on all relevant problems, i.e., problems with a deep task hierarchy. Further work will be devoted to the construction and evaluation of other types of landmark-aware strategies and to the investigation of those domain model and problem features that suggest their deployment.

ACKNOWLEDGEMENTS

This work is done within the Transregional Collaborative Research Centre SFB/TRR 62 “Companion-Technology for Cognitive Technical Systems” funded by the German Research Foundation (DFG).

References

Biundo, S., and Schattenberg, B. 2001. From abstract crisis to concrete relief – a preliminary report on combining state abstraction and HTN planning. In *Proc. of the 6th European Conference on Planning (ECP 2001)*, 157–168. Springer.

Bonet, B., and Helmert, M. 2010. Strengthening landmark heuristics via hitting sets. In *Proc. of ECAI 2010*, volume 215 of *Frontiers in Artificial Intelligence and Applications*, 329–334. IOS Press.

Elkawkagy, M.; Schattenberg, B.; and Biundo, S. 2010. Landmarks in hierarchical planning. In *Proc. of ECAI 2010*, volume 215 of *Frontiers in Artificial Intelligence and Applications*, 229–234. IOS Press.

Erol, K.; Hendler, J.; and Nau, D. S. 1994. UMCP: A sound and complete procedure for hierarchical task-network planning. In *Proc. of the 2nd Int. Conf. on Artificial Intelligence Planning Systems (AIPS 1994)*, 249–254.

Helmert, M., and Domshlak, C. 2009. Landmarks, critical paths and abstractions: What’s the difference anyway? In *Proc. of ICAPS 2009*, 162–169.

Marthi, B.; Russell, S. J.; and Wolfe, J. 2008. Angelic hierarchical planning: Optimal and online algorithms. In *Proc. of ICAPS 2008*, 222–231.

McCluskey, T. L. 2000. Object transition sequences: A new form of abstraction for HTN planners. In *Proc. of the 5th Int. Conf. on Artificial Intelligence Planning Systems (AIPS 2000)*, 216–225. AAAI Press.

Nau, D. S.; Au, T.; Ilghami, O.; Kuter, U.; Murdock, W.; Wu, D.; and Yaman, F. 2003. SHOP2: An HTN planning system. *JAIR* 20:379–404.

Porteous, J.; Sebastia, L.; and Hoffmann, J. 2001. On the extraction, ordering, and usage of landmarks in planning. In *Proc. of the 6th European Conf. on Planning (ECP 2001)*, 37–48.

Richter, S., and Westphal, M. 2010. The LAMA planner: Guiding cost-based anytime planning with landmarks. *JAIR* 39:127–177.

Schattenberg, B.; Bidot, J.; and Biundo, S. 2007. On the construction and evaluation of flexible plan-refinement strategies. In *Proc. of the 30th German Conf. on Artificial Intelligence (KI 2007)*, LNAI 4667, 367–381. Springer.

Tsuneto, R.; Nau, D. S.; and Hendler, J. A. 1997. Plan-refinement strategies and search-space size. In *Proc. of the 4th European Conf. on Planning (ECP 1997)*, volume 1348 of *LNCS*, 414–426. Springer.

Zhu, L., and Givan, R. 2004. Heuristic planning via roadmap deduction. In *IPC-4*, 64–66.

Table 2: This table shows the impact of the deployed modification ordering functions on the planning process. While SHOP and UMCP denote strategy function combinations that simulate the respective search procedures, all other strategy implementations use fmf as the plan ordering function. The tests were run on a machine with a 3 GHz CPU and 256 MB Heap memory for the Java VM. *Space* refers to the number of created plans and *Time* refers to the used time in seconds including pre-processing, which takes only a few seconds even for large problem specifications. Values are the arithmetic means over three runs. Dashes indicate that no solution was found within a limit of 5,000 created nodes and a run-time of 150 minutes. The best result for a given problem is emphasized bold, the second best bold and italic.

(a) Results for the UM-Translog domain. Problems with different versions differ in the number and kind of available locations and/or the number of parcels to transport.

Mod. ordering function f^{ModOrd}	Hopper Truck		Auto Truck		Reg. Truck (a)		Reg. Truck (b)		Reg. Truck (c)		Reg. Truck (d)		Flatbed Truck	
	Space	**Time**	**Space**	**Time**	**Space**	**Time**	**Space**	**Time**	**Space**	**Time**	**Space**	**Time**	**Space**	**Time**
da-HotSpot	144	352	644	2077	239	562	114	257	148	352	723	2560	99	237
du-HotSpot	101	224	459	1304	1508	4097	160	460	117	258	–	–	1047	2601
HotZone	55	121	197	527	191	473	55	117	55	137	–	–	159	399
lm_1	***52***	***111***	**133**	**329**	145	**374**	62	135	***53***	**122**	***291***	1172	63	155
lm_1^*	**51**	**109**	***135***	***462***	154	430	**52**	**112**	65	142	**266**	***1162***	**61**	**144**
lm_2	62	162	***135***	464	***141***	469	***53***	123	55	151	339	**1128**	109	315
lm_2^*	124	340	146	489	**137**	413	57	148	**51**	**122**	305	1318	110	308
lcf	55	118	155	470	162	463	78	173	127	222	327	1278	***62***	179
ems	147	295	405	976	211	507	127	262	114	235	–	–	1571	3797
SHOP	89	212	164	433	146	***406***	106	241	83	190	926	4005	98	257
UMCP	58	122	156	474	177	506	55	***113***	57	***127***	308	1263	63	***149***

Mod. ordering function f^{ModOrd}	Armored R-Truck		Auto Traincar (a)		Auto Traincar (b)		Mail Traincar		Refr. Reg. Traincar		Airplane	
	Space	**Time**	**Space**	**Time**	**Space**	**Time**	**Space**	**Time**	**Space**	**Time**	**Space**	**Time**
da-HotSpot	120	359	–	–	184	705	641	2031	588	1958	172	620
du–HotSpot	75	201	–	–	1390	4018	424	1090	307	775	643	2134
HotZone	122	355	–	–	701	1616	81	224	***76***	***196***	345	1323
lm_1	***71***	***177***	**158**	**596**	183	608	**75**	**184**	**72**	**180**	142	441
lm_1^*	**61**	**155**	304	1473	***158***	***543***	***78***	***205***	89	212	189	676
lm_2	73	199	420	1519	211	888	84	248	91	256	***104***	***320***
lm_2^*	81	228	367	1446	**142**	**511**	87	238	84	226	114	436
lcf	86	198	–	–	227	926	79	209	90	225	247	798
ems	113	269	–	–	2558	6447	879	1806	500	1048	784	2517
SHOP	95	227	–	–	247	963	121	274	173	353	150	450
UMCP	75	172	***220***	***739***	161	546	92	229	90	244	**70**	**215**

(b) Results for the Satellite domain. The description "x — y — z" stands for a Satellite problem with x observations, y satellites, and z modi.

Mod. ordering function f^{ModOrd}	1 — 1 — 1		1 — 2 — 1		2 — 1 — 1		2 — 1 — 2		2 — 2 — 1		2 — 2 — 2	
	Space	**Time**	**Space**	**Time**	**Space**	**Time**	**Space**	**Time**	**Space**	**Time**	**Space**	**Time**
da-HotSpot	**56**	**60**	68	78	782	1131	832	1301	2186	6841	**142**	***175***
du-HotSpot	100	107	139	150	–	–	–	–	–	–	–	–
HotZone	***61***	**60**	57	62	1281	4764	–	–	1094	1338	871	1114
lm_1	73	80	194	208	***560***	***652***	***352***	***400***	693	***785***	295	362
lm_1^*	78	85	**34**	**37**	847	969	1803	2569	739	813	619	1228
lm_2	78	86	128	140	4890	5804	**200**	**251**	–	–	483	965
lm_2^*	73	80	91	99	–	–	1905	2553	–	–	***146***	**161**
lcf	86	93	71	77	1120	1338	3022	4069	***407***	**701**	–	–
ems	65	64	47	53	1586	2608	–	–	1219	1579	–	–
SHOP	62	66	105	111	**138**	**155**	–	–	1406	1780	–	–
UMCP	83	91	***36***	***41***	883	1035	1558	1894	**278**	1097	1062	1270