Improving Hierarchical Planning Performance by the Use of Landmarks

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 tasks that occur on every search path leading from the initial plan to a solution. In this work, we present novel domain-independent planning strategies based on such hierarchical landmarks. Our empirical evaluation on four benchmark domains shows that these landmark-aware strategies outperform established search strategies in many cases.

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 (decomposition) 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 executable primitive ones.

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 four different benchmark domains. It turned out that our novel strategies outperform their conventional counterparts on practically all problems if the decomposition hierarchy of the underlying domain is of non-trivial depth.

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). One of the currently best performing planners uses such a landmark heuristic (Richter and Westphal 2010). The work of Zhu and Givan (2004) generalized landmarks to so-called action landmarks. Marthi, Russell, and Wolfe (2008) specify a semantics for HTN planning in which the preconditions and effects of abstract tasks can be interpreted as abstract landmarks, as they are gained via incremental abstraction of more basic tasks.

In the following, we briefly introduce 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 formalization which is the fusion of HTN planning with partial-order causal-link (POCL) planning (Biundo and Schattenberg 2001). A *task* (or *task schema*) $t(\bar{\tau}) = \langle \text{prec}, \text{eff} \rangle$ specifies preconditions and effects 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 (partial) *plan* $P = \langle S, \prec, V, C \rangle$ consists of a set S of *plan steps*, i.e., uniquely labeled task instances, 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 task parameters with other parameters or constants. We denote by $Ground(S, V)$ the set of ground tasks obtained by replacing 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. A task is called *abstract* if at least one method is provided for refining it, otherwise it is called *primitive*. A method $m = \langle t(\bar{\tau}), P \rangle$ relates the abstract task $t(\bar{\tau})$ to the plan P, which is called an *implementation* of $t(\bar{\tau})$.

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 finite sets of tasks and 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 $l{:}t(\bar{\tau}) \rightarrow_{\varphi} l'{:}t'(\bar{\tau}')$ the ordering constraints in $\prec$ ensure that no plan step $l''{:}t''(\bar{\tau}'')$ with an effect that is unifiable with $\neg\varphi$ can be placed between $l{:}t(\bar{\tau})$ and $l'{:}t'(\bar{\tau}')$; (3) the ordering constraints are consistent, i.e., $\prec$ respects the ordering implied by C and it does not induce cycles on S; (4) the variable constraints are consistent, i.e., the (in)equations in V are not contradictory; and (5) all plan steps in S correspond to 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, criteria (2) to (5) guarantee 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 the 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 generic 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** $\texttt{Fringe} = \langle P_1 \dots P_n \rangle \neq \varepsilon$ **do**
2 $\quad F \leftarrow f^{\text{FlawDet}}(P_1)$
3 $\quad$ **if** $F = \emptyset$ **then return** P_1
4 $\quad \langle \texttt{m}_1 \dots \texttt{m}_k \rangle \leftarrow f^{\text{ModOrd}}(\bigcup_{\texttt{f} \in F} f^{\text{ModGen}}(\texttt{f}))$
5 $\quad \texttt{succ} \leftarrow \langle \texttt{app}(\texttt{m}_1, P_1) \dots \texttt{app}(\texttt{m}_k, P_1) \rangle$
6 $\quad \texttt{Fringe} \leftarrow f^{\text{PlanOrd}}(\texttt{succ} \circ \langle P_2 \dots P_n \rangle)$
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 the planning problem Π. Then, for*

any plan $P \in \mathcal{V}_\Pi$, $\mathcal{SolSeq}_\Pi(P) := \{\langle P_1 \dots P_n\rangle \mid P_1 = P$, $P_n \in \mathcal{Sol}_\Pi$, *and for all* $1 \leq i < n$, $(P_i, P_{i+1}) \in \mathcal{E}_\Pi\}$.

Definition 3 (Landmark) *A ground task* $t(\overline{\tau})$ *is called a* landmark *of planning problem* Π*, if and only if for each* $\langle P_1 \dots P_n\rangle \in \mathcal{SolSeq}_\Pi(P_{\text{init}})$ *there is an* $1 \leq i \leq n$*, such that* $t(\overline{\tau}) \in \mathit{Ground}(S_i, V_n)$ *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* $t(\overline{\tau}) \in \mathit{Ground}(S, V)\}$.
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* $\langle P_1 \dots P_n\rangle \in \mathcal{SolSeq}_\Pi(P)$ *there is an* $1 \leq i \leq n$*, such that* $t'(\overline{\tau}') \in \mathit{Ground}(S_i, V_n)$ *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 tasks 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 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 the planning problem* Π *if and only if*

1. $t(\overline{\tau}) \in V_T$ *for all* $t(\overline{\tau}) \in \mathit{Ground}(S, V)$*, for* $P_{init} = \langle S, \prec, V, C\rangle$,
2. *if* $t(\overline{\tau}) \in V_T$ *and if* $\langle t(\overline{\tau}'), \langle S, \prec, V, C\rangle\rangle \in M$ *s.t.* $\overline{\tau}$ *is compatible with* $\overline{\tau}'$ *and* V*, 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* $t'(\overline{\tau}') \in \mathit{Ground}(S, V)$ *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 methods and 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 generally becomes 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* 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:*

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

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

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 disjoint 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.

Let $\Pi = \langle D, s_{\text{init}}, P_{\text{init}}\rangle$ an HTN planning problem with $D = \langle\{t_1(\tau_1), \dots, t_5(\tau_5)\}, \{m_a, m'_a, m_b, m'_b\}\rangle$, $P_{\text{init}} = \langle\{l_1{:}t_1(\tau_1)\}, \{\tau_1{=}c_1\}\rangle$[1], and constants c_1 and c_2, where:

$$m_a := \langle t_1(\tau_1), \langle\{l_1{:}t_3(\tau_1), l_2{:}t_3(\tau_2), l_3{:}t_2(\tau_1)\}, \{\tau_1{\neq}\tau_2\}\rangle\rangle$$
$$m'_a := \langle t_1(\tau_1), \langle\{l_4{:}t_2(\tau_1), l_5{:}t_1(\tau_1)\}, \emptyset\rangle\rangle$$
$$m_b := \langle t_3(\tau_1), \langle\{l_6{:}t_4(\tau_1), l_7{:}t_5(\tau_1)\}, \emptyset\rangle\rangle$$
$$m'_b := \langle t_3(\tau_1), \langle\{l_8{:}t_4(\tau_1)\}, \emptyset\rangle\rangle$$

The TDG for Π is given in Figure 1; the according landmark table is given as follows:

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 of treating landmarks as characteristic, "inevitable" elements on the refinement paths to any solution. The mandatory sets

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

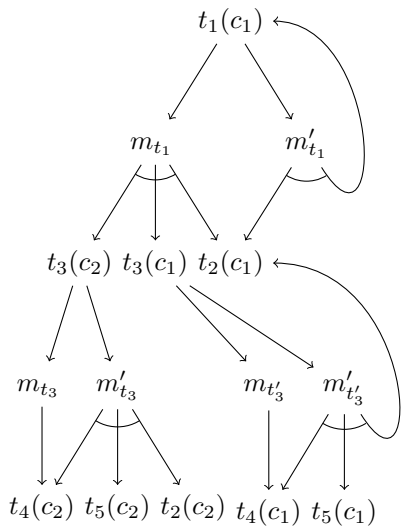

Figure 1: The TDG for the planning problem Π. The method vertices are given as follows:
$m_{t_1} = \langle t_1(c_1), m_{a|_{\tau_1=c_1,\tau_2=c_2}} \rangle$, $m'_{t_1} = \langle t_1(c_1), m'_{a|_{\tau_1=c_1}} \rangle$,
$m_{t_3} = \langle t_3(c_2), m_{b|_{\tau_1=c_2}} \rangle$, $m'_{t_3} = \langle t_3(c_2),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

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 $\langle t(\overline{\tau}), M(t(\overline{\tau})), O(t(\overline{\tau})) \rangle$ denotes that all tasks in $M(t(\overline{\tau}))$ are introduced into the refinement plan, no matter which method is used for decomposing $t(\overline{\tau})$. 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, adding the efforts for all implementations stored in $O(t(\overline{\tau}))$ allows to estimate an upper bound for the "expansion effort".

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 can heavily overestimate 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 fewer 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:*

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

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

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

Considering the previous example, 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" is defined in terms of decompositions that have to be applied until a solution is obtained. The following strategies take into account that primitive tasks contribute to the costs for developing the current plan into a solution, as well. 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 one.

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 our novel strategies on four hierarchical planning domains along two dimensions: (1) we compared the time needed to find a solution in comparison to conventional hierarchical search strategies and (2) these benchmark tests were done on both the original planning domains and the corresponding ones that resulted from applying our landmark-based domain reduction technique (Elkawkagy, Schattenberg, and Biundo 2010). Hence, we base our evaluation on the same benchmark problems, but include two additional domains.

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 transformed 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" concretization of plan steps. We also include 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. HotSpot strategies examine 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 (Schattenberg, Bidot, and Biundo 2007). The *Direct Uniform HotSpot (du-HotSpot)* strategy avoids addressing flaws that refer to HotSpot plan components, and the *Direct Adaptive HotSpot (da)* strategy does so by increasing problem-specific weights on binary combinations of flaw types that occur in the plan. Finally, the *HotZone* strategy takes structural connections between HotSpots into account and tries to avoid modifications that deal with these clusters.

Benchmark Problem Set

We conducted our experiments on three well-established planning domains plus a domain taken from an ongoing research project. *Satellite* is a benchmark from the International Planning Competition (IPC) 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. *WoodWorking*, also originally defined for the IPC in a non-hierarchical manner, specifies the processing of raw wood into smooth and varnished product parts. It uses 13 primitive tasks, 6 abstract tasks, and 14 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. In addition to that, we also employed the so-called *SmartPhone* domain, a new hierarchical planning domain that is concerned with the operation of a smart phone by a human user, e.g., sending messages and creating contacts or appointments. SmartPhone is a rather large domain with a deep decomposition hierarchy, containing 50 abstract, 87 primitive tasks, and 94 methods.

Evaluation of Experimental Results

Tab. 1 shows the time required to solve the problems in our benchmark set for solving the original planning problem specification and the problem posed in a reduced domain model (Elkawkagy, Schattenberg, and Biundo 2010),

respectively. By doing so, we evaluate the search guidance power of our landmark-aware strategies in relation to the domain reduction preprocessing technique.

In the UM-Translog domain (cf. Tab. 1a), at least one of our landmark-aware strategies belongs to the two best performing search strategies in all problem instances, at which lm_1 is clearly dominating the other landmark-based strategies; in fact, in 6 of 14 cases it has the best performance, and in 11 of 14 cases it is one of the two best strategies. 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. A well-informed selection of the decomposition refinements obviously compensates for poor choices on the causality issues.

Our strategies show a similar behavior in the WoodWorking domain (cf. Tab. 1b): In all problems but one either lm_1 or lm_1^* is the best of the evaluated strategies. The similarity of the results to the ones in the UM-Translog domain is not surprising to us, as the depths of their decomposition hierarchies are similar.

The SmartPhone domain (cf. Tab. 1c) is the domain with the deepest decomposition hierarchy. It therefore carries the most information that is exploitable for landmarks, which results in well-informed landmark-aware strategies. Not surprisingly, in all but one problem instances one of our landmark-aware strategies performed best or second-best; in half of the instances, they performed best *and* second-best. The best result achieved lm_1, which was the best strategy in four of five instances and the second-best in another one.

On the Satellite domain (cf. Tab. 1d) our landmark-aware strategies do not clearly dominate any other strategy. Obviously, there is hardly any landmark information available due to the very shallow decomposition hierarchy in this domain. However, no other strategy in this domain dominated any landmark-aware strategy; thus, all evaluated strategies can be regarded as equally good.

An interesting facet of almost all problem instances (even among different domains) is that while the strategies lm_1^*/lm_2^* are the better informed heuristics they repeatedly perform worse than lm_1/lm_2. 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. 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 an abstract task, this refinement puts more constraints on the remaining decisions than the strategy can infer from the feasibility analysis underlying the landmark table.

6 Conclusion

We introduced four novel search strategies for hierarchical planning which base their heuristic guidance on landmark information. We ran experiments on several benchmark domains and compared their performance with the performance of standard search heuristics from the literature. The results show that our landmark-aware strategies outperform the established ones on almost all problems with a deep decomposition hierarchy.

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 ECP 2001*, 157–168.

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

Elkawkagy, M.; Schattenberg, B.; and Biundo, S. 2010. Landmarks in hierarchical planning. In *Proc. of ECAI 2010*, volume 215, 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 AIPS 1994*, 249–254. AAAI Press.

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

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

McCluskey, T. L. 2000. Object transition sequences: A new form of abstraction for HTN planners. In *Proc. of 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 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 KI 2007*, 367–381. Springer.

Tsuneto, R.; Nau, D. S.; and Hendler, J. A. 1997. Plan-refinement strategies and search-space size. In *Proc. of ECP 1997*, volume 1348, 414–426. Springer.

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

Table 1: These results show 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 columns labeled with **red** show the time in seconds needed to solve the problem if our domain reduction technique (Elkawkagy, Schattenberg, and Biundo 2010) is used, whereas columns labeled with **org** show the time needed to solve the *original* (unreduced) problem, respectively. All times include time for pre-processing. All values are the arithmetic means over three runs. Dashes indicate that no solution was found within a limit of 150 minutes. The best result for a given problem is emphasized bold, the second best bold and italic.

(a) UM-Translog: the problems differ in number and kind of locations and/or number of parcels to transport.

Mod. ordering function f^{ModOrd}	#1		#2		#3		#4		#5		#6		#7	
	org	**red**	**org**	**red**	**org**	**red**	**org**	**red**	**org**	**red**	**org**	**red**	**org**	**red**
lcf	1878	225	**198**	173	3020	209	598	470	187	118	5047	1278	267	322
HotZone	***473***	***196***	255	***117***	498	224	549	527	***149***	121	–	–	**171**	***137***
lm_1	**243**	**180**	***221***	135	***447***	**184**	**512**	***434***	**121**	***111***	–	1172	***190***	**122**
lm_1^*	1772	212	593	**112**	**370**	***205***	1592	**420**	657	**109**	–	***1162***	1002	140
lm_2	3311	255	754	123	1670	248	1659	464	716	162	–	**1128**	925	151
lm_2^*	846	226	839	148	991	238	1712	487	583	340	***4921***	1318	1755	**122**
UMCP	952	244	278	114	994	229	***529***	474	187	122	**4893**	1263	215	127
ems	2056	1048	984	262	2199	1806	1889	976	696	295	–	–	876	235
da-HotSpot	2414	1958	350	257	–	2030	4695	2077	589	352	–	2560	578	352
du-HotSpot	1319	775	859	460	987	1090	1904	1304	692	224	–	–	391	258
SHOP	1735	353	283	241	1911	274	–	–	5874	4012	–	4005	911	190

(b) WoodWorking domain: the problems define variations of parts to be processed.

Mod. ordering function f^{ModOrd}	#1		#2		#3		#4		#5	
	org	**red**	**org**	**red**	**org**	**red**	**org**	**red**	**org**	**red**
lcf	2067	350	–	–	–	–	–	–	–	–
HotZone	–	–	–	–	–	418	–	–	–	–
lm_1	***96***	***55***	2396	1815	**171**	159	732	**184**	**564**	**197**
lm_1^*	**82**	**50**	**669**	**245**	614	**98**	1561	1395	2109	1245
lm_2	881	433	–	1259	–	362	–	–	–	–
lm_2^*	1359	403	–	–	–	367	1935	1514	–	893
UMCP	228	133	3207	2936	***259***	125	***618***	***356***	892	218
ems	415	298	–	1275	–	2457	–	2256	–	512
da-HotSpot	113	85	***968***	***828***	355	***110***	**328**	357	***573***	***201***
du-HotSpot	–	–	–	–	–	–	–	–	–	–
SHOP	–	–	–	–	–	–	–	–	–	3578

(c) SmartPhone domain: assisting the user in managing different daily-life tasks.

#1		#2		#3	
org	**red**	**org**	**red**	**org**	**red**
63	40	–	159	***8455***	6827
65	***33***	490	212	–	–
50	**30**	**134**	**53**	–	**465**
65	50	392	173	–	–
60	50	***181***	**53**	–	***680***
98	76	1632	327	–	697
80	**30**	256	***115***	–	–
107	52	235	148	–	–
45	43	–	203	**1747**	1041
52	46	638	166	–	3421
95	73	–	–	–	–

(d) Satellite domain: a column labeled with "x — y — z" stands for a problem instance with x observations, y satellites, and z different modes.

Mod. ordering function f^{ModOrd}	1 — 1 — 1		1 — 2 — 1		2 — 1 — 1		2 — 1 — 2		2 — 2 — 1		2 — 2 — 2	
	org	**red**	**org**	**red**	**org**	**red**	**org**	**red**	**org**	**red**	**org**	**red**
lcf	95	93	154	77	1551	1338	–	4069	–	**701**	–	–
HotZone	76	***64***	142	62	–	4764	–	–	–	1338	–	1114
lm_1	89	80	209	208	***767***	***652***	**458**	***400***	**802**	***785***	***3307***	362
lm_1^*	86	85	***54***	***43***	1024	969	***2617***	2569	***960***	813	–	1228
lm_2	132	86	151	140	–	5804	2816	**251**	–	–	–	965
lm_2^*	102	80	191	99	–	–	2636	2553	–	–	–	**161**
UMCP	91	91	**51**	**41**	2035	1336	4150	1894	1215	1097	**2517**	1270
ems	74	60	62	53	2608	2856	–	–	1756	1579	4484	***175***
da-HotSpot	***69***	67	85	78	2136	1131	–	1131	6850	6841	–	–
du-HotSpot	107	**49**	270	150	–	–	–	–	–	–	–	–
SHOP	**66**	67	113	111	**270**	**264**	–	–	–	1780	–	–