Exploiting Landmarks for Hybrid Planning

Mohamed Elkawkagy  Pascal Bercher
Bernd Schattenberg  Susanne Biundo

Institute of Artificial Intelligence
Motivation

- Landmarks in classical state based planning are facts that have to hold in some intermediate state of every plan that solves the given planning problem.

- Hierarchical Planning
  - Accomplish some set of tasks, rather than to achieve a goal
  - Based on the concepts of tasks and methods
  - High-level tasks are recursively decomposed down to sub-tasks

- Landmarks in hierarchical planning are tasks that occur in any sequence of decompositions leading from the initial plan to a solution plan.
Formal Framework (I)

- **Task**
  - Primitive task → action in state based planning
  - Abstract task → complex task (implemented by primitive tasks)
    
    \[
    t(\tau) = <\text{prec}(t(\tau)), \text{add}(t(\tau)), \text{del}(t(\tau))>
    
    \]
  - Difference: Primitive tasks are executed directly while abstract tasks require a sequence of primitive tasks to be performed

- **Plan**: \( P = <S, C> \)

- **Method**: \( m = <t, P> \)

- **Declarative domain model**: \( D = (T, M) \)
Formal Framework ($\Pi$)

Planning problem specification $\Pi = < D, S_{init}, P_{init} >$

Plan refinement $\rightarrow$ transforming the current plan into a more specific plan

A task that needs to be refined

Possible ways to refine it

Solution Plan of a planning problem $\Pi$ is obtained by refining the initial plan $P_{init}$ into a plan $P = < S, C >$ that has only primitive plan steps and the set of constraints is consistent.

Planning Strategy compares the available plan refinements to choose the most suitable one to refine the current plan.
Our Approach

1) Analyzing task decomposition structure
   - Building Task Decomposition Tree (TDT).

2) Extracting Landmark
   - Identify the essential tasks.

3) Exploiting Landmark information during the planning process
   - Operating on reduced domain model by ignoring unsuccessful decomposition methods.
   - Providing search strategy with focal information
Task Decomposition Tree (TDT): AND/OR tree that represents all possible ways to decompose the abstract tasks of $P_{\text{init}}$ by methods in $D$ until a primitive level is reached or a task is encountered that is already included in an upper level of the TDT.
Operators

Common Task Set Operator: In the TDT, for two methods $m_i$, and $m_j$ of a task $t$, the Common Task Set Operator is defined via:

$$m_i \cap m_j = \text{Tasks}(S_i) \cap \text{Tasks}(S_j)$$

Remaining Task Set Operator: In the TDT, for two methods $m_i$, and $m_j$ of a task $t$, the Remaining Task Set Operator of $m_i$ and $m_j$ is defined via:

$$m_i \hat{o} m_j = \{\text{Tasks}(S_i) \setminus (m_i \cap m_j), \text{Tasks}(S_j) \setminus (m_i \cap m_j)\}$$
Identifying Landmarks

**Landmark Table:** represents a mapping between an abstract task and its subtasks in the decomposition methods that refine this abstract task.

- The intersection $I(t)$ contains those subtasks which occur on every possible path of decompositions that transform $t$ into a primitive plan.
Identifying Landmarks

**Landmark Table:** represents a mapping between an abstract task and its subtasks in the decomposition methods that refine this abstract task.

- The remaining task sets \( R(t) \) (aka options) represent sets of those subtasks that optionally occur when decomposing an abstract task towards a solution plan.
- Every set is indexed by the name of the method which contains these subtasks.
Landmark Extraction

Input: A task decomposition tree TDT.
Output: The filled landmark table LT.

LT ← ∅, infeasible ← ∅
for i ← 1 to TDT.maxDepth() do
   foreach abstract task t in level i of TDT do
      if LT contains an entry for t then continue
   repeat
      Let M be the methods of t in the TDT.
      I(t) ← \bigcap_{m \in M} m
      R(t) ← (\bigcup_{m \in M} m) \setminus \{\emptyset\}
      foreach primitive task t' ∈ I(t) do
         if t' can be proven infeasible then
            remove all m ∈ M from the TDT, including all sub-nodes.
            break
         end
      end
      foreach remaining task set r ∈ R(t) do
         foreach primitive task t' ∈ r do
            if t' can be proven infeasible then
               remove the method m = (t, P), with
               Tasks(P) = I(t) ∪ r from the TDT, including all
               sub-nodes.
               continue
            end
         end
      end
   until no method was removed from TDT
   LT ← LT ∪ \{(t, I(t), R(t))\}
   if I(t) = R(t) = ∅ then
      infeasible ← infeasible ∪ \{t\}
   end
end
return propagate(LT, TDT, infeasible)

It runs iteratively through all levels of the TDT until the maximum level has been reached.
The Methods $M=\{m_1, m_2, \ldots, m_n\}$ that decompose a current task $t$ are collected.

The intersection $I(t)$ and remaining tasks sets $R(t)$ are computed.
The reachability of each primitive task \( t' \) in \( I(t) \) is investigated by estimating the achievability of the preconditions of a task.

If test succeeds: TDT is updated by pruning all methods of \( t \) (this triggers further updates).
The reachability of each primitive task $t'$ in each set $r$ in $R(t)$ is investigated by estimating the achievability of the preconditions of a task. If test succeeds: TDT is updated by pruning failed methods of $t$ (this may trigger further updates).
Landmark Extraction

**Input**: A task decomposition tree TDT.
**Output**: The filled landmark table LT.

1. Initialize:
   
   \[
   \begin{align*}
   LT & \leftarrow \emptyset, \\
   \text{infeasible} & \leftarrow \emptyset
   \end{align*}
   \]

2. For each level \(i\) from 1 to \(TDT.\text{maxDepth}()\):
   
   - For each abstract task \(t\) on level \(i\) of TDT:
     
     - If LT contains an entry for \(t\), continue.
     
   - Let \(M\) be the methods of \(t\) in the TDT.
     
     \[
     I(t) \leftarrow \bigcap_{m \in M} m
     \]
     \[
     R(t) \leftarrow \left( \bigcup_{m \in M} m \right) \setminus \{\emptyset\}
     \]
   
   - For each primitive task \(t'\) in \(I(t)\):
     
     - If \(t'\) can be proven infeasible, remove all \(m \in M\) from the TDT, including all sub-nodes.
     
   - For each remaining task set \(r\) in \(R(t)\):
     
     - For each primitive task \(t'\) in \(r\):
       
       - If \(t'\) can be proven infeasible, remove the method \(m = (t', P)\), with \(Tasks(P) = I(t) \cup r\) from the TDT, including all sub-nodes.
     
   - Until no method was removed from TDT

3. Update LT:

   \[
   LT \leftarrow LT \cup \{(t, I(t), R(t))\}
   \]

4. If \(I(t) = R(t) = \emptyset\), set infeasible = infeasible \(\cup \{t\}\)

Return `propagate(LT, TDT, infeasible)`

**Landmark Extraction**

- Landmark table is updated

<table>
<thead>
<tr>
<th>Landmark</th>
<th>(I(t))</th>
<th>(R(t))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task_1</td>
<td>{t_{11}, t_{12}, \cdots}</td>
<td>{t_{m1}, t_{m2}, \cdots}</td>
</tr>
<tr>
<td>\cdots</td>
<td>\cdots</td>
<td>\cdots</td>
</tr>
<tr>
<td>Task_n</td>
<td>{t_{n1}, t_{n2}, \cdots}</td>
<td>{t_{m1}, t_{m2}, \cdots}</td>
</tr>
</tbody>
</table>

- Recursive procedure is called for propagating the results of the feasibility analysis.
Hierarchical planning refines an abstract task by considering all decomposition methods in the domain model that implement it.

The process of refining abstract tasks in our system is deployed with a reference to the Landmark Table of the planning problem.

It operates on a reduced set of applicable methods according to the respective options $R(t)$ in the Landmark Table.
### Landmark Exploitation – Strategies

- Modification Ordering Functions implement preferences on refinements
- Landmark-Aware Strategy
  - For two given modification $m_i$ and $m_j$, let $f_i$ and $f_j$ be the addressed (abstract task) flaws
  - Let $t_i$ and $t_j$ be the tasks referenced by $f_i$ and $f_j$, then
    \[ R^*(t_i) = \{ r | r \in R(t_i) \text{ for } (t, I(t), R(t)) \in LT \text{ or } r \in R(t') \text{ for } (t', I(t'), R(t')) \in LT, t' \in r', \text{ and } r' \in R^*(t) \}. \]
    
    **Criterion:** $|R^*(t_i)| < |R^*(t_j)|$
  
- Implements the least commitment principle by preferring a lower branching factor estimate
We run our evaluations over two distinguished benchmark domains:

- UM-Translog
  - Logistics, difficulty of problems due to various transportation means.
- Satellite
  - Earth observation, problems become difficult when modeling a repetition of observations: small number of methods is used multiple times in different contexts of the plan.
Evaluation – Model Reduction

UM-Translog

<table>
<thead>
<tr>
<th>Problem</th>
<th>abstr. Tasks (of 21)</th>
<th>Methods (of 51)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regular Truck Problems</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hopper Truck,</td>
<td>12</td>
<td>30</td>
</tr>
<tr>
<td>Auto Truck,</td>
<td>(57%)</td>
<td>(59%)</td>
</tr>
<tr>
<td>Regular Truck-3 Locations</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Regular Truck-2 Region,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Regular Truck-1,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Regular Truck-2,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Various Truck Type Problems</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Flatbed Truck,</td>
<td>12</td>
<td>32</td>
</tr>
<tr>
<td>Armored-R-Truck</td>
<td>(57%)</td>
<td>(63%)</td>
</tr>
<tr>
<td>Traincar Problems</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Auto Traincar,</td>
<td>14</td>
<td>32</td>
</tr>
<tr>
<td>Mail Traincar,</td>
<td>(67%)</td>
<td>(63%)</td>
</tr>
<tr>
<td>Auto Traincar bis,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Refrigerated Regular Traincar</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Airplane Problem</td>
<td>14</td>
<td>37</td>
</tr>
<tr>
<td>Airplane</td>
<td>(67%)</td>
<td>(73%)</td>
</tr>
</tbody>
</table>

Average performance improvement over all strategies and problems is about 40% in search space size and about 30% in CPU time.

Satellite does not benefit significantly from landmark technique due to a shallow decomposition hierarchy.
In all cases, one of the Landmark-Aware strategies outperforms all benchmark candidates.
Conclusion

- Landmark table is generated automatically.
- Avoids unsuitable plan refinements.
- Domain- and strategy independent technique.
- Information helps any hierarchical planner to improve its performance.
- Significance performance gain, especially for problems with a deep hierarchy of tasks.
- … but many open issues left