Landmarks in Hierarchical Planning

Mohamed Elkawkagy **Bernd Schattenberg**

Susanne Biundo

Institute of Artificial Intelligence

Ulm University, Germany

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)

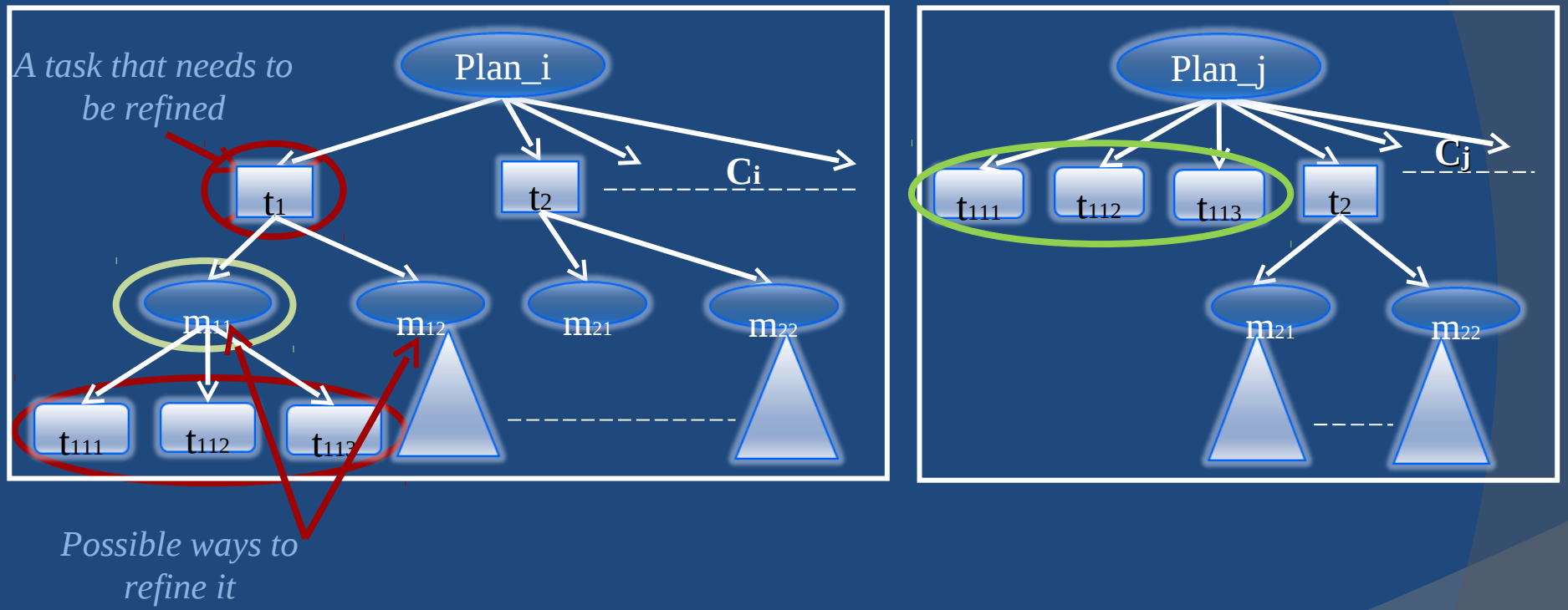
$$t(\tau) = < prec(t(\tau)), add(t(\tau)), del(t(\tau)) >$$

  - 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 (Π)

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

Plan refinement → transforming the current plan into a more specific plan



Solution Plan of a planning problem **Π** is obtained by refining the initial plan P_{init} stepwise into a plan **P = (S, C)** that has only primitive plan steps and the set of constraints are consistent .

Planning Strategy compare the available plans refinement 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

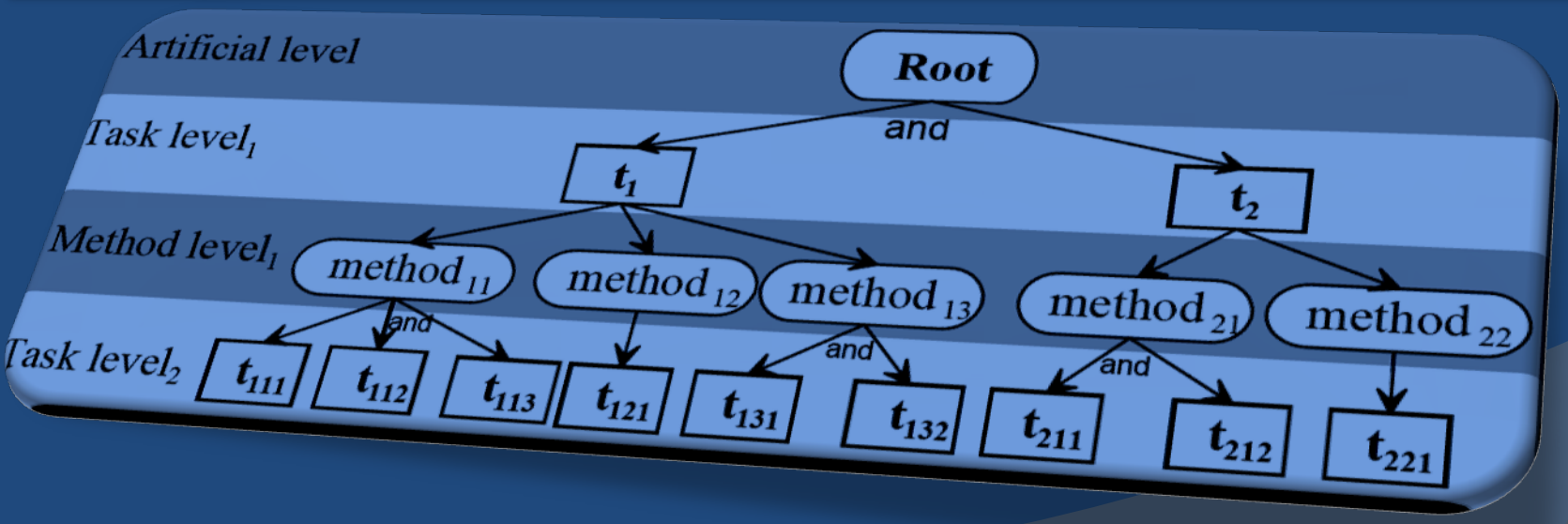
- Identify the essential tasks.

3) Exploiting Landmark information during the planning process

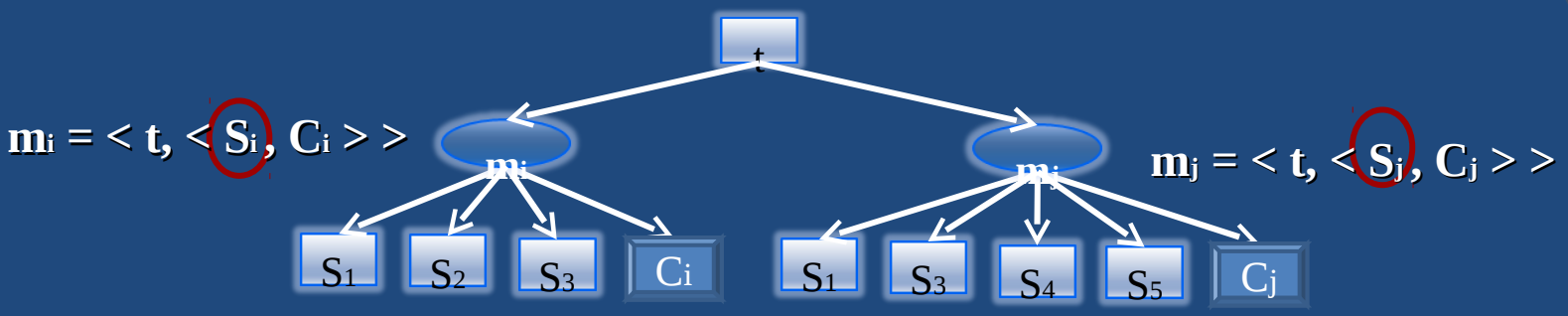
- Operating in reduced domain by ignoring unsuccessful decomposition methods.

Task Decomposition Tree

Task Decomposition Tree (TDT): AND/OR tree that represents all possible ways to decompose the abstract tasks of P_{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 $\hat{\cap}$: For two methods m_i, and m_j of a task t, the Common Task Set is defined as:

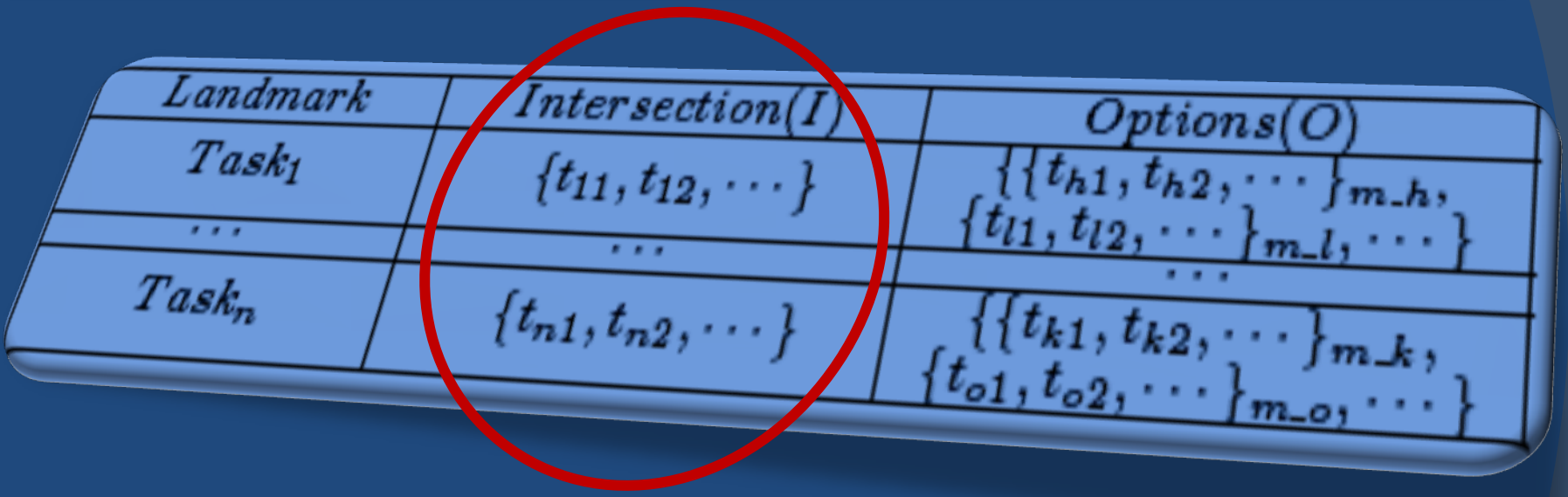
$$m_i \hat{\cap} m_j = S_i \cap S_j$$

Remaining Task Sets $\hat{\square}$: For two methods m_i, and m_j of a task t, the Remaining Task Set of m_i and m_j is defined as:

$$m_i \hat{\square} m_j = \{\{S_i \setminus (m_i \hat{\cap} m_j)\}, \{S_j \setminus (m_i \hat{\cap} m_j)\}\}$$

Identifying Landmarks

Landmark Table : represents a mapping between abstract landmark and the subtasks in the decomposition methods that define the abstract task.

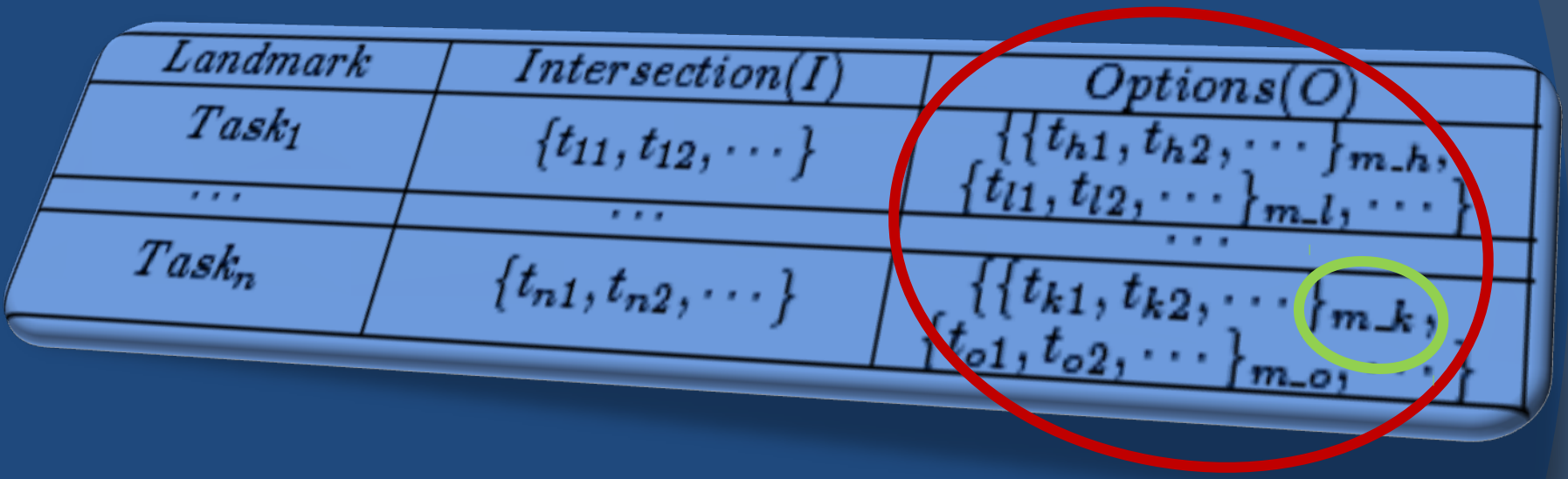


Landmark	Intersection(I)	Options(O)
$Task_1$	$\{t_{11}, t_{12}, \cdots\}$	$\{\{t_{h1}, t_{h2}, \cdots\}_{m\_h}, \{t_{l1}, t_{l2}, \cdots\}_{m\_l}, \cdots\}$
...	...	...
$Task_n$	$\{t_{n1}, t_{n2}, \cdots\}$	$\{\{t_{k1}, t_{k2}, \cdots\}_{m\_k}, \{t_{o1}, t_{o2}, \cdots\}_{m\_o}, \cdots\}$

- 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 abstract landmark and the subtasks in the decomposition methods that define the abstract task.



Landmark	Intersection(I)	Options(O)
$Task_1$	$\{t_{11}, t_{12}, \cdots\}$	$\{\{t_{h1}, t_{h2}, \cdots\}_{m\_h}, \{t_{l1}, t_{l2}, \cdots\}_{m\_l}, \cdots\}$
$\cdots$	$\cdots$	$\cdots$
$Task_n$	$\{t_{n1}, t_{n2}, \cdots\}$	$\{\{t_{k1}, t_{k2}, \cdots\}_{m\_k}, \{t_{o1}, t_{o2}, \cdots\}_{m\_o}, \cdots\}$

- The options **O(t)** represent sets of those subtasks that optionally occur when decomposing the respective landmark task towards a solution plan.
- Every set is indexed by the name of the method which contains these subtasks.

Landmark Extraction

Landmark Extraction(TDT, i, LT)

Initialize: $LT \longleftarrow null, i \longleftarrow 1$
Input : TDT : Task Decomposition Tree,
i : Index of the current level in TDT, LT: LandmarkTable
Output: a LandmarkTable
begin
  if $i \geq maxlevel(TDT)$ **then**
    return LT
  else
    foreach *abstract task t in task level i with* $t \notin LT$ **do**
      $\{m_1, m_2, \cdots, m_n\} \longleftarrow Methods(TDT_i(t))$
      $I(t) \longleftarrow \hat{\cap}_{i=1}^{n} m_i$
      $O(t) \longleftarrow \hat{\cup}_{i=1}^{n} m_i$
      foreach *set* $T \in O(t)$ **do**
        foreach *task tst* $\in T$ **do**
          if *tst is a primitive task with tst is unreachable* **then**
            $TDT \longleftarrow Remove(TDT, ta)$: $\forall$ tasks $ta \in T$
            $O(t) \longleftarrow O(t) \setminus T$
            continue with next set T from $O(t)$.
      $LT \longleftarrow Append(LT, (t, I(t), O(t)))$
    return Landmark Extraction$(TDT, i+1, LT)$
end

It runs recursively through all levels of the TDT until the maximum level has been reached.

Landmark Extraction

Landmark Extraction(TDT, i, LT)

Initialize: $LT \longleftarrow null, i \longleftarrow 1$
Input : TDT : Task Decomposition Tree,
i : Index of the current level in TDT, LT: LandmarkTable
Output: a LandmarkTable
begin
  if $i \geq maxlevel(TDT)$ **then**
    return LT
  else
    foreach *abstract task t in task level i with* $t \notin LT$ **do**
      $\{m_1, m_2, \cdots, m_n\} \longleftarrow Methods(TDT_i(t))$
      $I(t) \longleftarrow \hat{\cap}_{i=1}^{n} m_i$
      $O(t) \longleftarrow \hat{\cup}_{i=1}^{n} m_i$
      foreach *set* $T \in O(t)$ **do**
        foreach *task tst* $\in T$ **do**
          if *tst is a primitive task with tst is unreachable* **then**
            $TDT \longleftarrow Remove(TDT, ta)$: $\forall$ tasks $ta \in T$
            $O(t) \longleftarrow O(t) \setminus T$
            continue with next set T from $O(t)$.
      $LT \longleftarrow Append(LT, (t, I(t), O(t)))$
    return Landmark Extraction$(TDT, i+1, LT)$
end

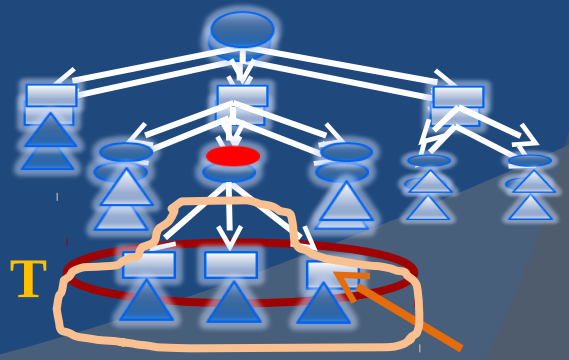
- The Methods M={m_1, m_2, …, m_n} that decompose a current task t are collected.
- The intersection I(t) and Options O(t) sets are computed.

Landmark Extraction

Landmark Extraction(TDT, i, LT)

Initialize: $LT \longleftarrow null, i \longleftarrow 1$
Input : TDT : Task Decomposition Tree,
i : Index of the current level in TDT, LT: LandmarkTable
Output: a LandmarkTable
begin
  if $i \geq maxlevel(TDT)$ **then**
    return LT
  else
    foreach *abstract task t in task level i with* $t \notin LT$ **do**
      $\{m_1, m_2, \cdots, m_n\} \longleftarrow Methods(TDT_i(t))$
      $I(t) \longleftarrow \widehat{\cap}_{i=1}^{n} m_i$
      $O(t) \longleftarrow \widehat{\cup}_{i=1}^{n} m_i$
      foreach *set* $T \in O(t)$ **do**
        foreach *task tst* $\in T$ **do**
          if *tst is a primitive task with tst is unreachable* **then**
            $TDT \longleftarrow Remove(TDT, ta)$: $\forall$ tasks $ta \in T$
            $O(t) \longleftarrow O(t) \setminus T$
            continue with next set T from $O(t)$.
      $LT \longleftarrow Append(LT, (t, I(t), O(t)))$
    return Landmark Extraction$(TDT, i+1, LT)$
end

- The reachability of each primitive task in set T in O(t) is investigated by estimating the achievability of the preconditions of a task.
- TDT is updated by pruning all sub trees with root tasks $\in$ T.



Landmark Extraction

Landmark Extraction(TDT, i, LT)

Initialize: $LT \longleftarrow null, i \longleftarrow 1$
Input : TDT : Task Decomposition Tree,
i : Index of the current level in TDT, LT: LandmarkTable
Output: a LandmarkTable
begin
  if $i \geq maxlevel(TDT)$ **then**
    return LT
  else
    foreach *abstract task t in task level i with* $t \notin LT$ **do**
      $\{m_1, m_2, \cdots, m_n\} \longleftarrow Methods(TDT_i(t))$
      $I(t) \longleftarrow \hat{\cap}_{i=1}^{n} m_i$
      $O(t) \longleftarrow \hat{\cup}_{i=1}^{n} m_i$
      foreach *set* $T \in O(t)$ **do**
        foreach *task tst* $\in T$ **do**
          if *tst is a primitive task with tst is unreachable* **then**
            $TDT \longleftarrow Remove(TDT, ta)$: $\forall$ tasks $ta \in T$
            $O(t) \longleftarrow O(t) \setminus T$
            continue with next set T from $O(t)$.
      $LT \longleftarrow Append(LT, (t, I(t), O(t)))$
    return Landmark Extraction$(TDT, i+1, LT)$
end

- Landmark Table is updated

Landmark	*Intersection(I)*	*Options(O)*
$Task_1$	$\{t_{11}, t_{12}, \cdots\}$	$\{\{t_{h1}, t_{h2}, \cdots\}_{m\_h}, \{t_{l1}, t_{l2}, \cdots\}_{m\_l}, \cdots\}$
...	...	...
$Task_n$	$\{t_{n1}, t_{n2}, \cdots\}$	$\{\{t_{k1}, t_{k2}, \cdots\}_{m\_k}, \{t_{o1}, t_{o2}, \cdots\}_{m\_o}, \cdots\}$
t	**I(t)**	**O(t)**

- Landmark Extraction is called recursively with the modified TDT and updated Landmark Table to inspect the next level of the tree.

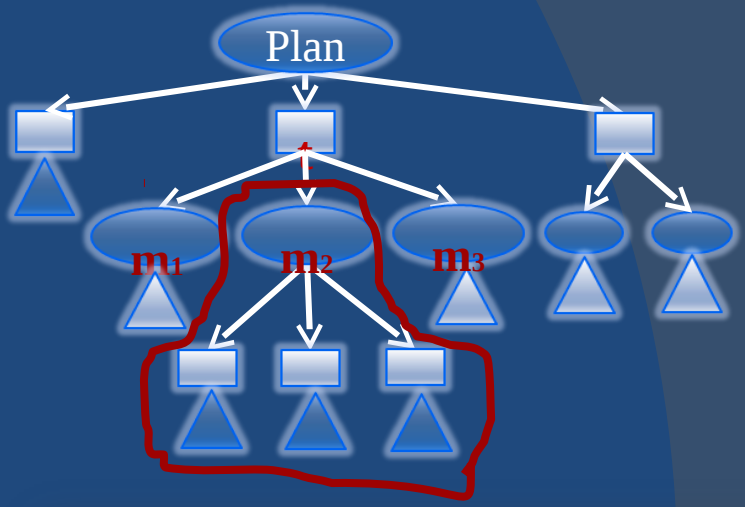
Landmark Exploitation

- Hierarchical planning refines an abstract task by considering all decomposition methods in the domain model that implement it.

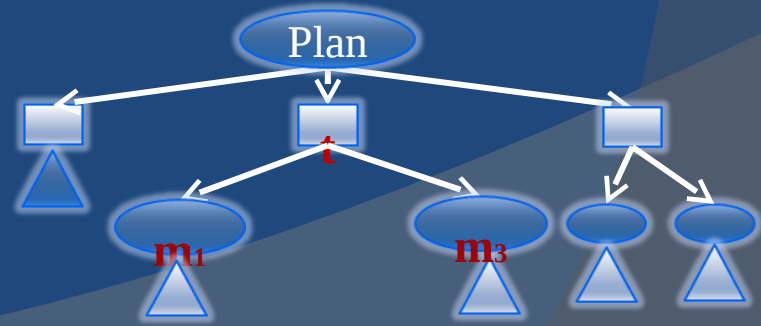
- The process of refining abstract tasks in our system is deployed with a reference to the Landmark Table of the planning problem.

- It operates on a reduced set of applicable methods according to the repective options

O(t) in the Landmark Table



Landmark	*Intersection*(I)	*Options*(O)
$Task_1$	$\{t_{11}, t_{12}, \cdots\}$	$\{\{t_{h1}, t_{h2}, \cdots\}_{m\_h}, \{t_{l1}, t_{l2}, \cdots\}_{m\_l}, \cdots\}$
$\cdots$	$\cdots$	$\cdots$
$Task_n$	$\{t_{n1}, t_{n2}, \cdots\}$	$\{\{t_{k1}, t_{k2}, \cdots\}_{m\_k}, \{t_{o1}, t_{o2}, \cdots\}_{m\_o}, \cdots\}$



Evaluation

- We run our evaluations over two distinguished benchmark domains:

Domain Name	Methods	Abstract tasks	Primitive tasks
UM-Translog	51	21	48
Satellite	8	3	5

- **UM-Translog Domain** describes scenarios of transporting various types of goods by various means (trucks, trains,…) via appropriate infrastructures (roads, transport centers,…).
- The difficulty of UM-Translog problems is due to various transportation means.

Evaluation

- **Satellite domain** manages scientific stellar observations by earth-orbiting instrument platforms .
- The satellite problems become difficult when modeling a repetition of observations, which means that a small number of methods is used multiple times in different context of the plan.

➢ Evaluation factors: -

- Search Space Size (SSS) : The number of plans that are visited for obtaining the first solution.
- CPU time : The total running time of the planning system in seconds.

➢ It is performed with various decomposition planning strategies

Evaluation – UM-Translog Domain

Problem name	Mod. Selection	Plan Selection	PANDA		PANDA+LM	
			SSS	Time	SSS	Time
Flatbed Truck	Lcf+Hz	Fmh+Fmf	**81**	**182**	**58**	**140**
	Lcf+Ems	Fmh+Fmf	**120**	**269**	**90**	**216**
	Lcf+Du	Fhz+Fmf	**96**	**216**	**54**	**129**
	Hz+Lcf	Fhz+Lcp+Fmf	**130**	**299**	**69**	**162**
	Shop Strategy	First plan	**243**	**595**	**98**	**257**
Regular Truck-3-Location	Lcf+Hz	Fmh+Fmf	**149**	**377**	**73**	**203**
	Lcf+Ems	Fmh+Fmf	**234**	**613**	**105**	**206**
	Lcf+Du	Fhz+Fmf	**241**	**483**	**131**	**370**
	Hz+Lcf	Fhz+Lcp+Fmf	**190**	**458**	**115**	**307**
	Shop Strategy	First plan	**163**	**479**	**146**	**406**
Regular Truck – 2	Lcf+Hz	Fmh+Fmf	-	-	**275**	**1237**
	Lcf+Ems	Fmh+Fmf	-	-	**293**	**1144**
	Lcf+Du	Fhz+Fmf	**753**	**2755**	**295**	**1262**
	Hz+Lcf	Fhz+Lcp+Fmf	-	-	**787**	**3544**
	Shop Strategy	First plan	-	-	**926**	**4005**

- The average performance improvement over all strategies and over all problems in the UM- Translog domain is about 40% in search space size and about 30% in CPU time .
- Dashes indicate that the plan generation process did not find a solution within the allowed maximum number of 5,000 plans and 9,000 seconds.

Evaluation – UM-Translog Domain

Problem name	Mod. Selection	Plan Selection	PANDA		PANDA+LM	
			SSS	Time	SSS	Time
Mail Traincar	Lcf+Hz	Fmh+Fmf	**380**	**1241**	**89**	**221**
	Lcf+Ems	Fmh+Fmf	**590**	**1805**	**138**	**313**
	Lcf+Du	Fhz+Fmf	**559**	**1450**	**64**	**160**
	Hz+Lcf	Fhz+Lcp+Fmf	**93**	**213**	**70**	**171**
	Shop Strategy	First plan	**832**	**1911**	**121**	**274**
Refrig. Regular Traincar	Lcf+Hz	Fmh+Fmf	**384**	**1240**	**89**	**215**
	Lcf+Ems	Fmh+Fmf	**634**	**1861**	**138**	**315**
	Lcf+Du	Fhz+Fmf	**446**	**1074**	**64**	**159**
	Hz+Lcf	Fhz+Lcp+Fmf	**92**	**198**	**70**	**172**
	Shop Strategy	First plan	**777**	**1735**	**173**	**353**
Auto Traincar	Lcf+Hz	Fmh+Fmf	**342**	**1137**	**144**	**421**
	Lcf+Ems	Fmh+Fmf	**360**	**1425**	**177**	**477**
	Lcf+Du	Fhz+Fmf	**365**	**1044**	**107**	**328**
	Hz+Lcf	Fhz+Lcp+Fmf	**357**	**958**	**278**	**770**
	Shop Strategy	First plan	**541**	**1282**	**247**	**963**

- The biggest improvements in the transportation tasks that involve special goods and transportation means, e.g., the transport of auto-mobiles, frozen goods, and mail via train saves between 53% and 71%.

Evaluation – Satellite Domain

Problem name	Mod. Selection	Plan Selection	PANDA		PANDA+LM	
			SSS	Time	SSS	Time
1-Obs – 1-Sat – 1-Mod	Lcf+Hz	Fmh+Fmf	**38**	**41**	**37**	**41**
	Lcf+Ems	Fmh+Fmf	**46**	**51**	**46**	**51**
	Lcf+Du	Fhz+Fmf	**67**	**72**	**67**	**72**
	Hz+Lcf	Fhz+Lcp+Fmf	**58**	**62**	**53**	**60**
	Shop Strategy	First plan	**61**	**67**	**57**	**61**
2- Obs – 1- Sat – 1- Mod	Lcf+Hz	Fmh+Fmf	**602**	**788**	**539**	**708**
	Lcf+Ems	Fmh+Fmf	**964**	**1631**	**903**	**1428**
	Lcf+Du	Fhz+Fmf	**1135**	**1319**	**901**	**1030**
	Hz+Lcf	Fhz+Lcp+Fmf	**1468**	**1699**	**1216**	**1474**
	Shop Strategy	First plan	**251**	**270**	**237**	**264**
2- Obs – 2- Sat – 1- Mod	Lcf+Hz	Fmh+Fmf	-	-	-	-
	Lcf+Ems	Fmh+Fmf	-	-	-	-
	Lcf+Du	Fhz+Fmf	-	-	**2821**	**3353**
	Hz+Lcf	Fhz+Lcp+Fmf	-	-	-	-
	Shop Strategy	First plan	-	-	**1406**	**1780**

- The Satellite domain does not benefit significantly from the landmark technique due to its shallow decomposition hierarchy.

Conclusion

- LandmarkTable is generated automatically.
- Avoids unsuitable plan refinements.
- Domain- and strategy independent.
- Help any hierarchical planner to improve its performance.
- Significance performance gain, especially for problems with a deep hierarchy of tasks.