Improving the Performance of Hybrid Planning

Mohamed Elkawkagy

Department of Computer Science
Faculty of Computers and Information
Menoufia University,
Shibin Elkom, Menoufia, Egypt
mohamed.shamseldeen@ci.menofia.edu.eg

ABSTRACT

In recent year, the planning techniques provide a large variety of methods to construct plan and reason about plan component and plan itself. However solving many planning problems are still difficult. In this paper, we propose a new pre-processing technique for hybrid planning - a method that combines reasoning about procedural knowledge and causalities. The proposed technique depends on two modules. Firstly, rule out the irrelevant knowledge from the declarative hybrid planning domain and problem description. Secondly, constructs a transformation technique to convert the hybrid planning domain that is produced from the first step to a classical domain in STRIPS form. In addition, we proved that the plan existence problem is decidable. Furthermore, the experimental studies depend on four different benchmark problems, and the experimental results show that the proposed technique improves the performance of hybrid planning system.

Keywords: Classical Planning, Hierarchical Planning, Reachability Analysis.

Mathematics Subject Classification: 68T20.
Computing Classification System: I.2.8.

1 Introduction

Today, the field of Artificial Intelligence (AI) planning provides many techniques to construct a plan for a given planning problem Π_{cp}. Planning is the process of generating structured actions (called solution plan) that satisfy the desired goal when executed (Nau, Ghallab and Traverso, 2004). In general, the most popular paradigms in AI planning is so-called: classical planning.

Definition 1: (Planning problem in Classical planning Π_{cp}): A planning problem $\Pi_{cp} = \langle IS, A, GS \rangle$ consists of an initial world state (IS), a number of actions (A), and a set of ground literals so-called goal state (GS). A world state (S) is represented by a set of logical facts. The initial world state IS is expressed as a set of literals.

In classical planning, actions are represented by STRIPS (Fikes and Nilsson, 1971).

Definition 2: (Action A): An action A is represented by $\langle pre(a), add(a), del(a) \rangle$. where, (1) $pre(a)$ represents the pre-conditions that must be achieved before the action can be executed.

(2) $add(a)$, and $del(a)$ represent a set of facts so-called post-conditions that change the state of the world after action execution.
Note that $add(a)$ and $del(a)$ respectively represent positive and negative literals as well as $add(a) \cup del(a) = \phi$

Definition 3: (State Change ($App(S, a)$)): Applying the action a in the current state S produces new state $App(S, a)$ as follows:

$$App(S, a) = \begin{cases} (S \cup add(a))/del(a) & if\ pre(a) \subset S \\ S & Otherwise \end{cases}$$

So, the result of applying a sequence of actions $\langle a_1,\ a_2,\ \cdots,\ a_n \rangle$ to the current state s recursively is defined as the following:

$$Result(s, \langle a_1,\ a_2,\ \cdots,\ a_n \rangle) = App(Result(s, \langle a_1,\ a_2,\ \cdots,\ a_{n-1} \rangle), a_n).$$

Therefore, a plan in classical planning is a sequence of actions $A^* \in A$ that transform the initial world state IS stepwise into a state that has the literal goals of GS.
Hierarchical Task Network (HTN) planning (Seegebarth, Muller, Schattenberg and Biundo, 2012) is a widely used planning paradigm. An HTN planning relies on two concepts: tasks and methods. Complex tasks represent compound activities like transporting certain products from specific location to another location. Primitive tasks correspond to classical planning actions. Hierarchical domain models include some decomposition methods for each complex task. Each method provides a task network (partial plan), which specifies a pre-defined (abstract) solution of the corresponding complex task. Hierarchical planning problems are initial task networks. They are refined by incrementally decomposing the complex tasks until the network contains only consistent primitive tasks. The decomposition of a complex task by an appropriate method replaces the complex task by the plan specified by the respective decomposition method. In classical planning, the planning effort is reduced by doing pre-processing for a planning domain model and/or problem description (Hoffmann and Nebel, 2001; Gregory, Cresswell, Long and Porteous, 2001; Haslum, Bonet and Geffner, 2005).
In general, plan existence in the classical planning is known to be decidable, while in HTN planning has been proved undecidable. Using decomposition method in hierarchical planning speed up the process of finding solution plan, but at the same time it causes undecidability. However, hierarchical planner overcome this problem by introducing a set of restrictions on the decomposition methods that cause plan existence decidable (Kutluhan, Hendler and Nau, 1994).
In this work, we introduce a novel technique that achieve decidability without introducing restriction on decomposition methods by introducing a pre-processing technique and then transform the resulted hierarchical domain model into the classical model in the STRIPS-style.
The pre-processing technique was inspired by (Elkawkagy, Schattenberg and Biundo, 2010). They adapted the landmark concept from classical planning which introduced by (Hoffmann, Porteous and Sebastia, 2004; Sebastia, Onaindia and Marzal, 2006; Helmert and Domshlak, 2009; Keyder, Richter and Helmert, 2010) to work in hierarchical planning.
In classical planning, landmarks are defined as a set of facts that must be true at some point in every solution plan for a given problem (Gregory, Cresswell, Long and Porteous, 2004). The

knowledge that gained from extracting landmarks is used in forward search planner to compute heuristic functions (Karpas and Domshlak, 2009; Richter, Helmert and Westphal, 2008). It is also used to investigate their relations to the critical path, and abstraction-heuristics (Bonet and M., 2010; Domshlak, Katz and Lefler, 2010; Keyder et al., 2010; Helmert and Domshlak, 2009). On the other side, landmarks in hierarchical planning are defined as a set of complex or primitive tasks that must exist in any path to a solution plan. In contrast landmarks are also used to generate part of hierarchy (Shivashankar, Alford, Kuter and Nau, 2013) and landmark knowledge is used to generate heuristic search techniques for HTN planning (Elkawkagy, Bercher, Schattenberg and Biundo, 2012). In summary, the performance of classical planners as well as hierarchical planning can significantly improve using landmark information.
In this paper, we introduce a new pre-processing technique and exploit it in the hybrid planning. The hybrid planning paradigm is an integration technique that combine partial-order causal-link planning and hierarchical planning. It is the most applicable paradigm for solving complex real-world planning problems. The benefits of hybrid planning over hierarchical planning paradigm is in a lot of complex real worlds can be encoded in the hierarchical planning, but a lot of parts of the domain might be non-hierarchical and need to be modeled by classical planning paradigm. Fortunately, hybrid planning considers both, in that it allows for the specification of an initial plan and complex tasks as in the HTN planning. In addition, it allows to insert new task to close open pre-conditions as in classical planning. Furthermore, the complex tasks in hybrid planning have pre- and post-conditions. Therefore, they can be inserted into intermediate plans thereby improving the search efficiency (Geier and Bercher, 2011). So, the efficiency of hybrid planning is increased due to the decomposition method (abstract solution) achieving the post-conditions of the relatively complex task. In the hybrid planning Goal description is identified in the hybrid planning problem like in the classical planning. Note that the resulting hybrid planning (Kambhampati, Dattatraya and Srivastava, 1998) integrates HTN planning in a general framework for refinement planning, thereby making use of operator-based techniques. In this view, the algorithm uses reduction schemes where available, and primitive actions otherwise. Causal interaction is also analyzed at the abstract level and refined by mapping conditions and effects of abstract tasks on conditions and effects in their sub-tasks. Abstract conditions are solved by phantom establishers that are identified at a later stage while the hybrid framework proposed here postpones such steps if no suitable task is less abstract enough. Conflict detection and resolution can only be done at the primitive level, as in contrast to our methodology, there is no link between causalities in the different levels of abstraction. Kambhampati addresses user intent by defining a subset of abstract effects explicitly for condition establishment, and by explicitly representing the incompleteness of scheme definitions. For the latter, a specific predicate prevents insertion of new steps. Generally, AI-planning is applied in a lot of domains such as control systems (Precup, David, Petriu, S. and Radac, 2013), removing binary artifacts(Ramírez-Ortegón, Märgner, Cuevas and Rojas, 2013) and power management system (El Sehiemy, Abou El-Ela and Shaheen, 2013), (Tomin, Zhukov, Sidorov, Kurbatsky, Panasetsky and Spiryaev, 2015)
Before introducing the pre-processing hybrid planning technique in section 3 and 4, we will introduce hybrid planning in general and our framework in section. 2. Before concluding this

paper with some remarks in section 6, we will show some experimental results on some benchmark problems in section 5.

2 The Hybrid Planning Framework

Our pre-processing technique is applied on a hybrid planning framework that integrates the features of Partial-Order-Causal-Link (POCL) and HTN techniques (Bercher, Keen and Biundo, 2014; Bercher, Hller, Behnke and Biundo, 2015). POCL planning is a technique used to solve classical planning problems. Plan in POCL consists of a set of partially ordered actions and the dependencies between actions are represented explicitly via causal links. This allows the user to understand the causal structure of the plan.
As opposed to the classical planning, HTN planning represents task by two different kinds: primitive tasks with pre-conditions and post-conditions such as action in classical planning. As well as complex tasks that represent complex activities (abstract task) such as transporting goods, and predefined standard solutions (decomposition methods) of these abstract tasks.
Our framework relies on the ADL language (Pednault, 1989). Therefore, a task schema $t(\tau) = \langle prec(t(\tau)), add(t(\tau)), del(t(\tau))\rangle$ identifies the pre-conditions and post-conditions of a task through a combination between a set of positive and negative literals over the task parameters $\tau = \tau_1, ..., \tau_n$. Note that in the hybrid planning framework, both primitive and complex tasks show pre-conditions and post-conditions, which give us the opportunity to encode POCL planning operations even on complex levels. Note that there may be more than one instance of a task in the same partial plan. To keep these instances apart, we introduce unique identifiers for tasks. The resulting tuple $ps(\tau) = \langle id, t(\tau)\rangle$ is called a plan step, with $id \in N$(natural number).
In our framework, a partial plan is a tuple $P = \langle PS, CS\rangle$, which consists of a set of plan steps PS and a set of constraints $CS = \langle \prec, VC, CL\rangle$. In the partial plan, the set of constraints CS consists of three types of constraints. The symbol $\prec$ represents a partial order on the plan steps. The symbol VC implements the equations of codesignating and non-codesignating parameters that occur in PS with each other and with constants (i.e., $\{v_1 = v_2\}$). Finally a set of causal links is represented by the symbol CL that establish causal relationships among the plan steps PS in a partial plan P. Each causal link has the form $\langle ps_i, Æ, ps_j\rangle$, indicating that $Æ$ is implied by pre-condition and post-condition of plan step ps_j and ps_i respectively. Methods $m(\tau) = \langle ps(\tau), P\rangle$ relate an abstract task $ps(\tau)$ to its implementing partial plan P. In general, each complex task is implemented by multiple methods.
In order to find a solution plan P_{sol} for a given hybrid planning problem, the hybrid planner recursively refines the the initial plan P_{init} into a partial plan $P_{sol} = \langle PS_{sol}, CS_{sol}\rangle$ st., $CS_{sol} = \langle \prec_{sol}, VC_{sol}, CL_{sol}\rangle$.
Definition 4:(Hybrid planning problem): A hybrid planning problem $\Pi_{hyb} = \langle D, P_{init}, IS, GS\rangle$ consists of a domain model D, an initial state IS, a goal state GS like classical planning, and an initial partial plan P_{init}. A domain model $D = \langle T_p, T_C, M\rangle$ formed from a finite set of primitive task schemata T_p, a finite set of complex task T_C and a set of decomposition methods M.
Definition 5:(Solution plan P_{sol}): A solution plan $P_{sol} = \langle PS_{sol}, CS_{sol}\rangle$ consists of a set of

plan steps PS_{sol} and a set of constraints $CS_{sol} = \langle \prec_{sol}, VC_{sol}, CL_{sol} \rangle$ such that ordering constraints $\prec_{sol}$, variable constraints VC_{sol} and causal link constraints CL_{sol}.
The solution plan have to achieve the following solution criteria:
1) P_{sol} is a refinement of P_{init} and PS_{sol} contains only ground instances of primitive task schemata. 2) All precondition of a plan step in PS_{sol} is supported by a causal link in CL_{sol}, 3) The ordering constraints in $\prec_{sol}$ does not include a cycle on the plan steps PS_{sol} 4) None of the causal links in CL_{sol} is threatened. A causal threat is the situation in which the partial order of a plan would allow a plan step ps_k with a post-condition that implies $\neg Æ$ to be ordered between two plan steps ps_i and ps_j for which there is a causal link $\langle ps_i, Æ, ps_j \rangle$.
To find a solution plan, the planner explore a plan space of plan refinements. Refinement process includes recursively decompose the abstract tasks by their decomposition methods; Causal links can be solved by adding open pre-conditions of plan steps or by adding ordering and variable constraints. We call such a refinement process a plan modification.
Our planning algorithm (Algorithm 1) takes the planning problem Π_{hyb} as an input and updates it step by step until a solution plan is found.

Algorithm 1: *Hybrid planning Algorithm*

```
Input  : Planning Problem $\Pi_{hyb}$,
         Fringe $F \longleftarrow P_{init}$
Output: Solution plan or Failure
1 while (F not empty) do
2     P ⟵ PlanSel(F)
3     F ⟵ (F) \ P
4     if (Flaws(P) is empty) then
5         return P
6     f ⟵ FlawSel(Flaws (P))
7     F ⟵ F $\bigcup \{apply(m(f), p)\}$
8 return Failure
```

For a given hybrid planning problem Π_{hyb}, our planner performs a search in a plan space. The fringe of the algorithm is a data structure used to maintain a sequence of plans $P_1 \dots P_n$ that are ordered for further consideration. This means it contains all non-visited plans that are direct successors of visited non-solution plans (intermediate plan). The used search strategy decides which plan can lead to a solution such as P_i leads to a solution plan more quickly than a plan P_j where $j>i$. Therefore, after selecting a partial plan P from a fringe using the module **planSel** the plan P is removed from the fringe set.
The planning algorithm works recursively as long as no solution is found, and there are still plans in the fringe needed to refine (line 1). The selected plan via plan selection module **PlanSel** is removed from the fringe F (line 3). Hence, all flaws in the selected plan are determined via flaw selection module **FlawSel** (line 6). A flaw is defined as a set of plan component that causes violation in a solution criterion such as (1) complex task flaw: the presence of an abstract task raises a flaw that includes that task, and (2) Threat flaw: a causal threat consists of the causal link and the threatening plan step. In case of there is no flaws in the current plan,

and all solution criteria are satisfied then the algorithm is terminated and return the current plan as a solution (line 4 to 5).
Note that a plan selection **planSel** orders the generated plans in the fringe. During this step plans that have unresolvable flaws such as inconsistent ordering of tasks are removed from the fringe. The planner is terminated successfully with the solution plan or returns with failure in case of the fringe becomes empty, and no solution exist.
For a single flaw f, there may be a lot of possibilities to solve it such as for an open precondition, there might be several actions that can be used as a producer for the respective causal link. Each possibility called a modification m to the current plan P. Those modifications are computed and applied to the current plan P that lead to a set of successor plans (line 7).
The proposed framework defines the control search strategy in an explicit manner. In this framework, there are two control strategies modification and plan search strategy. The modification selection strategy $FlawSel()$ decides, which branches to visit first. In this way, the plan selection strategy $PlanSel()$ is used to prioritize the plans; several strategies can be concatenated into cascades. This means that our framework can be used to construct a rich variety of planning strategies.

3 Our Approach

The proposed approach depends on two phases:
1. Using hybrid domain model and problem description we will build a Decomposition Graph (DG), and hence perform the pruning technique on it to rule out domain parts that might be irrelevant to the given planning problem. ***(Pruning phase)***
2. The new domain model that is produced from the previous phase will be translated into a classical planning domain by translating each occurrence of a task either complex or primitive into operators of classical planning. ***(Transformation phase)***
Now we will start by identifying some terminology based on DGs that is used in our preprocessing technique. For a given planning problem $\Pi_{hyb} = \langle D, P_{init}, IS, GS \rangle$, a DG is defined as an AND/OR graph. A DG represents all possible decompositions methods that break down the complex tasks in the initial plan P_{init}. So, a DG is represented by $DG = \langle V_T, V_M, E \rangle$. The symbol V_T represents task vertices (AND node) that includes all ground tasks either complex or primitive which are produced from decomposing tasks in the initial plan Pinit. V_M represents vertices of decomposition method (OR node) that solve the complex tasks in the VT and finally E represents the connection edges between different vertices from V_T to V_M or from V_M to V_T.
In general, task node in the DG contains a number of ground tasks instead of the concrete task in the Task Decomposition Tree (TDT) (Elkawkagy et al., 2010), and it is used as a basis for a set of search strategies. Therefore, DG has an enormous number of ground tasks and method instances that can be used to decompose the complex tasks. Therefore, to build a finite DG we will neglect the tasks that are already encountered in the DG. Therefore, tasks in the DG are mandatory tasks or optional tasks. Mandatory tasks are the tasks that must occur in all decomposition methods of a certain abstract task. These tasks are used as an intermediate step to extract another set of tasks called optional tasks. The optional tasks are used to decide

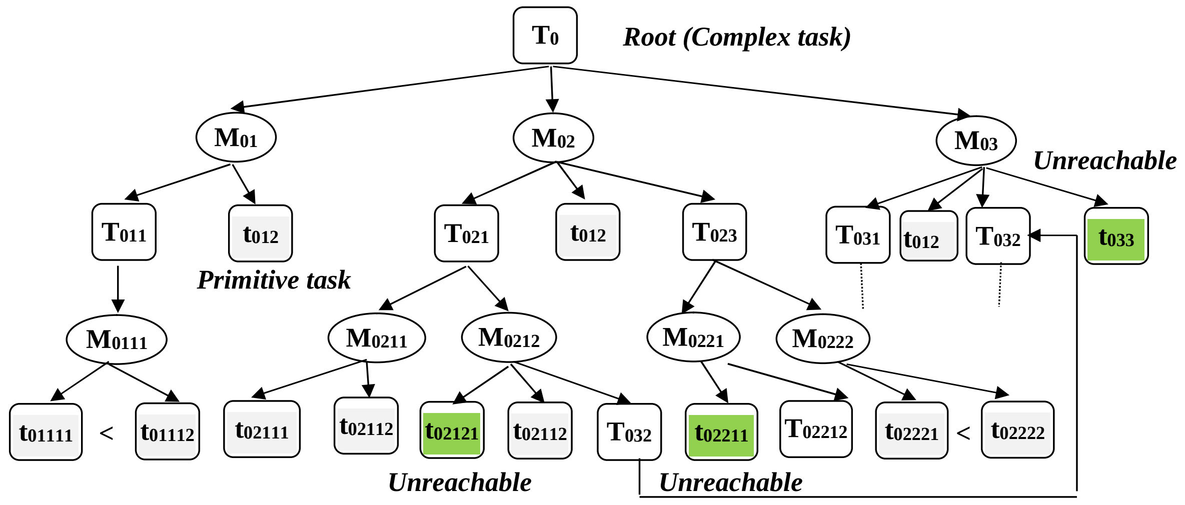


Figure 1: DG before applying pruning algorithm: DG is formed from decomposing the initial plan recursively until reach to primitive nodes. Gray color: primitive tasks; Green color: Unreachable task

which parts are irrelevant respect to the given planning problem.
To eliminate the irrelevant parts from the DG, we will use the Pruning algorithm (Algorithm 2) which rely on the concept of mandatory and optional tasks. Mandatory tasks help us to identify decomposition methods that solve the specified complex tasks. To identify mandatory tasks, we firstly define the set of subtasks $PS(t(\tau'))$ of a ground abstract task $t(\tau')$ where τ' represents task parameters. For simplicity suppose the DG in the Figure 1. For each decomposition method m, for a complex task $t(\tau')$, $PS(t(\tau')) = \{t'(\tau''), \exists m | t'(\tau'') \in E) \& (t(\tau'), m) \in E\}$ includes a set of all tasks in the set of subtasks in the task network referenced by a decomposition method m. Therefore according to the DG in Figure 1,

$$PS(T_0) = \{\{T_{011}, t_{012}\}, \{T_{021}, t_{012}, T_{023}\}, \{T_{031}, t_{012}, T_{032}, t_{033}\}\},$$
$PS(T_{011}) = \{\{t_{01111}, t_{01112}\}\}$, and $PS(T_{023}) = \{\{t_{02211}, T_{02212}\}, \{t_{02221}, t_{02222}\}\}$.

The set of mandatory tasks $\mu(t(\tau'))$ of a ground complex task $t(\tau')$ is then calculated by:

$$\mu(t(\tau')) = \bigcap_{ps \in PS(t(\tau'))} ps$$

The set $\mu(t(\tau'))$ includes tasks either primitive or complex which exist in all decomposition methods that solve the complex task $t(\tau')$. Therefore, $\mu(t(\tau'))$ represents the lower bound for estimating the decomposition effort of solving abstract task $t(\tau')$. In our example, $\mu(T_0) = \{t_{012}\}$, $\mu(T_{021}) = \{t_{02112}\}$, $\mu(T_{023}) = \phi$, $\mu(T_{011}) = \{t_{01111}, t_{01112}\}$.
The set of remaining tasks $\gamma(t(\tau'))$ identify the set of all tasks that still exist in the set of subtasks $PS(t(\tau'))$ and not exist in the mandatory set .

$$\gamma(t(\tau')) = \bigcup_{ps \in \{PS(t(\tau')) \setminus \mu(t(\tau'))\}} ps$$

In our example, $\gamma(T_0) = \{\{T_{011}\}, \{T_{021}, T_{023}\}, \{T_{031}, T_{032}, t_{033}\}\}$.
Now we are ready to demonstrate the algorithm (Algorithm 2) that shows how to analyze the given planning problem and domain model to identify unreachable tasks and then to remove

the respective tasks from the created DG. It takes a DG, as input and runs recursively to identify the irrelevant parts that will be pruned from the DG.
The pruning algorithm (Algorithm 2) computes the mandatory set and optional sets γ for each complex task in the DG. The algorithm takes a DG, which is computed previously before calling the algorithm, as input and returns a DG after removing irrelevant parts. It is depend on the type structure of the domain model of the planning problem and identifies whether some preconditions of a primitive task can never be satisfied such as in (Fox and Long, 1998). The proposed reachability analysis starts with the root of the DG (tasks in the initial plan) and go further until the frontier becomes empty. So, the pruning algorithm uses a relaxed reachability analysis to test the reachability of all primitive tasks either in the mandatory set or optional sets. If a task is unreachable, the decomposition method introducing this task is pruned from the DG and all its sub-nodes (and so forth). After all infeasible methods of a complex task t have been pruned from the DG, this task, its intersection, and the remaining complex tasks are stored into the frontier for the next examination.
Now, we will show how to achieve this task by our algorithm (Algorithm 2): First, the frontier queue F, visited V, and unreachable set gets initialized (line 1 to 3). Afterward, each complex task, which is not yet examined (not stored into the visited set) is considered (line 4 to 6). To avoid examining a task twice, the current task is removed from the frontier and add it to the visited set(line 7 to 8). Moreover, hence, for the current complex task at hand, line 9 calculates the mandatory tasks $\mu(t)$ and the optional tasks sets $\gamma(t)$ in the unpruned DG.
The pruning algorithm is interested by the tasks that are remaining. Therefore, to compute the optional set γ, the empty sets are neglected. After that, the reachability analysis is performed for each primitive task. First, for each primitive task exist in the mandatory task $\mu(t)$ (line 10). If such a task is infeasible, then all methods including t become irrelevant and can hence be pruned from the DG (line 11 to 14). After this test, each optional task set is tested for reachability.
If a task is found infeasible, only this specified method gets pruned from the DG (line 16 to 23). After that, add each new complex task either in mandatory (line 17 to 18) or optional sets (line 25 to 27) to frontier F for further examination. Finally, when storing an entry in unreachable set (line 28 to 29) recursively remove again all methods containing this task as well as removing its parents and so on.

Algorithm 2: *Pruning Algorithm*

```
Input  : A decomposition graph DG
Output: The pruning decomposition graph DG
1  $Unreachable \longleftarrow \phi$
2  $Visted\ V \longleftarrow \phi$
3  $Fringe\ F = newQueue(root)$
4  while (F is not empty) do
5      for each complex task t ∈ F do
6          if V contains t then continue
7          $V = V \cup t$
8          F.pop(t)
9          Compute the mandatory $\mu$ and Optional sets $\gamma$
           $\mu(t(\tau')) = \bigcap_{ps \in PS(t(\tau'))} ps$
           $\gamma(t(\tau')) = \bigcup_{ps \in \{PS(t(\tau')) \setminus \mu(t(\tau'))\}} ps$
10         for each primitive task $t' \in \mu$ do
11             if (infeasible (t') = = true) then
12                 Remove all methods m ∈ M from the DG including all sub-nodes
13                 $Unreachable = Unreachable \cup t'$
14                 break
15
16         for each complex task $ct \in \mu$ do
17             F.push(ct)
18         for each optional task set $ots \in \gamma$ do
19             for each primitive task $pt \in ots$ do
20                 if (infeasible (pt) = = true ) then
21                     Remove the method $m = \langle pt, P \rangle$, with $Tasks(P) = \mu(pt) \cup \gamma$ from the DG,
                       including all sub-nodes
22                     $Unreachable = Unreachable \cup pt$
23                     continue
24
25             for each complex task $oct \in ots$ do
26                 F.push(oct)
27 for each task t ∈ Unreachable set do
28     recursively find all parents of task t
29     remove these parents from DG
30 return DG
```

4 Transforming Hybrid Domain

A hybrid planning domain will be translated into a classical planning domain by considering the DG that is produced from execution of the Algorithm 2 on the initial hybrid planning domain. This will be done by translating each ground complex task as well as translating each ground primitive task as described below.

Translating ground of complex tasks: Each ground complex task $t(\tau')$ will be translated into A new operator $t_{new}(\tau')$ with STRIP-style as follows:
(1) The execution of a complex task $t(\tau')$ is completed if it is solved by one of its decomposition methods. The decomposition method is accomplished if and only if all its sub-tasks $(i.e, t_1, t_2, ..., t_k)$ are achieved. Therefore, to ensure the execution of the respective method, a new task is built, for each decomposition method, instead of a decomposition method. The new task is so-called $TaskRef - new - MethodName$. The pre-conditions of this task are considered as artificial literals. Each literal represent task in the method sub-tasks. These artificial literals are so-called $t_1 - solved, t_2 - solved, ..., t_k - solved$.
(2) Sometimes, sub-tasks are ordered such as $t_1 < t_2$ and $t_2 < t_3$. Therefore, to achieve these ordered constraints the effect and artificial literal (i.e., post-condition(t_3) and $t_3 - solved$) of the last task are considered the precondition of a new task.
(3) A single artificial post-condition $TaskRef-achieved$ is added to each new task. This means that the complete execution of a task is achieving the respective decomposition method.
(4) In general, each complex task can be solved by some decomposition methods, and then a number of new tasks are created according to the number of decomposition methods. All of these new tasks have the same post-condition (i.e., $TaskRef - achieved$).
(5) Finally, the pre-condition of a $t_{new}(\tau')$ is also the artificial post-condition for all tasks $TaskRef - new - MethodName$ and its post-condition is $TaskName - solved$.

Translating ground of primitive tasks: Each ground primitive task $t(\tau')$ is transformed as follows: (1) A new task $t_{new}(\tau')$ is build without any updating in its pre-conditions.
(2) In addition to the original post-condition of the respective ground primitive task, an artificial literal $t_{new} - solved$ is added to this post-condition.
(3) The ordering constraints between ground sub-tasks are translated by adding pre-conditions between sub-tasks. Such as the ordering constraints between sub-tasks $(t_i < t_j)$ is represented in the transformed domain by adding the artificial literal $t_1 - new - solved$ to the pre-conditions of predecessor task t_2. If sub-task t_2 is a complex task, the literal $t_1 - new - solved$ will be added to every sub-task existing in the decomposition method that solve the complex task t_2.
To ensure that the new task either primitive or complex is used at most once in any solution plan, we add a literal $\neq t_{new} - achieved$ to the pre-condition of the respective task.

Translating a hybrid planning problem: A hybrid planning problem $\Pi_{hyb} = \langle D, P_{init}, IS, GS \rangle$, is transformed into a new STRIPS-style planning problem $\Pi'_{hyb} = \langle D', \phi, IS', GS' \rangle$ as follows:
(1) The proposed transformation technique is used to convert the original planning domain to the STRIPS-style domain model. So, the original domain $D = \langle T, M \rangle$ is translated to a new form $D' = \langle T', \phi \rangle$. The symbol T' represents all tasks as well as decomposition methods M in the original domain model.

(2) All sub-tasks in the initial plan P_{init} are translated by adding a new task, the so-called $t - solve$ to the domain model D'. The pre-conditions of $t - solve$ is the post-condition of all root tasks in a DG and its post-condition is a new literal, so-called $t - solve - achieved$. Note that, adding new task $t - solve$ means that the execution of all tasks in the initial plan P_{init} is completed.
(3) In addition the set of literals in the original goal, a new literal $t - solve - achieved$. which assure a complete execution of the complex task in the root, is added.

$$GS' = GS \bigcup \{t - solve - achieved\}$$

(4) The new initial state IS' is represented by the original initial state IS. It is extended by the negative literals in the tasks in the new domain model D' as well as literal $\neg t - solve - achieved$.

$$IS' = \{IS \bigcup \{\neg t_{new} - achieved | t \in T' and\, t_{new} - achieved \notin IS\} \bigcup \{\neg t - solve - achieved\}\}$$

Finally, we ruled out the irrelevant parts, which cannot reach to the solution plan, from a DG. After that, a hybrid domain model is translated into a domain model with the style-STRIPS domain model.
As well as a hybrid planning problem is converted to a new planning problem with style-STRIPS (classical) planning problem.
Correctness. To proof the correctness of the proposed transformation technique, we need to define proof that there is a one to one and onto mapping function between the hybrid planning problem $\Pi_{hyb} = \langle D, P_{init}, IS, GS \rangle$ and the translated classical planning problem $\Pi'_{hyb} = \langle D', IS', GS' \rangle$ in the form of STRIPS-style.
Theorem 1: Let $\Pi_{hyb} = \langle D, P_{init}, IS, GS \rangle$ be any hybrid planning problem. Let δ is the set of all derivations of solutions for Π_{hyb}. While the notation Ω represents all solutions for translated planning problem $\Pi'_{hyb} = \langle D', IS', GS' \rangle$. If the set of methods M and the set of tasks T is complete (i.e., each complex task in the domain is identified by at least one decomposition method), then there is a one to one correspondence that maps δ onto Ω.
Proof: We need to define a mapping function $F : \delta \to \Omega$ and show that F is one-to-one and onto. Induction can do this.
Let Ω be a solution for a translated planning problem Π'_{hyb} with σ that is a sequence of actions and method instances that constitute the solution plan Ω. In particular, σ is a concatenation of subsequences $\sigma_1; \ldots; \sigma_k$ corresponding to subtasks $t_1; \ldots; t_k$. Therefore, let $F(\sigma) = Concatenate\,(F(\sigma_1), \ldots, F(\sigma_k))$, where each $F(\sigma_i)$ for all $(1 \leq i \leq k)$ is defined recursively. In general, $F(\sigma_i)$ is defined in two cases as follows:
Case 1: suppose $\sigma_i = \sigma_{i1}; \ldots; \sigma ik$ and its derivation is identified by $\sigma'_i = \sigma'_{i2}; \ldots; \sigma'_{ik}$, if σ_{i1} is primitive task then $F(\sigma_i)$ is defined as a concatenation between the translation of task σ_{i1} and the mapping function of σ'_i.
Case 2: if σ_{i1} is a complex task. So, suppose the respective task is decomposed by decomposition method m and substitution Θ then σ'_i is identified by the concatenation of the sequence of derivations $\sigma'_{i1}; \ldots; \sigma'_{ij}$ that are produced from the method subtasks. Therefore, $F(\sigma_i) = Concatenate \left(m'_0\Theta, m'_1\Theta, F(\sigma'_{i1}), \ldots, m'_j\Theta, F(\sigma'_{ij}) \right)$. Where $m'_1, \ldots, m'_j$ are the translated tasks in the new domain that is corresponding to the sub-tasks in the decomposition method.

Theorem 2: The time and space complexity of computing the translation planning problem Π'_{hyb} are both $O(\alpha + \beta)$. Where α and β respectively are the size of tasks and the recursive decomposition of methods M in the domain model D.
Proof: The translation of each task $t \in T$ (either T_A or T_C), is a single task. The computation of this task is a linear-time scan of task T, and it can be seen by examining the size of that task is $\alpha = O(|T|)$. In case of there is no recursive decomposition for methods M. This means that $\beta = O(|M|)$ in this case. For each $m \in M$, the translation of methods M is produced by a linear-time scan of m, and it can be seen by examining the set of methods $O(|m|)$. On the other hand, if there is recursive decomposition for methods M, then β is given as input and it is a fixed number. Thus, the theorem follows.
Example: To illustrate our pre-processing and transformation approaches, assume the following artificial example in Figure 1. Note that the complex and primitive tasks are represented by capital and small letters respectively. The oval shape represents decomposition methods and symbol $\prec$ represents ordering constraints between sub-tasks. Assume that the root of a DG has only one complex task T_0 can be decomposed by three decomposition methods $M_{01}, M_{02}, and M_{03}$. By applying the proposed pruning algorithm (Algorithm 2), suppose we found that the primitive tasks t_{033}, t_{02121}, and t_{02211} are unreachable. Therefore, the branches that contain these tasks will be pruned. Then a new DG is produced as shown in Figure 2. After that, the tasks and decomposition methods in the new DG are translated according our approach as follows: Table 1 shows segment from the domain translation according to our approach. The decomposition methods will be converted to new tasks so-called $T_0 - new - M_{01}$ and $T_0 - new - M_{02}$. They have the same effect $T_0 - achieved$ but with different pre-conditions. These preconditions confirm the execution of the sub-tasks in the respective decomposition method. Therefore, the pre-condition of the first new task $T_0 - new - M_0$ is post-condition of the primitive sub-task $t_{012} - new$ and the artificial effect $T_{011} - solved$ of abstract task T_{011}.
On the other hand, the decomposition method M_{0111} will be converted to task $T_{011} - new - M_{0111}$ with pre-condition $eff(t_{1112})$, and its post-condition is $T_{011} - achieved$. This is because task t_{01111} is ordered before t_{01112} then in our transformation approach the post-condition of task t_{01111} is added to the pre-condition of task t_{01112}. This means that the task t_{01112} cannot be performed before task t_{01111} is completed firstly.

5 Implementation and Experiments

To measure the performance gained by our approach practically, we run our planning framework with a series of problems over different domains. The experimental runs on a machine with a 3 GHz CPU and 256 MB Heap memory for the Java VM. Note that this machine has only one single processor unit.
The evaluation of our approach depends on two directions. Firstly, we compared the time needed to find a solution plan using the proposed technique (Translation-DG) with conventional hierarchical planning (HP), pruning hierarchical (PHP)(Elkawkagy et al., 2010), SHOP (Nau, Tsz-Chiu, Ilghami, Kuter, Murdock and Yaman, 2003), which prefers task expansion

Table 1: Transforming the hierarchical domain model into STRIPS operators

Original tasks And Methods	Translated Tasks in the STRIPS-style form
T_0	Name: $T_0 new$ Pre: $T_0 - achieved$ post: $T_0 solved$
M_{01}	Name: $T_0 - newM_{01}$ Pre: $post(t_{012} - new)$ $T_{011} - solved$ post: $T_0 achieved$
T_{011}	Name: $T_{011} - new$ Pre: $T_{011} - achieved$ post: $T_{011} solved$
t_{012}	Name : $t_{012} new$ Pre: $pre(t_{012})$ post: $post(t_{012}) \cup t_{012} solved$
M_{0111}	Name : $T_{011} - newM_{0111}$ Pre : $post(t_{01112} - new)$ post: $T_{011} - achieved$
t_{01111}	Name : $t_{01111} - new$ Pre: $pre(t_{01111})$ Eff: $post(t_{01111}) \cup t_{01111} - solved$
t_{01112}	Name : $t_{01112} new$ Pre : $pre(t_{01112}) \cup post(t_{01111} - new)$ Eff: $post(t_{01112}) \cup t_{01112} - solved$
etc.	*etc.*

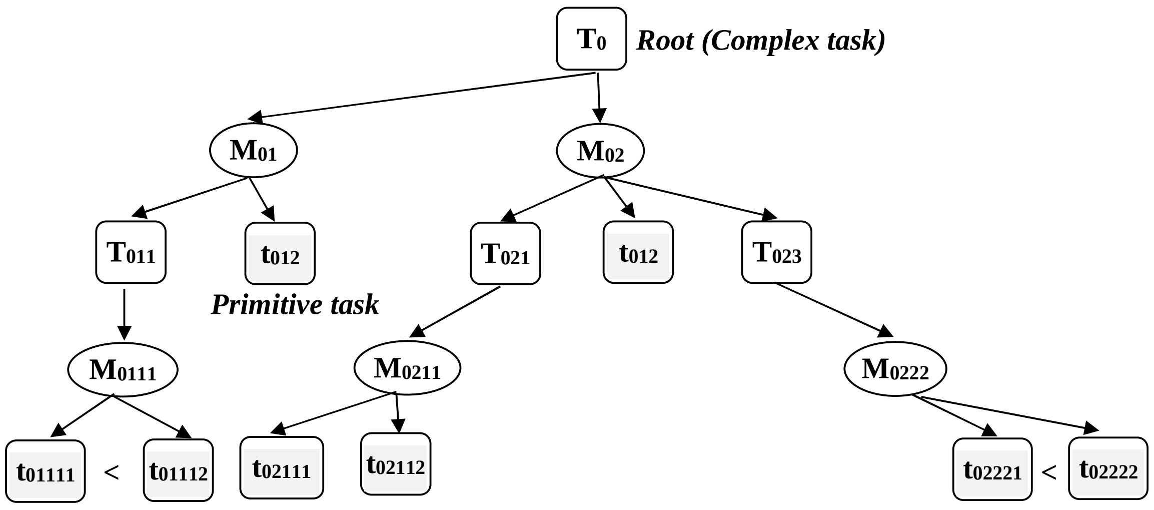


Figure 2: DG after applying pruning algorithm: New DG after ruling out the irrelevant nodes that never lead to a solution plan.

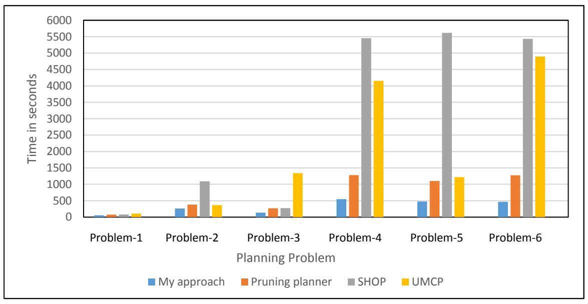
Table 2: Domain Model Hierarchy

Domain Name	Complex Task	Decomposition Method	Primitive Task
UM-Translog Domain	21	51	48
Satellite Domain	3	8	6
WoodWorking Domain	6	14	13
SmartPhone domain	50	94	87

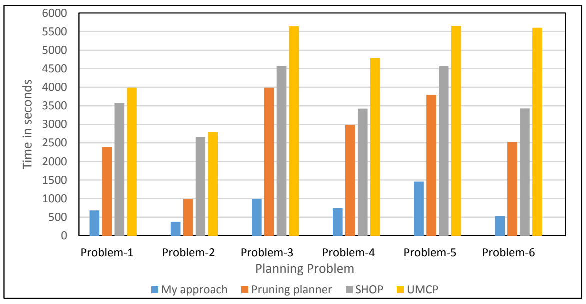
for the abstract tasks in the order in which they are to be executed, and UMCP (Kutluhan et al., 1994), which plans are primarily developed into primitive plans that mean all tasks are primitive and after that the causal interactions between these tasks, are handled. Secondly, the search space size, which means that the number of intermediate plans that are visited to obtain the first solution, is measured.

5.1 Benchmark Problem Set

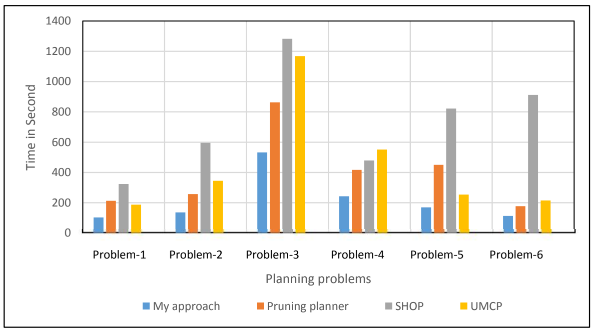
Our experiment depends on four domains model. Two of them from the IPC (McDermott, 2000) and well known in classical planning, and we adapted them to work in the hierarchical environment. The other domains are adapted from an ongoing research project.
Firstly, From IPC, we adopted a non-hierarchical planning domain so-called Satellite domain to work in hierarchical planning. As depicted in Table 2, the adapted Satellite domain model includes three complex tasks and six primitive tasks, as well as eight decomposition methods. The second domain that adapted from IPC is so-called WoodWorking.It consists of thirteen primitive tasks, six complex tasks, and fourteen methods the processing of raw wood into smooth and varnished product parts. UM-Translog is a deep hierarchical planning domain that supports transportation and logistics. In addition, we applied the so-called SmartPhone domain, a new hierarchical planning domain that is concerned with the operation of a smart-



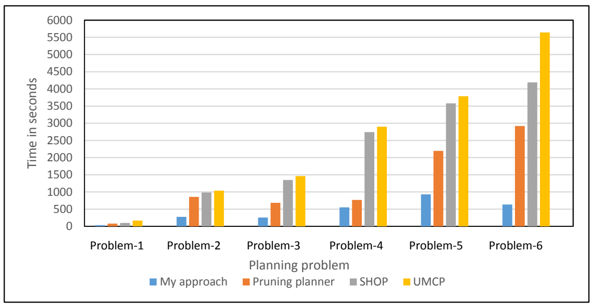
(a) Satellite Domain.



(b) SmartPhone Domain.



(c) UM-Translog Domain.

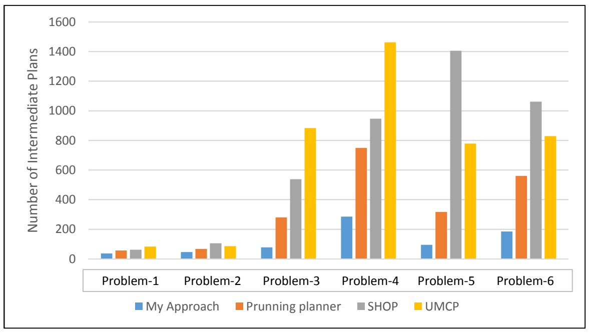


(d) WoodWorking Domain.

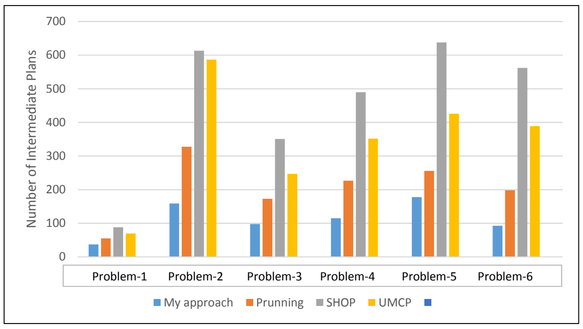
Figure 3: Evaluation results of the Different Domains: Processing Time.

Phone by a human user, e.g., sending messages and creating contacts or appointments. As depicted in Table 2, SmartPhone is a rather large domain with a deep decomposition hierarchy.

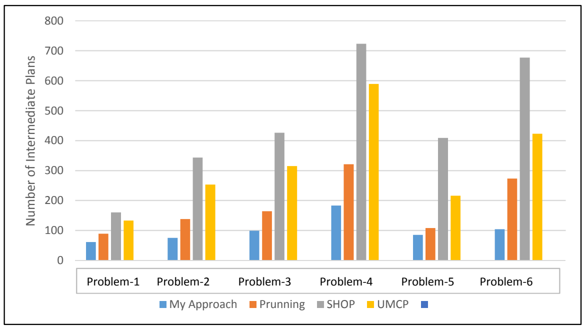
Note that the Satellite planning problems become difficult when modeling some observations, which means that a small number of decomposition methods is used many times in different contexts of a plan. The evaluated scenarios are thus defined as observations on one, two or three satellites. The complexity of planning problems of woodWorking domain come from the variations of parts to be processed. In the smartPhone domain, the difficulty of planning problem appears when managing different daily life tasks. On the other hand, the difficulties of UM-Translog problems come from the decomposition structure, because specific packages are transported in various ways, such as toxic liquids in trains require completely different decomposition methods than transporting regular packages in trucks. Therefore, we conducted our experiments on qualitatively various problems by specifying different locations and/or number of packages.
Figure 3 shows the runtime needed to solve the specified planning problem of our benchmark set using different planners. The time represents the total running time of the planning system in seconds, including the pruning and transformation technique. Note that if the plan generation process did not find a solution plan within the time constraint (maximum time 9,000 seconds) and had therefore been canceled.



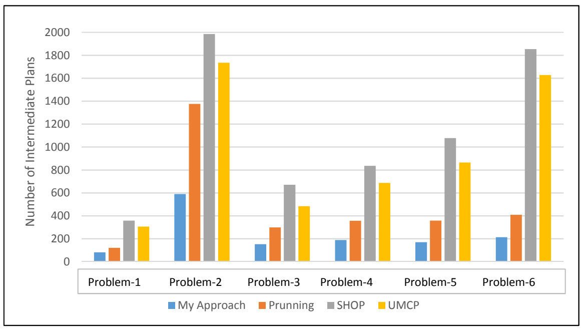
(a) Satellite Domain.



(b) SmartPhone Domain.



(c) UM-Translog Domain.



(d) WoodWorking Domain.

Figure 4: Evaluation results of the Different Domains: Plan space Size.

As depicted in Figure 3, the average performance improvement overall problems in the UM-Translog domain is about 46% in comparison with hybrid pruning planning without transformation. The improvement performance reaches to 71% in case of comparing with the SHOP planner. From our results, we observed that the highest gain is accomplished in the transportation tasks that include special goods such as frozen products and transportation means such as automobiles.
In general, the shallow decomposition hierarchy such as Satellite domain (see Figure 3-a) does not benefit significantly from the pruning technique, it achieves high improvement from applying our pruning and transformation approach.
The SmartPhone and WoodWorking domains (Figures 3-b and 3-d) are two examples of deepest hierarchy domains. Therefore, these domains contain a lot of information that help our approach to achieve high performance. As well as our planner can solve planning problems the other planners cannot find solution plan within the given resource bounds.

In general, our planner improves the planning time performance of hybrid planning system. The average performance improvement of our hybrid planner is about 59% in comparison with pruning hybrid planner while the improvement is about 77% comparing with the hierarchical planner.
On the other direction, as depicted in Figures 4, the number of visited plans to reach a solution

plan is decreased in different evaluation domains. The Satellite domain (See Figure 4-a) gains the biggest improvement. The average improvement of the number of intermediate plans is 68%.

6 CONCLUSION

In this paper, we presented an efficient hybrid planner. It analyzes the hybrid planning domain according to the knowledge in the given hierarchical planning problem and then prunes the irrelevant parts, which cannot reach to a solution. After that, a transformation technique is applied to translate the reduced domain model to an STRIPS-style form. In addition, the decidability of plan existence is approved. Furthermore, we ran experiments on a set of benchmark planning problems that are relevant to our approach and a number of hybrid planning domains. We compared the performance of the proposed approach with the performance of standard planners from the literature. The results show that the average performance that gain from applying our approach is about 59%, especially in the deepest hybrid domains. In case of comparing our approach with the traditional hierarchical planner, the performance improvement reaches 77%. In addition, the evaluation shows that our approach outperform the traditional approaches on all problems with a deep decomposition hierarchy. The presented approach is domain independent and can be applied on any hierarchical planner to improve its performance.

References

Bercher, P., Hller, D., Behnke, G. and Biundo, S. 2015. User-centered planning - a discussion on planning in the presence of human users, *Procedings of the International Symposium on Companion Technology* .

Bercher, P., Keen, S. and Biundo, S. 2014. Hybrid planning heuristics based on task decomposition graphs, *Procedings of the 7th Annual Symposium on Combinatorial Search (SoCS)* pp. 35–43.

Bonet, B. and M., H. 2010. Strengthening landmark heuristics via hitting sets, *Procedings of the ECAI-10* pp. 329–334.

Domshlak, C., Katz, M. and Lefler, S. 2010. When abstractions met landmarks, *Procedings of the ICAPS-2010* pp. 50–56.

El Sehiemy, R., Abou El-Ela, A. and Shaheen, A. 2013. Multi-objective fuzzy-based procedure for enhancing reactive power management, *Journal of IET Generation, Transmission and Distribution* **7**(12): 1453–1460.

Elkawkagy, M., Bercher, P., Schattenberg, B. and Biundo, S. 2012. Improving hierarchical planning performance by the use of landmarks, *Procedings of the 26th National Conference on Artificial Intelligence (AAAI 2012)* pp. 1763–1769.

Elkawkagy, M., Schattenberg, B. and Biundo, S. 2010. Landmarks in hierarchical planning, *Procedings of the ECAI-2010* .

Fikes, R. and Nilsson, N. 1971. strips: a new approach to the application of theorem proving to problem solving, *Artificial Intelligence* **2**: 189–208.

Fox, M. and Long, D. 1998. The automatic inference of state invariants in tim, *JAIR* **9**: 367–421.

Geier, T. and Bercher, P. 2011. On the decidability of htn planning with task insertion, *Procedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI - 2011)* pp. 1955–1961.

Gregory, P., Cresswell, S., Long, D. and Porteous, J. 2001. On the extraction, ordering, and usage of landmarks in planning, *Procedings of ECP* pp. 37–48.

Gregory, P., Cresswell, S., Long, D. and Porteous, J. 2004. On the extraction of disjunctive landmarks from planning problems via symmetry reduction, *Procedings of the SymCon-2004* pp. 34–41.

Haslum, P., Bonet, B. and Geffner, H. 2005. New admissible heuristics for domain-independent planning, *Procedings of AAAI-05* pp. 1163–1168.

Helmert, M. and Domshlak, C. 2009. Landmarks, critical paths and abstractions: Whats the difference anyway?, *Proceedings of ICAPS-09* pp. 162–169.

Hoffmann, J. and Nebel, B. 2001. The ff planning system: Fast plan generation through heuristic search, *Journal of Artificial Intelligence Research* **14**(1): 253–302.

Hoffmann, J., Porteous, J. and Sebastia, L. 2004. Ordered landmarks in planning, *Journal of the Artificial Intelligence Research* **22**: 215–278.

Kambhampati, S., Dattatraya, M. and Srivastava, B. 1998. Hybrid planning for partially hierarchical domains., pp. 882–888.

Karpas, E. and Domshlak, C. 2009. Cost-optimal planning with landmarks, *Procedings of the IJCAI-2009* pp. 1728–1733.

Keyder, E., Richter, S. and Helmert, M. 2010. Sound and complete landmarks for and/or graphs, *Procedings of the ECAI-2010* pp. 335–340.

Kutluhan, E., Hendler, J. and Nau, D. 1994. umcp: A sound and complete procedure for hierarchical task network planning, *Procedings of AIPS, ICAPS 2009 influential paper honorable mention* **61**: 249–254.

McDermott, D. 2000. The 1998 ai planning systems competition, *Journal of the AI Magazine* **21(2)**: 35–55.

Nau, D., Ghallab, M. and Traverso, P. 2004. automated planning: Theory and practice, *Morgan Kaufmann Publishers Inc., San Francisco, CA, USA* .

Nau, D., Tsz-Chiu, A., Ilghami, O., Kuter, U., Murdock, W. and Yaman, F. 2003. Shop2: An htn planning system, *Journal of Artificial Intelligence Research* **20**: 379–404.

Pednault, E. 1989. Adl: Exploring the middle ground between strips and the situation calculus, *Procedings of the KR-89* pp. 324–332.

Precup, R., David, R., Petriu, E., S., P. and Radac, M. 2013. Fuzzy logic-based adaptive gravitational search algorithm for optimal tuning of fuzzy-controlled servo systems, *Journal of IET Control Theory and Applicat* **7**(1): 99–107.

Ramírez-Ortegón, M., Märgner, V., Cuevas, E. and Rojas, R. 2013. An optimization for binarization methods by removing binary artifacts, *Journal of Pattern Recognition Letters* **34**(11): 1299–1306.

Richter, S., Helmert, M. and Westphal, M. 2008. Landmarks revisited, *Procedings of the AAAI-2008* pp. 975–982.

Sebastia, L., Onaindia, E. and Marzal, E. 2006. Decomposition of planning problems, *Journal of the AI Communications* **19**(1): 49–81.

Seegebarth, B., Muller, F., Schattenberg, B. and Biundo, S. 2012. Making hybrid plans more clear to human users a formal approach for generating sound explanations, *Procedings of the 22nd International Conference on Automated Planning and Scheduling (ICAPS-2012)* pp. 225–233.

Shivashankar, V., Alford, R., Kuter, U. and Nau, D. 2013. The godel planning system: a more perfect union of domain-independent and hierarchical planning, *Procedings of the Twenty-Third international joint conference on Artificial Intelligence* pp. 2380–2386.

Tomin, N., Zhukov, A., Sidorov, D., Kurbatsky, V., Panasetsky, D. and Spiryaev, V. 2015. Random forest based model for preventing large-scale emergencies in power systems, *International Journal of Artificial Intelligence* **13**(1): 211–228.