International Journal of Innovative Technology and Exploring Engineering (IJITEE)
ISSN: 2278-3075 (Online), Volume-10 Issue-7, May 2021

Reduce Artificial Intelligence Planning Effort by using Map-Reduce Paradigm

Mohamed Elkawkagy, Heba Elbeh

Abstract: ***While several approaches have been developed to enhance the efficiency of hierarchical Artificial Intelligence planning (AI-planning), complex problems in AI-planning are challenging to overcome. To find a solution plan, the hierarchical planner produces a huge search space that may be infinite. A planner whose small search space is likely to be more efficient than a planner produces a large search space. In this paper, we will present a new approach to integrating hierarchical AI-planning with the map-reduce paradigm. In the mapping part, we will apply the proposed clustering technique to divide the hierarchical planning problem into smaller problems, so-called sub-problems. A pre-processing technique is conducted for each sub-problem to reduce a declarative hierarchical planning domain model and then find an individual solution for each so-called sub-problem sub-plan. In the reduction part, the conflict between sub-plans is resolved to provide a general solution plan to the given hierarchical AI-planning problem. Pre-processing phase helps the planner cut off the hierarchical planning search space for each sub-problem by removing the compulsory literal elements that help the hierarchical planner seek a solution. The proposed approach has been fully implemented successfully, and some experimental results findings will be provided as proof of our approach's substantial improvement inefficiency.***

Keywords**:** ***Artificial Intelligence Planning, Map-Reduce, Hadoop, Big Data.***

I. INTRODUCTION

The paper concerns novel artificial intelligence (AI) hierarchical planning technique that combines the hierarchical AI-planning with map-reduce paradigm to increase the planning performance. Today, the Artificial Intelligence (AI) planning field offers a range of techniques for constructing a solution for a planning problem (P) (Definition-1). The process of generating a sequence of actions that achieve the desired goal (solution) is defined as the Planning process (Definition-2 and 3) [1]. The classical planning paradigm is standard AI planning.

Definition 1. An initial state (S), actions (A), and goal state (G) are three parameters constitutes the classical planning problem (P_{Cls}) P_{Cls}= <S, A, G> such that the state is defined as a collection of logical facts, and A set of literals expresses State S as initial and State G as a goal.

Definition 2. Each action A consists of two components <Pre(a), Eff(a)> . The Pre(a) parameter reflects the pre-conditions that have to be fulfilled before performing the operation. Eff(a) is a collection of facts that alter the world state's status after the operation's execution. In classical planning, actions are depicted by Stanford Research Institute Problem Solver (STRIPS) [2].

Definition 3. (Transition state $\left(T(S,a,S')\right)$): The new state S' is created by performing the action in the current S as the following:

$$T(S,a,S')=(S \cup Eff(a)) \text{ iff } Pre(a) \subset S.$$

So that, in classical planning, the plan is defined as a series of actions $a \subset A^*$ which gradually changes the initial state S to goal state G.

Hierarchical Task Network (HTN) planning [3], [20] is a commonly used planning process model. Tasks and Methods are the basis of an HTN preparation. Complex tasks are complex activities such as the transportation of certain goods from a particular place to a different place. Primary tasks are classical planning behavior. The hierarchical domain model provides a variety of decomposition methods to solve the related complex task. Each method offers a set of tasks ordered partially so-called task network, specifying the predefined solution of the complex task. The initial task network represents the hierarchical planning problem. They are solved by progressively breaking down the complex tasks to consistent rudimentary tasks is in the network. The breakdown of complex tasks by a suitable method substitutes the complex task with the decomposition process's scheme.

Several approaches used Multi-agent planning (MAP) to solve big planning problems. The central concept of these approaches is separating the issue of planning problem into sub-problems. Each of them is resolved to create a solution for each sub-problem. These solutions are merged to produce a general solution plan [4]. Most work in the MAP focuses on how to coordinate between the agent's planning. In general, there are three approaches used to resolve the coordination between agents planning. The first category discusses the collaboration between completed plans [5]. The authors proposed a less expensive merging plan in this category by leveraging positive interactions and overcoming inconsistency among sub-problems solutions.

Manuscript received on May 03, 2021.
Revised Manuscript received on May 06, 2021.
Manuscript published on May 30, 2021.
* Correspondence Author
Mohamed Elkawkagy*, Computer Science Department, Faculty of Computers and Information, Menofiya University, Shebin El Kom, 32511, Egypt. Email: m_nabil_shams@yahoo.com
Heba Elbeh, Computer Science Department, Faculty of Computers and Information, Menofiya University, Shebin El Kom, 32511, Egypt. Email: heba_elbeh@yahoo.com

The second category discusses the interleaving between coordination and planning processes, which means conflicts between sub-plans are overcome during constructing sub-plans [6]. The third category discusses two different kinds of coordination: implicit coordination, which manages agent behavior through issuing some rules, and explicit coordination, which starts by issuing new constraints to the planning problem to guarantee the feasibility of the issued solution [7]. In order to increase planning efficiency, many scientists employed a hierarchical structure in the MAP approach. Nets Of Action Hierarchies (NOAH) [8] framework discussed the idea of interconnect the hierarchical planning and merging through sharing resources. It was built through the emphasis on efficient communication between agents. Furthermore, another approach worked in a dynamic environment by arranging the agents' set into layers as a patent agent and a set of cooperative agents (children) working together towards the target [9]. Afterward, the authors in [10] introduced a technique to detect agent plans' conflicts at the primitive and abstract levels. As opposed to these approaches, the hierarchical planning approach is incorporated only once time with the Map-Reduce paradigm to reduce the planning effort. The author in [11] presented a cooperative technique to construct solution plans. Landmark heuristics guide the coordination process among agents while building the solution. This paper introduces a new hierarchical planning approach that combines hierarchical planning with the map-reduce paradigm to improve hierarchical planning performance. The given problem is divided into sub-problems according to the proposed clustering algorithm in the mapper phase. Afterward, identical mappers are initiated according to the number of sub-problems and then apply the pre-processing technique for each sub-problem to reduce the domain model by extracting the mandatory tasks to be a member solution plan. After that, the mappers start to find a solution for the given sub-problem. Each mapper uses HTN-style planning to construct its solution. The merging process is done in the reduction phase. The merging process is handled through two-step detection steps that discover the conflicts between individual solutions and solving steps that overcome the detected conflicts. MapReduce framework [12] is a parallel and distributed programming paradigm. It is designed to facilitate a parallel program that can perform a data-intensive computation efficiently. It also provides a great offer to convert parallel programs to cloud computing environments and server clusters. Section 2 explains the planning structure. After that, section 3 introduces the proposed architecture. The experimental scenario and the outcomes of the assessments are illustrated in Section 4. In section 5, the paper ends with a conclusion.

II. FORMAL FRAMEWORK

Our proposal uses hybrid planning structure which incorporates Partial-Order-Causal-Link (POCL) and HTN [13], [14], [21], [22]. The planning of POCL is a tool used for solving conventional planning problems. The POCL plan includes a sequence and partially organized actions and demonstrates the action's dependencies through causal ties. $t(\tau)$= <Pre$(t(\tau))$, Eff$(t(\tau))$> represents The structure of the task, which identified by pre-conditions and effects of a task, where effects consist of two forks: positive and negative literals over the parameters $\tau=\tau_1, \tau_2\ldots\ldots, \tau_n$ The hybrid planning framework represents pre-conditions and effects for primitive and complex tasks to encode POCL operations. To encode multiple instances for the same task in the same partial plan, we use identifier plan steps PS(τ), which has the form PS(τ)=<id,t(τ)>, where an id is a natural number. A partial hybrid plan comprises four components: plan steps PS which specify a task either primitive or complex, notation < specifies order restriction between PS, variable constraints are represented by notation V, and C represents causal links. The variable constraints (V) are composed of (in) equations associating variables with constants or other variables. It represents the (partial) substitution in Ps of the plan steps. Gnd(PS, V) means several ground tasks accomplished in a way consistent with V, which equates all parameters in all the P with constants' tasks. Causal links are drawn from the preparation of the POCL: the causal relation $PS_i \xrightarrow{\vartheta} PS_j$ means that the pre-condition ϑ of the plan Step PS_i is inferred, and it is considered a result of the plan step PS_j. Methods M=<t(τ'), P> are the implementation of the abstract task t. Generally, each complex task is resolved or implemented using multiple methods. A Hybrid planning problem is formalized from four issues HP_{prob}=<D, S_{init},P_{init}, G>. Notation D. represents a domain model. It consists of a task schema T and implementation methods M. S_{init}, P_{init}, $\mathcal{G}$ represents the initial world state, initial plan and goal state respectively. Notice that this paper focuses on pure HTN planning, so that the goal state G is omitted from the planning problem. A partial plan P=<PS, $\prec$, V, C> is a solution of a problem iff: (1) P is a refinement of P_{init}, i.e., a successor to the initial plan in the search space (Definition. 4), (2) a causal link in C supports each pre-condition of a plan step in P, and no such connection is threatened. For example, In causal connections $PS_i \xrightarrow{\vartheta} PS_j$ the ordering constraints restrict that no plan step PS_k with an effect that implies$\neg\vartheta$ can be imposed between plan steps PS_i and PS_j, (3) The constraints in order and variable are consistent, i.e., do not result in S cycles and the (in-) equations in V are consistent, and (4) The Plan Steps in S are all primitive tasks. There are various refining measures (or plans modifications) in the form of the HTN plan: (1) the decomposition of complex task employing methods; (2) use causal links to close open-plan steps

pre-conditions; (3) adding ordering constraints; and (4) the inclusion of variable constrain.

Definition 4 (search space). The directed graph S_{space}=<V_{space}, E_{space}> is called the planning problem search space iff (1) $P_{init} \in V_{space}$ (2) if there is a refinement for plan $P \in V_{space}$ to a plan P', then $P' \in V_{space}$ and $(P,P') \in E_{space}$, and (3) S_{space} is minimal holding such that includes items (1) and (2). In S_{space}, instated writing $P \in V_{space}$., we will write $P \in S_{space}$. Generally, S_{space} has no acyclic or finite.

Our hierarchical planning algorithm (Alg. 1) guides the search space heuristically to find solutions. The fringe F= <P_1,……., P> is a series of unexplored plans which directly successors of visited non-solution plans in the Planning problem HP_{prob}, which initially started with only one plan, P_{init}. The Fringe <P_1,……., P_n> is structured such that a P_i the plan leads quicker to a solution than a plan P_j for j□i. After that, use module PlanSel to delete the selected plan from the fringe set after selecting it.

The planning algorithm starts by considering the Fringe as input; if there are plans in the Fringe, they recursively select one plan from the Fringe through plan selection strategy PlanSel(F) (lines 1-2). Note that the selected plan is removed from the fringe F (line 3). The algorithm will terminate with the solution if the selected plan does not have any flaw (lines 4-5). Otherwise, the flaw selection module (FlawSel) will detect all flaws in the specified plan (line 6). Any components that violate the solution conditions are considered a flaw, such as a complex task flaw and a Threat flaw. A plan selection (PlanSel) revised the Fringe from the plans with unresolvable flaws such as inconsistent ordering constraints and then ordered the rest plans in the Fringe. Hence apply the flaw refinement in the selected plan and add a new plan in the Fringe (line 7). The planner algorithm is ended by returning the solution or failure (line 8).

Algorithm 1: Hierarchical Planning Algorithm

Input : Hierarchical planning problem $\mathbb{HP}_{prob}$

Fringe $\mathbb{F} \leftarrow < \mathcal{P}_{init} >$

Output : Solution Plan $\mathcal{P} =< PS, \prec, V, C >$ or *unsolvable problem*

```
1  while (F ≠ ∅) do
2  |  P ← PlanSel(F)
3  |  F ← (F) \ P
4  |  If (Flaws (P) = ∅) then
5  |  Return P
6  |  F ← FlawSel(Flows (P))
7  |  F ← F ∪ {apply (m(F), P)}
8  return unsolvable problem
```

III. THE PROPOSED ARCHITECTURE

A lot of planning and Multi-agent planning (MAP) approaches have been presented to address broad planning problems. This paper proposes a new approach to integrate Map Reduce Architecture regarding hierarchical planning to enhance planning systems. Figure 1 shows modules of the proposed architecture. The pre-processing module is used to remove irrelevant information from the domain model when considering the hybrid planning sub-problem. The clustering algorithm is used to divide the defined planning problem into various individual sub-problems and a set of constraints, so-called shared constraints pool (SCP). Shared Constraint Pool is a generic memory that contains a set of constraints. The Mapper module encapsulates several steps, such as pre-processing for the given sub-problem, finding the sub-plan for the specified problem. All mappers clone all literals in the initial state of the hierarchical planning problem. Reduction module will receive SCP and an individual solution (sub-plan) of each sub-problem to overcome the conflicts between these solutions to find a general solution to the concerned hierarchical problem. Before introducing the hierarchical planning problem to the mappers, the proposed clustering technique is used to divide the hierarchal problem into a series of sub-problems. The clustering technique is based on the principle of creating a series of independent clusters.

In the split process, the planning problem is broken down into clusters according to two clustering algorithms: Dependent and Independent. Dependent algorithm establishes several dependent clusters, while the independent algorithm generates a series of separate clusters. The behavior of the two techniques is shown in the experimental section. The independent tasks in the current plan are isolated within a single cluster. The task constraints in various clusters are simultaneously inserted into shared constraints pool SCP, As shown in Alg. 2. The input of the Dependent algorithm is the planning problem, and the output is the Γ set. The set Γ includes the shared constraints pool SCP and different clusters $\Pi_\gamma = \bigcup_{i=0}^{n} \{\Pi_{\gamma_i}\}$. To extract all clusters, the Dependent algorithm executes recursively to traverse different tasks in the initial plan Pinit. Each cluster Π_γ=<D, S_{init}, $P_{\gamma_{init}}$> consists of D, S_{init}, $P_{\gamma_{init}}$ Which represents the domain model, initial state, and initial partial plan, respectively.

Now, the question is how the Dependent algorithm constructs the set Γ. As shown in Alg. 2, the algorithm starts by initializing the set Γ by a null value (line 1). After that, from the current plan, a new cluster Π_{γ_i} is created by collecting the free pre-request tasks te_i and added to the task network of the initial plan $P_{\gamma_{init}}$ in Π_{γ_i} (line 3-5). Hence, the task network TE of the plan P_{init} in the planning problem, Π is updated by removing the tasks te_i (line 6). For the current tasks PS_i at hand, the created cluster Π_{γ_i} is extend recursively by adding different constraints and updating SCP by appending all constraints that indexed in the relationship between the current task t and other tasks PS_i (lines 7 to 15). In the end, the Γ is modified by adding clusterΠ_{γ_i}, and also shared constraints pool SCP is modified (line 16). The SCP will play a significant role in the reduction phase. According to the updated plan and the set Γ, the Dependent algorithm works recursively to construct a new cluster (line 17).

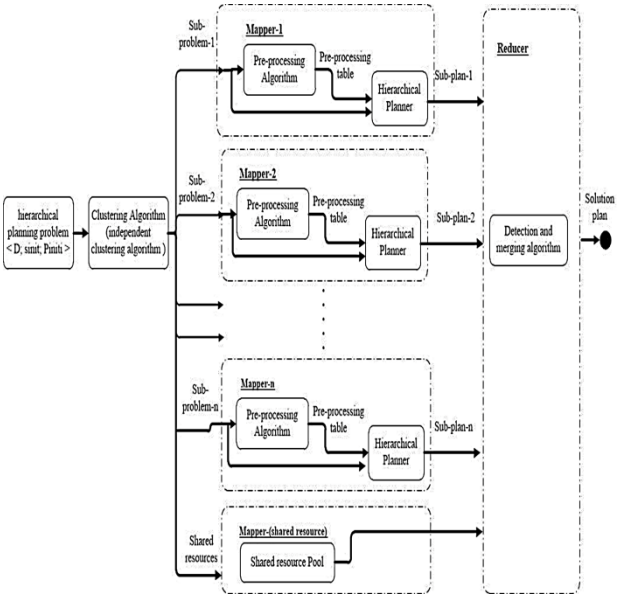

Figure 1. The proposed planning-Map-Reduce Architecture

On the other side, the Independent algorithm collects the dependent tasks in one cluster; this means the SCP is empty. So, to divide the planning problem into several clusters using the Independent algorithm, line 3 in Alg. 2 is updated by "Select the dependent tasks PS_i. This means the resulted clusters are explicitly independent, and the SCP is empty. After that, the Dependent or Independent algorithm has generated clusters distributed among the mappers to build a solution plan (sub-plans) for each cluster.

A. Planning Mapper

Once the planning mappers received sub-problems, each mapper computes the landmark table for the specified sub-problems. After that, each mapper applied the planning algorithm (Alg. 1) to construct individual solutions (Definition-5) for the specified sub-problem.

The inputs of the planner algorithm are the initial plan P_{init} of the allocated cluster and landmark table. Hence, refines the specified cluster (sub-problem) recursively until the individual solution plan is found. To this end, each mapper starts to apply the pre-processing technique to produce the landmark table for the specified sub-problem. The pre-processing technique will help the planner reduce the search space to a smaller one before starting the search process to minimize the planning effort.

Recent work implemented a landmark technique that reduces the hierarchical planning domain and problem definition of the hierarchical planning to a smaller domain [15-16]. As a result, the pre-processing phase based on hierarchical landmarks produces– primitive tasks that appear in the task sequences (plan), which lead the initial plan to its solution (Definition 6-7).

Definition 5 (Solution plan). Let $<V_{HP}, E_{HP}>$ is the hierarchical planning search space of problem HP. Then, the $\cup(P)=\{<P_1.....P_n> | P_1= P(P_1,P_{i+1}) \in E_{HP}$, $\forall$ one $\leq i<n$, and $P_n \in Sol_{HP}, n\geq 1\}$.

Definition 6 (Mandatory Landmark). The landmark of the hierarchical planning problem HP_{prob} is a ground task $t(\tau')$, iff $\forall <P_1.....P_n> \in solution(P_{init})$ there is a $1 \leq i < n$, s. t. $t(\tau') \in Gnd(PS,V_n)$ for $P_i=<PS_i, \prec_i, V_i,C_i>$and $P_n=<PS_n,\prec_n, V_n,C_n>$..

Definition 7 (Complex Landmark). For a complex ground task $t(\tau')$, let $P_{HP_{prob}}(t(\tau')) := \{P \in P_{HP_{prob}} | P=<PS,\prec, V, C>$and $t(\tau') \in Gnd(PS,V)\}$. A ground task $t'(\tau'')$ identifies complex landmark of $t(\tau')$, iff $P \in P_{HP_{prob}}(t(\tau'))$ and each $<P_1.....P_n> \in$ solution $(P_{HP_{prob}})$ s.t. $1\leq i<n$ and $t'(\tau'') \in Gnd(PS_i, V_n)$ for $P_i=< PS_i, \prec_i, V_i, C_i>$and $P_n=<PS_n,\prec_n, V_n, C_n>$.

Definition 8 (Landmark Table). Assume $<V_T,V_M, E>$ is a TDG of the problem $\mathbb{HP}_{prob}$. The landmark table of HP_{prob} $LT=\{<t(t'), M(t(t'))O(t(t'))> | t(t') \in VT\}$, s.t. $M(t(t'))$ and $O(t(t'))$, are defined as the following: $M(t(\tau'))=\{t'(t'') \in VT | t'(\tau'') \in Gnd(PS, V)\}$ $\forall <t(\tau'),\prec,V,C> \in VMO(t(t'))$ $=\{Gnd(PS, V) \setminus M(t(t')) | <t(t'), \prec, V, C> \in VM\}$

The landmark literals will be extracted from the domain model planning problem at hand through the pre-processing phase. The result of the pre-processing procedure is organized in a table so-called a landmark table. Extracting landmark depends on building graph so-called a Task Decomposition Graph (TDG), representing different ways of exploring planning problems by AND/OR graph [17-18]. A TDG is finite consists of several ground tasks, either primitive or complex, and several methods. The landmark algorithm's output is a data structure used to introduce the mandatory landmark tasks and details complex landmark tasks.

The landmark table (Definition 8) includes two distinct types of tasks. Mandatory landmarks $M(t(\tau'))$ are the necessary ground tasks that exist in all solution plans presented by the abstract implementation method that is related to task $t(\tau')$.

On the other hand, the tasks are categorized as complex landmarks of $t(\tau')$ which defined as an optional task $O(t(\tau'))$. This means these tasks are not necessary for the final solution. The pre-processing phase is terminated after building the landmark table and submitting this table to the planner. The planner cans benefit from the landmarks information in two ways. (1) By reducing the domain model through ignoring infeasible methods [15]. (2) By using the landmark literals in the planning strategy [18]. Each mapper will use landmark information to construct its solution. The proposed approach relies on a hierarchical planning model; extracting landmarks about the hierarchical declarative domain can be extracted while generating the task expansion modifications.

First, the module F^{ModGen}ComTask of modification generation is deployed regarding the landmark table LT_i of the specified problem Π_γ In the traditional hierarchical planner, in each iteration, the module F^{ModGen}ComTask considers all methods of the specified complex task flaw step $t(\tau')$. Instead of that, in the proposed work, the optional set $O(t(\tau'))$ in the LT is used, Which means that the algorithm will rely on the produced reduced pre-processing phase. It is essential to show that the planning process generally works with the domain model reduction with flaw and modification modules or search control strategies. If the planning algorithm is carried out failure, the current mapper can divide the current sub-problem into more pieces and perform all the above functions. Finally, each mapper generates an individual solution plan p and sends it to the reducer.

Algorithm 2 : Clustering Algorithm (Dependent)

Input : Hierarcichal planning problem $\mathbb{HP}_{prob} =< \mathbb{D}, S_{init}, \mathcal{P}_{init} >$

Output : $\Gamma = (\Pi_\gamma, SCP)$

1. Initialize $\Gamma \leftarrow$ null
2. If $(\mathcal{P}_{init}(TE) == \emptyset)$ return Γ
3. $PS_i \leftarrow (\mathcal{P}_{init}(TE))$ *where* PS_i *are prerequest free*
4. $\Pi_{\gamma_i} \leftarrow < \mathbb{D}, \mathcal{S}_{init}, \mathcal{P}_{\gamma init} >$
5. $\Pi_{\gamma_i}(\mathcal{P}_{\gamma init}) \leftarrow PS_i$
6. $(\mathcal{P}_{init}(TE)) \leftarrow (\mathcal{P}_{init}(TE)) \setminus PS_i$
7. For each ($task\ t \in PS_i$) do
8. Add all constraints $<\prec_t, V_t, C_t >$ that point to tasks t with other tasks in PS_i to $(\mathcal{P}_{\gamma init})$
9. $(\mathcal{P}_{init}(V)) \leftarrow (\mathcal{P}_{init}(V)) \setminus V_t$
10. $(\mathcal{P}_{init}(\prec)) \leftarrow (\mathcal{P}_{init}(\prec)) \setminus \prec_t$
11. $(\mathcal{P}_{init}(C)) \leftarrow (\mathcal{P}_{init}(C)) \setminus C_t$
12. $SCP \leftarrow SCP \cup$ s.t. these constraints point to task t with anoter tasks $t' \in PS_i$
13. $(\mathcal{P}_{init}(V)) \leftarrow (\mathcal{P}_{init}(V)) \setminus V_i$
14. $(\mathcal{P}_{init}(\prec)) \leftarrow (\mathcal{P}_{init}(\prec)) \setminus \prec_i$
15. $(\mathcal{P}_{init}(C)) \leftarrow (\mathcal{P}_{init}(C)) \setminus C_i$
16. $\Gamma \leftarrow (\Pi_\gamma \cup \Pi_{\gamma i}, SCP)$
17. Return $\Gamma = (\Pi_\gamma, SCP)$

B. Reducer

After all planning mappers have been generated individual solution plans for all sub-problems, the reducer will start running the detection and merging process to generate a general solution plan. Employing the detection and merging process, the reducer identifies conflicts between different solutions to resolve these conflicts through a merging process.

The proposed merging technique relies on Fragment terminology. The merging technique consists of two stages; (1) The output of the mappers, which are individual plans $P\Gamma=\{P_{\gamma 1}, P_{\gamma 2}, \ldots P_{\gamma n}\}$ is split into groups of so-called Fragments. The constraints especially ordering constraints in the SCP, are used in building the group of Fragments.

Each fragment $F= <P_\gamma, O_\gamma>$ consists of a set of individual plans P_γ ($P_\gamma \subset P\Gamma$) and ordering constraints O, which add partial order on individual plans $\mathcal{P}$. For example, assume that the set of ordering in SCP is $\{PS_1 \prec PS_2, PS_2 \prec PS_3, PS_4 \prec PS_5\}$

Let a set of individual plans $P\Gamma=\{P_{\gamma 1}, P_{\gamma 2}, \ldots P_{\gamma 7}\}$ is respectively, a solution for complex plan steps $PS_1, PS_2, \ldots PS_7$. Then these complex plans in $P\Gamma$ construct three different fragments $F_1=<\{P_{\gamma 1}, P_{\gamma 2}, P_{\gamma 3}\}, \{P_{\gamma 1} \prec P_{\gamma 2}, P_{\gamma 2} \prec P_{\gamma 3}\}>$, $F_2=<\{P_{\gamma 4}, P_{\gamma 5}\}, \{P_{\gamma 4} \prec P_{\gamma 5}\}>$ and $F_0=<\{P_{\gamma 6}, P_{\gamma 7}\}, \{\varphi\}>$. Please consider that in the merging technique, we have two different kinds of fragments. dependent-Fragment which have individual dependent plans like F_1 and F_2 And independent-Fragment, which have individual plans which do not have an explicit dependency like F_0. (2) Each fragment's plans are combined to construct MFP, which is a Merged-Fragment plan. Hence all MFPs are put together to construct a solution plan. So, the implicit dependency among plans is determined, especially in independent-Fragment. This was done through matching the pre-conditions and effects of tasks in different plans within the same fragment, which means the specific effect of plan pγi tasks is found as pre-conditions for plan $P_{\gamma i}$ tasks.

Definitely, determining plan dependency is necessary to remove the negative interactions, which means the task can remove another task's effect or pre-condition. It also helps the merging technique benefit from the positive interactions that mean specific pre-condition is needed by two different tasks and ensure that one of them does not remove it or constructs the needed pre-condition of the other task. On the other side, the independent plans will be performed simultaneously through their integration into a single plan. If it detects dependency between plans in an independent-Fragment, the reducer updates it by adding ordering constraints. Regarding the dependency between plan steps, some tasks in these plans will be executed at the same time. Note that tasks cannot operate at the same time if there is an inconsistency between the pre-conditions or effect of successor scheme tasks and effects of predecessor schemes effects. The comparison between pre-conditions and effects in one side plan (successor) and the effects of tasks in the second side plan (predecessor) determines the concurrent tasks. The comparison process is started by checking pre-conditions and effects of the first task in the successor plan $P_{\gamma i}$ with post-conditions of different tasks in the predecessor plan $P_{\gamma i}$.

Case-1 (Sequential Case) is executed if the pre-conditions or effects of the first task in $P_{\gamma i}$ are violated so that this task will execute sequentially with plan $P_{\gamma i}$.

Otherwise, Case-2 (Concurrent Case) is executed; which means the first task will be executed concurrently with the plan $\mathcal{P}_{\gamma i}$. The comparison process in case-2 is run recursively on the successor plan $P_{\gamma j}$ with the next task. At this point, we neither need case-2 nor steps numbers 4 and 5 in case-1.

Case-1(Sequential Case): consists of six steps; (1) construct ordering constraint <last task $\in P_{\gamma i}$The current task in $P_{\gamma j}$> , (2) Delete the ordering constraint <last task $\in P_{\gamma i}$, goal task() $\in P_{\gamma i}$>, (3) Delete goal task () $\in P_{\gamma i}$, (4) Delete ordering constraint <initial task() $\in P_{\gamma j}$, current task in $P_{\gamma j}$> , (5) Delete initial task() $\in P_{\gamma j}$, (6) Terminate the comparison process.

Case-2 (Concurrent case): (1) Delete ordering constraint <initial task () $\in P_{\gamma j}$, current task $\in P_{\gamma j}$> , (2) Delete the initial task () $\in P_{\gamma j}$, (3) construct the ordering constraint <initial task ()$\in P_{\gamma i}$, current task $\in P_{\gamma j}$>, (4) go further in comparison with the new task in the successor plan $P_{\gamma j}$.

Note that two tasks in diverse plans are merged if their effects are the same, and at the same time, no task can violate these effects (Definition 9).

Definition 9 (Merging tasks merge (PS_i, PS_j)). ∀ plan steps PS_I and $PS_j \in P_{\gamma i}$, merge (PS_I, PS_j) iff ($(Eff(PS_i)=Eff(PS_j)) \wedge (\nexists\ PS_n \in P_{\gamma i}$ violate $eff(PS_j)$ s.t.$(PS_j \prec PS_n)))$

After merging tasks, the related constraints should be combined and assigned as new constraints for the merged task.

For all merged tasks $PS_I, PS_j \in P_{\gamma i}$: (1) The plan step PS_j is replaced by plan step PS_I , (2) ∀ $PS_n, PS_m \in P_{\gamma i}$ and ∃$<PS_n, PS_j>$, $<PS_j, PS_m> \in \prec$of plan $P_{\gamma i}$ add new order constraint $<PS_n, PS_m>$ to plan $P_{\gamma i}$, (3) Delete $<PS_n, PS_j>$, $<PS_j, PS_m>$ from plan $P_{\gamma i}$. These rules guarantee that the ordering constraints of the merged tasks are conserved.

On the causality constraints that include the merged task in its components should be modified. For all merged plan steps $PS_i, PS_j \in P_{\gamma i}$: ∀ $PS_n, PS_m \in P_{\gamma j}$:(1) ∀ causal link ∀ $PS_j \xrightarrow{\Phi} PS_m \in$CL of plan $P_{\gamma j}$ add a new causal link $PS_i \rightarrow PS_m$ to CL of plan $P_{\gamma j}$, (2) Delete causal link $PS_j \xrightarrow{\Phi} PS_m$ from CL of plan $P_{\gamma j}$, (3) Delete causal link $PS_n \xrightarrow{\Phi} PS_j$ from the CL of plan $P_{\gamma j}$.

IV. EXPERIMENTAL RESULTS

We have performed a variety of experiments to measure the realistic efficiency obtained by the proposed approach. The experiments were performed on a machine with 3 GHz CPU and 4GB RAM. The Hadoop-2.6.0.tar on ubuntu-14.04.1 for both Hadoop platforms Single node and multi-node cluster with a master and two slaves is installed. Besides, we installed the Hadoop requirement Java OpenJDK by using the apt-get command. We have checked Hadoop running components using the jps command. The output of the command in both the single platform and the multi-node cluster is shown in Figure4. Note, we can access a single node's interface from http://localhost:50070/ and in a multi-node cluster from http://hadoopmaster:50075/.

The experiment includes two different hierarchical domains. The first one is a shallow hierarchical domain, which is the satellite domain. It represents the stellar scientific observations on Earth-orbiting instrument platform problems. The domain satellite consists of 3 complex and five primitive tasks, where eight methods connect complex tasks. The problems derived from the satellite domain are typically simple, but it will be complicated when modelling observations more on time. This means that domain methods are repeated several times in various situations in the solution plan.

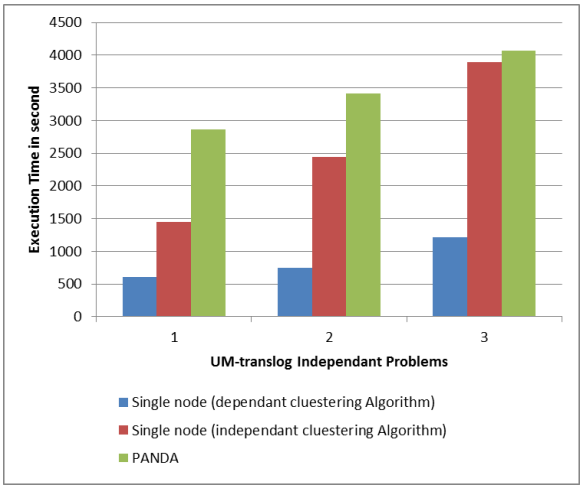

Figure 2. Independent UM-Translog problems (Map-Reduce single node with dependent and independent clustering techniques)

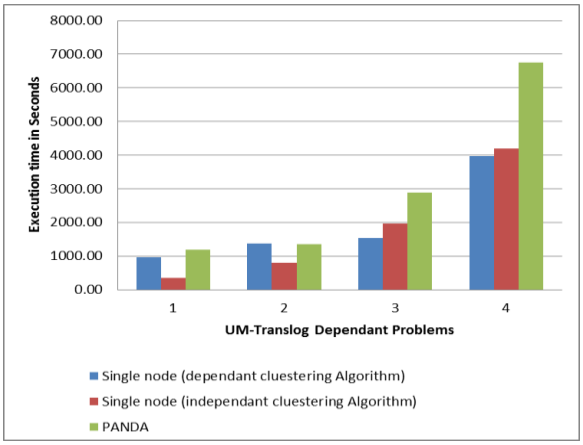

Figure 3. Dependent UM-Translog problems (Map-Reduce single node with dependent and independent clustering techniques)

The second one is the UMTranslog, which is deep in a hierarchical domain. It is motivated by the problem of transporting and logistics goods. UmTranslog domain includes 21 complex tasks connected with 51 methods and 48 primitive tasks. These two domain models discuss the problem features they trigger.

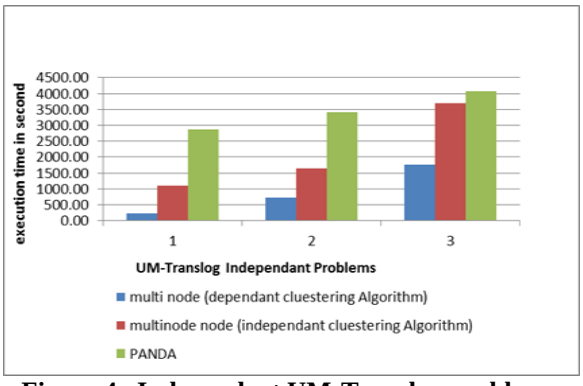

Figure 4. Independent UM-Translog problems (Map-Reduce multi-node with dependent and independent clustering techniques)

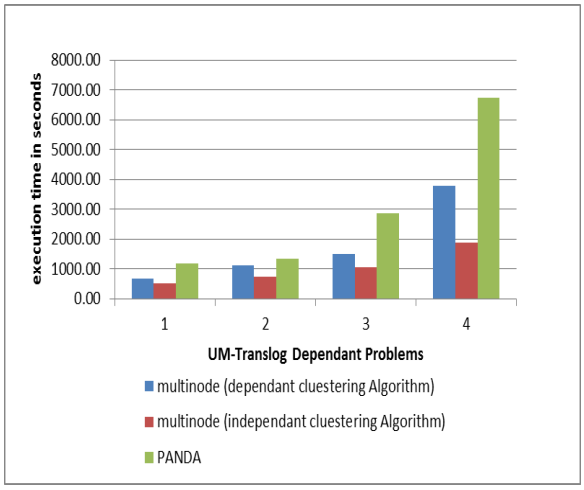

Figure 5. Dependent UM-Translog problems (Map-Reduce Multi-node with dependent and independent clustering techniques)

UM-Translog domain issues problems usually vary in terms of the decomposition structure since various ways handle particular transport products, e.g., hazardous liquids need various methods than standard transport packages in the vehicle. Consequently, we specified our experiments on qualitatively various problems via identifying different means of transport and products. In the initial plan, the number of task issues varies between one to six tasks. The proposed technique is performed over two different sets of problems: dependent and independent problems over two different domains.

Fig. 2, 3, 4, 5, 6, 7, 8, and 9 show our approach's runtime behaviour compared to the traditional planning system. In our approach, the planning time means the time of splitting the hierarchical planning problem, mapper execution time, which includes the time of solving sub-problems and the time of pre-processing, and finally, the detection and merging time in the reduce phase. The system would return no solution if the planner did not find a solution within the bounded 10000 plans and 18000 seconds. The system PANDA refers to the behavior of the traditional hierarchical planning system [19]. The other two columns refer to our system's behavior by considering clustering the hierarchical planning problem through two variant clustering techniques; Dependent and Independent technique.

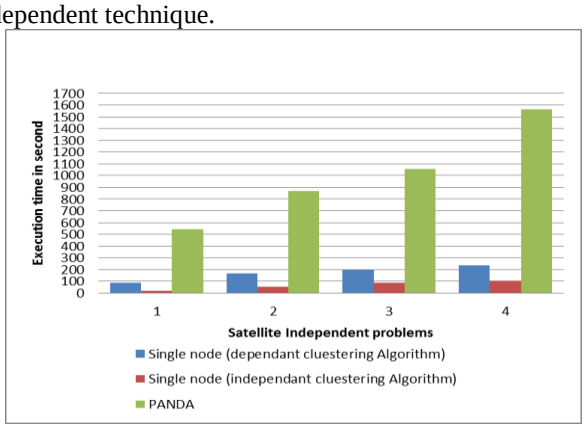

Figure 6. Independent Satellite problems (Map-Reduce single node with dependent and independent clustering techniques)

Our experiments proved that PANDA's traditional planning system achieves low efficiency either in the satellite or UM-Translog domains. In addition, it faces some difficulty in solving large hierarchical planning problems. As well as the experiments demonstrate that splitting hierarchical planning problem into various clusters using the Dependent or Independent clustering algorithm is more straightforward to solve than the original problem.

As shown in Fig. 2 and 3, in a single node, for the UM-Translog problems containing a set of independent complex tasks in the initial plan, approximately the average performance improvement of the proposed system with an independent clustering algorithm is 56% in comparison with PANDA planner. In contrast, the percentage of improvement with the dependent algorithms reaches 25%. On the other hand, when the initial plan contains abstract tasks, the proposed approach achieves improvement by about 36% and 41% with considering independent and dependent clustering algorithms, respectively, compared with the PANDA system.

As shown in Fig. 4 and 5, in multi-node implementation, the independent clustering techniques achieved better performance than dependent clustering by 8% w.r.t. independent abstract tasks in the initial plan. Not surprisingly, the proposed system's performance increases dramatically with increasing the number of tasks in the hierarchical planning problem.

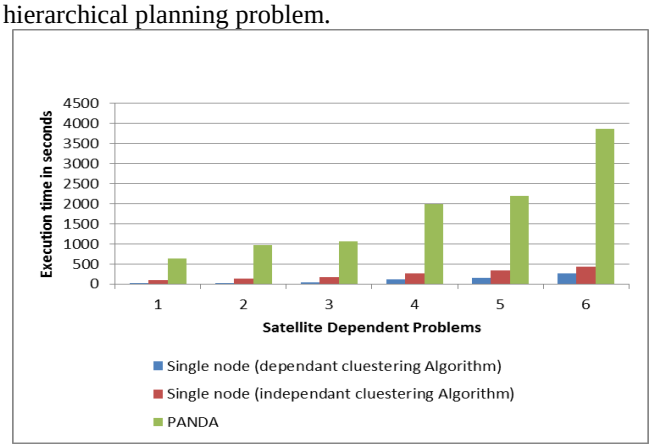

Figure 7. Dependent Satellite problems (Map-Reduce single node with dependent and independent clustering techniques)

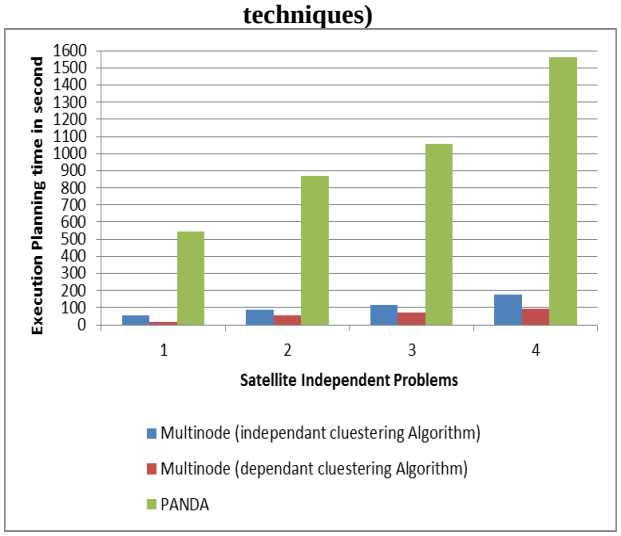

Figure 8. Independent Satellite problems (Map-Reduce Multi-node with dependent and independent clustering techniques)

As documented in fig. 7, the Independent algorithm attains an average improvement of 9% w. r. t. the Dependent algorithm in a single node, which means when the plan includes causal interaction between abstract tasks, the Independent clustering technique works efficiently than the Dependent clustering technique. While in multi-node implementation, archives 6% with the same problems. Although the satellite domain does not achieve high performance with the most hierarchical planning approaches according to the shallowest decomposition hierarchy, it accomplishes good performance with the proposed map-reduction approach, either considering the Dependent or Independent clustering technique depicted in Fig. 6, 7, 8, and 9. In addition, either in the satellite or Um-translog domains, the proposed approach succeeded in solving many problems, while the hierarchical planning system PANDA cannot solve it. Not surprisingly, applying the proposed approach in the multi-nodes Hadoop platform consumes execution time less than a single-node Hadoop platform by 13%.

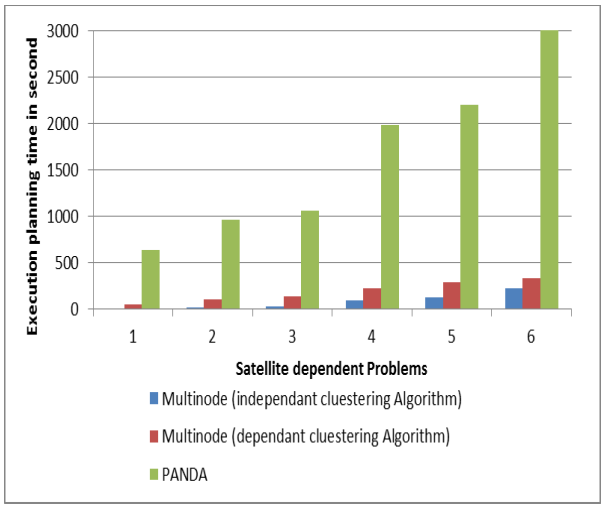

Figure 9. Dependent Satellite problems (Map-Reduce Multi-node with dependent and independent clustering techniques)

V. CONCLUSION

This paper introduced a new AI-planning technique that handling the hierarchical AI-planning by map-reduce paradigm. Our approach enables us to split the planning problem into clusters using two different techniques; Dependent and Independent. In addition, each sub-problem runs a pre-processing technique to rule out the irrelevant information from the domain model, which accelerates the planning process to find the solution plan. It guarantees that the conflicts between individually constructed sub-plans are detected and solved successfully. Hence, the global plan is generated without additional refinement in any individual plan. Our experiments' scenario relies on the run of the proposed approach over two different hierarchical domains in which the proposed approach, either using dependent or independent clustering techniques, competed with a hierarchical planner. The experiments run over 120 problems from the two mentioned domains. The experimental results proved a great improvement for the practical use of the proposed approach.

REFERENCES

1. D, Nau, M. Ghallab, and P. Traverso, "automated planning: Theory and practice," Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004.
2. R. Fikes, and N. Nilsson, "STRIPS: a new approach to the application of theorem proving to problem-solving," Artificial Intelligence, 2, 1971, PP 189–208.
3. D. Holler, P. Bercher, G. Behnke, and S. Biundo, ¨HTN planning as heuristic progression search," Journal of artificial intelligence research (JAIR) 2020, vol. (67): PP 835– 880.
4. D. Wilkins, K. Mayers, "A multi-agent planning architecture," In Proceedings of AIPS, 1998. pp. 162.
5. J. Tonino, A. Bos, M. deWeerdt, C. Witteveen, "Plan coordination by revision in collective agent-based systems," Journal of Artificial Intelligence 142, 2. 2002, pp. 121–145.
6. M. Weerdt, C. Witteveen, "A resource logic for multi-agent plan merging," In Proceedings of the 20th Workshop of the UK Planning and Scheduling, 2003, PP. 244–256.
7. A. Mors, J. Valk, C. Witteveen, "Task coordination, and decomposition in multi-actor planning systems," In Proceedings of the Workshop on Software-Agents in Information Systems and Industrial Applications (SAISIA), 2006. pp. 83–94.
8. M. desJardins, M. Wolverton, "Coordinating a distributed planning system," Journal of AI Magazine, 20(4).1999, PP. 4553.
9. H. Hisashi, "Stratified multi-agent HTN planning in dynamic environments," In Proceedings KES-AMSTA, Pp. 189–198 (2007).
10. M. Elkawkagy, S. Buindo, "Hybrid Multi-agent Planning," In Proceedings of the German Conference on Multiagent System Technologies, 2011, pp. 16-28.
11. M. Elkawkagy, H. Elbeh, "Improving AI Planning using Map Reduce," International Journal of Innovative Technology and Exploring Engineering, 9(4). 2020. PP 615-618.
12. N. Maleki, A. Rahmani, M. Conti, " MapReduce: an infrastructure review and research insights," The Journal of Supercomputing, Springer Nature, 75(10),2019, PP. 1-69.
13. M. Elkawkagy, P. Bercher, B. Schattenberg, and S. Biundo, "Improving hierarchical planning performance by the use of landmarks," In Proceedings of the 26th National Conference on Artificial Intelligence (AAAI), 2012, pp. 1763–1769.
14. P. Bercher, D. Hller, G. Behnke, and S. Biundo, "User-centered planning - a discussion on planning in the presence of human users," In Proceedings of the International Symposium on Companion Technology, 2015.
15. M. Elkawkagy, B. Schattenberg, S. Biund, "Landmarks in hierarchical planning," In Proceedings of ECAI, 2010, pp. 229–234.
16. M. Elkawkagy, P. Bercher, B. Schattenberg, S. Biundo, "Exploiting landmarks for hybrid planning," In proceedings of 25th PuK Workshop Planen, Scheduling und Konfigurieren, Entwerfen, 2010.
17. M. Elkawkagy, P. Bercher, B. Schattenberg, S. Biundo, S., "Landmark-aware strategies for hierarchical planning," In Proceedings of the 3rd Workshop on Heuristics for Domain-independent Planning, 2011, PP 73-79.
18. M. Elkawkagy, "Improving the performance of hybrid planning," International Journal of Artificial Intelligence, 14 (2), 2016, 98-116.
19. P. Bercher, D. H¨oller, G. Behnke, and S. Biundo, "On implications of preconditions and effects of compound HTN planning tasks" In Proceedings of the 22nd European Conference on Artificial Intelligence (ECAI), (2016) pp. 225–233, IOS Press
20. G. Behnke, D. Holler, and S. Biundo, "Finding opti- ¨ mal solutions in HTN planning – A SAT-based approach," In Proc. of the 28th Int. Joint Conf. on AI (IJCAI), 2019, 5500–5508, IJCAI Organization.
21. R. Goldman, and U. Kuter, "Hierarchical task network planning in common Lisp: the case of SHOP3", In Proc. of the 12th European Lisp Symposium (ELS), 2019, 73–80. ACM.
22. D. Holler, P. Bercher, G. Behnke, and S. Biundo, "On guiding search in HTN planning with classical planning heuristics," In Proc. of the 28th Int. Joint Conf. on AI (IJCAI), (2019), 6171–6175, IJCAI Organization.

AUTHORS PROFILE

Mohamed Elkawkagy, is Associate Professor in Artificial intelligence. He is a staff member at the Computer Sciences department, Community College, Imam Abdulrahman Bin Faisal University, KSA from November 2016 till now. He got his PhD in 2011 from the Faculty of Engineering, Ulm University, Germany. He was a Post-Doc at the Artificial Intelligence Research Group, Ulm University, Germany between 2011 and 2013. His research focuses on Artificial In intelligence, cloud computing, and Big data.

Heba Elbeh, is Lecturer in Artificial intelligence. She is a staff member at the Computer Sciences department, Community College, Imam Abdulrahman Bin Faisal University, KSA from November 2016 till now. He got his PhD in 2012 from the Faculty of Engineering, Ulm University, Germany. Her research focuses on Artificial In intelligence, Block chain, and Big data.