EXPERIMENTAL ANALYSIS OF MULTI-AGENT PLANNING DOMAINS WITH CONCURRENT ACTIONS

A Dissertation
Submitted in fulfilment of the requirements for the award of degree of

MASTER OF TECHNOLOGY
in
COMPUTER SCIENCE & ENGINEERING

By

Manish Kumar Dixit
(14535024)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY
ROORKEE – 247667 (INDIA)
MAY – 2016

Declaration

I declare that the work presented in this dissertation with title, "**Experimental Analysis of Multi-agent Planning Domains with Concurrent Actions**", towards the fulfilment of the requirements for award of the degree of **Master of Technology** in **Computer Science & Engineering**, submitted to the **Department of Computer Science and Engineering**, **Indian Institute of Technology-Roorkee**, India, is an authentic record of my own work carried out during the period from **June 2015 to May 2016** under the guidance of **Dr.RajdeepNiyogi**, Associate Professor, Department of Computer Science and Engineering, Indian Institute of Technology, Roorkee.

The matter presented in this dissertation has not been submitted by me for the award of any other degree of this or any other institute.

Date:

Place: Roorkee **(Manish Kumar Dixit)**

Certificate

This is to certify that the statement made by the candidate in the declaration is correct to the best of my knowledge and belief.

Date:

Place: Roorkee

Dr.RajdeepNiyogi
Associate Professor
Department of Computer Science and Engineering
Indian Institute of Technology, Roorkee

ABSTRACT

This research explores multi-agent planning where agents are not same. Unlike real world problem, most classical planner solves on a basic assumption that all agents are same. However, in most of the practical cases, Agents are of different capabilities. In this paper, we discuss shows that a slight modification in classical planning approach can be used as a planning method with heterogeneous agents. We propose an interface to handle the heterogeneity in agent`s capability.Since it is not desirable that we build a separate planner for heterogeneous agents, thus an existing planner is used. In this paper, we show how an existing classical planner with an extra interface can be proved very useful in solving heterogeneity problem of agent in planning.

Acknowledgements

I would never have been able to complete my dissertation without the guidance of my supervisor, help from friends, and support from my family and loved ones. I would like to express my deepest gratitude to my supervisor, **Dr.RajdeepNiyogi**, for his excellent guidance, meaningful insights and moral support.

I am also grateful to the Dept. of Computer Science, IIT-Roorkee for providing valuable resources to aid my research. I would like to thank Satyendrasinghchouhanwho supported me, and was always willing to help and give me his best suggestions.

Finally, hearty thanks to my parents and siblings, who encouraged me in good times, and motivated me in the bad times, without which this dissertation would not have been possible.

Dedication

To Dada and my Sister

CONTENTS

LIST OF FIGURES

LIST OF TABLES

INTRODUCTION

A plan is a sequence of actions to reach a desired goal. The need of achieving a goal is the need to plan. The process of making a plan is called planning. For planning, there is need of intelligent entity that can think ahead, i.e. it can forecast the action and their affect. This intelligent entity is called an agent. An agent is something that works on behalf of another entity. For planning one should know that what is the input on the given problem. The input itself defines the problem statement and current environment of the given domain. As to reach a goal, there are certain action to be taken, thus these actions should be well defined as input. The meaning of defining actions is that one can deterministically say that if this action is taken by an agent, then certain well defined effect will be seen in the environment

1.1 Planning:

Every intelligent agent needs a plan to perform some actions to reach a desirable goal that's the basic need of planning. The process of making plans by any intelligent entity is called a planning. An agent is any intelligent entity that solves problems and act independently. A planning problem is classically a package of initial state which itself contains initial state and domain knowledge, and Goal state. Traditionally it is considered as all states are observable, all action will result always same for the same input and environment i.e. deterministic in nature. Thus the main cause of the forecast of action required and their effects so by the fact that action is deterministic, thus one can easily optimize resource usage to reach a goal.

1.2 Application of Planning

There is a wide area where planning is applicable. If we look closely around us we find planning all over. For example, if you see how multiple flights fly around the world without colliding this is the result of planning. Even in inter planetary exploration is an intelligent rover is required that can observe the current environment and make plan to explore the planet where it is send because planets are millions of km from each other. Thus light can take several minutes from there to earth. Thus handling any emergent situation or navigating rover is not possible remotely from earth. Thus intelligent agents are required to make plan by itself.

Planning can also be seen in transportation developed recently where autonomous trucks and other vehicles drive itself from one place to another. These are designed to have fully GPS signal locating and knows what to do if something happens on the way. Like if a vehicle is heading to a point B from point A and some road is blocked by day long traffic then as an intelligent agent it will calculate an alternate and safe path to reach the goal i.e. point B. So we can see, as the application of machines growing, the need of plan also grows as all we want to not to instruct any machine all the time for silly works. Thus need of planning is to autonomous work.

1.3 Multi-agent Planning

In the past few years, multi-agent planning has grown rapidly since use of multiple simple entities increases over a single complex entity to automate work. But when multiple agents are in domain which are working on the same problem than in that case multi-agent planning (MAP) comes into the picture. In multi-agent planning environment, it is considered that the agents will cooperate to reach a common consensus. This consensus between agents to reach a common goal state is called a multi-agent plan and even though agents can have their own goal, but the union of all the agent`s goal should match a common goal state. Multi-agent planning is widely useful as in real world multiple agents are preferred to get work done faster. For example, if on moon, three rovers are send to collect its soil and rock data. Now there are multiple entities in the domain what work should be done by which rover, is totally depend on planning. As an intelligent being it work independently, but as they are cooperative in nature so they can divide the goal. Otherwise there may be a case that all are collecting soil from the same place but mission needs it from

different location. So the point is without planning, how will one knows that all are not doing the same work and resources are not getting wasted.

In multi-agent planning there are multiple issues like consensus, flocking etc. But this report will not focus on these issues. This research work is dedicated to the situation where there are agents with different capabilities. A good planning entity should not consider such agents as same. Thus the focus of this research work is multi-agent planning with agents which are differently capable (heterogeneous).

1.4 Dissertation Overview

In Chapter 1, we overview the topic by its introduction. The next chapter, Chapter 2 discuss a little about the background of multi agent planning where some basics about the topic are discussed. This chapter will help to understand the problem. Later in this chapter, we discussed the domains on which our system is tested. Then planning problem will show a basic definition about problem.

The Chapter 3 will discuss the related work done in this field. Later on the chapter different approaches of multi-agent planning are discussed.

The Chapter 4 will brief about the proposed system. This will contain the specification of system as well as define working of architecture.

The Chapter 5 Will discuss about how the proposed system is implemented later in the chapter results are shown using few tables and figures of output.

The chapter 6 will summarize the dissertation and tell the future plan of this research work.

CHAPTER 2

BACKGROUND

When Some intelligent want to achieve or get something, there are sequence of steps it takes to reach the goal. For example, if a lion needs food, he knew that food does not come itself in its mouth so first lion go where its food might exist.Then after finding location he has to find a way to hunt it down. So we see the following rule of obeying sequence of steps is built in nature. Thus if an entity is intelligent, it has to find out some sequence of steps to achieve anything desired. These sequence of steps called Plan and the method of making a plan is called planning.

2.1 Agent: Introduction

As technology evolving faster than human evolution, an emerging field get focus called Artificial intelligent. An agent is an entity that work on behalf of another entity. For example, if there is a meeting in your office and you know that you cannot make it in time, you send your colleague to attend it on behalf of you and you told him what to do, thus in that case you friend called as agent working on behalf of you.

There are many places where human either cannot go to work or don't wanted to. Thus we need something or someone who can work behalf of human as machines are doing so far. But there are certain places where if you want to send a machine, it should be intelligent enough to observe the current environment around it and take necessary actions without or with minimum human interference. So if an entity/agent is intelligent it surely has to find out a plan to take proper actions. Thus a need of planning arises in a single based system.

2.2 Multi-agent System:

It`s a common sense that, if a single person can do a work in one day, two persons with same capability do the same work in half day if they work together. So we know mostly if we increase the number of worker for a work it would be done fast and efficient. Now the same thing applies in the area of agent based system.

If there is a work that has to be done by an agent and if we increase the number of agents, the work can be done faster. Thus multi-agent system comes in the picture. A system where two or more agents exists to work called a multi-agent system. A multi-agent system is like a group of people in which people either can their individual work to achieve goal or they can do same work as a joint action. For example, a multi-agent system is designed clean an area where multiple rock stones are in the way. So there is a chance that each agent picks up one rock and clean the area but it is also possible that there are some rocks too heavy to lift by any single agent alone, then there is need of planning that multiple agent has to pick up same object with a proper coordination. Thus multi-agent planning is the process of coordinating between the agents to reach on consensus for achieving a particular goal.

2.3 Heterogeneous Agents:

In real World scenario, there are rare chance that agents are of same capability, are used to do a particular work. As per the need and budget one can assemble agent for a work. Now imagine that there are two agents who can drive form one place to another but one agent AG1 have capacity of petrol to go only 65 km while the other agent Ag2 having capability to go 72 km. Now if a task is given to drive and reach goal at distance 55 km it would be fine to use any of the agent. But if the goal is to reach 69 km, in that case if a planner treat both agent as same it might assign the work to Ag1 which will never reach the goal as it can go maximum 65 km and Ag2 is waiting for instruction. This disaster in planning should not happen. Thus heterogeneity of agents if also a considerable area. So if planner know that Ag1 is not capable, it can only assign work to Ag2 and in that case the goal will be achieved.

2.4 **Domain Knowledge:**

We tested our Planning interface for three domains; Block world, Sokoban and Rover Domain. Each Domain represents a different world and its problems to be solved. In each domain, there are some agents which are cooperative by nature and bound to make a plan before taking any action. This assumption makes us not to care about agent's communication and behavior. All information initially is given in the input form regarding the goal and initial state. These domains are explained below:

a. Block World Domain: -

In Block world domain problem, there are a finite number of blocks or cubic in the domain. All these blocks are placed over a large table. Each block is either on the table or on another block if not than some agent is definitely holding that block. So if proposition says like *On_Table(a)* indicates that block a is placed on table directly there may or may not be any other block on it. A problem in this domain is specified by giving two sets of ground atoms,1 one specifying an initial state of the world, and the other specifying necessary and sufficient conditions for a state to be a goal state. [14]

Block world Domain is a stack puzzle where some blocks are on a table or on other blocks given as initial state. The goal is to use one or more robotic hand/agent to gain a desirable position of the blocks. Like if there are five blocks A, B, C, D and E in the domain. In the initial state, Block A, C are on table while block E is on Block B and block B itself on Block D. Now the Goal is to put Block A on B and Block C on A. So if there are two agents, they can pick up the block at the same time and first block A is placed then block B. This problem is a direct application of machines in a factory where multiple agents are used to place a large container in less time. Than planning is the only method by which each agent knows what container it can pick and what it should.

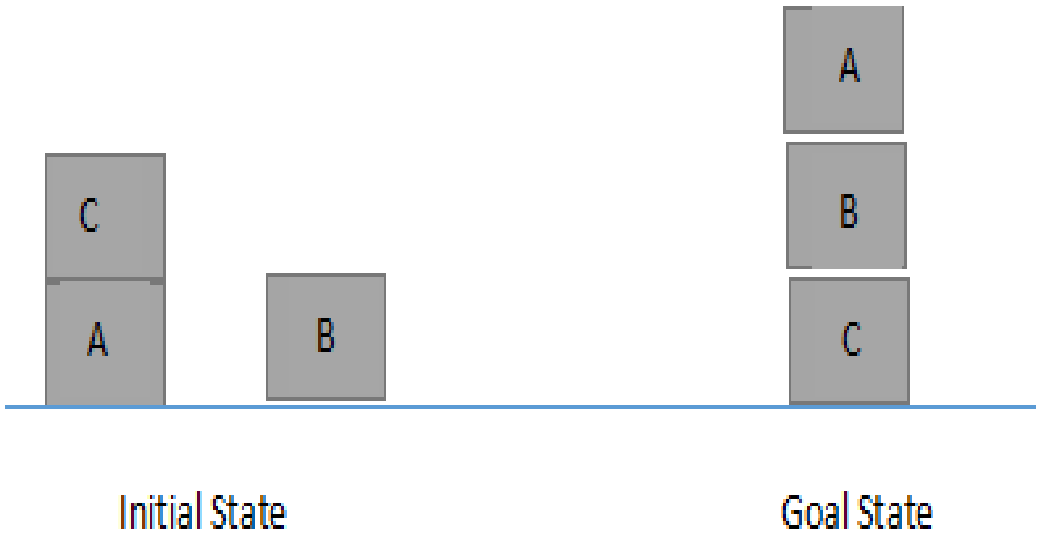

FIGURE 1: BLOCK WORLD DOMAIN

In Figure:1 above, we have shown the example which shows the initial state as block *A* and *B* are on the table while block *C* is on block *A* in the form of initial state. Now our goal is to put as shown in figure i.e. Block *C* would be on the table and above that *B* and at the top block *A* is placed.

b. Sokoban domain:

Box pushing is a classical puzzle where one or more agents are involved. The There is an initial state of each box and each agent. The rules are simple that we have to push the box until all races at goal state. An agent either can walk or Push a box left, right, up or down by one place at a time. One agent or object is considered as an obstacle to another agent or box. Thus the box can be pushed to a place only if the place is empty. One might also think that more agent will work faster,

but the catch is that if we increase an agent we also increasing one more obstacle in the system.

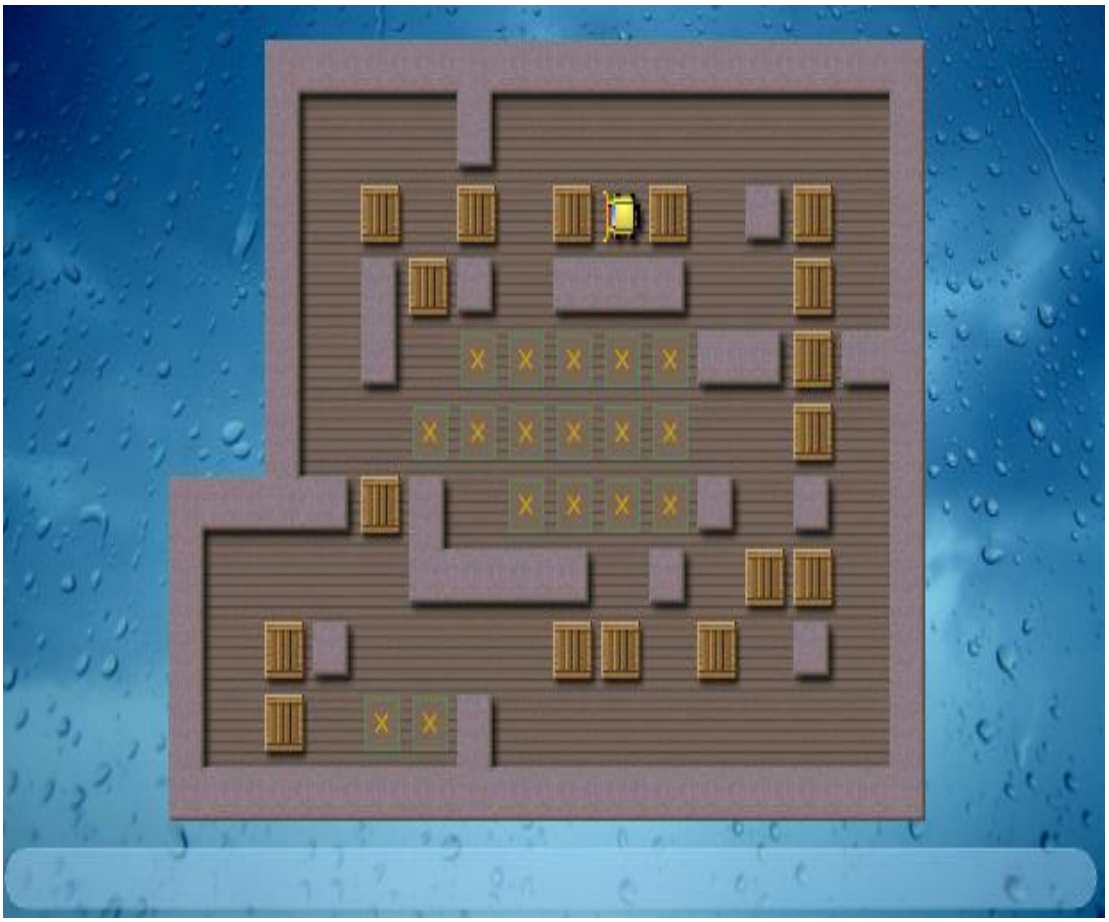

FIGURE 2 : SOKOBAN DOMAIN EXAMPLE

Sokoban is a famous puzzle game where few blocks are at a certain position on a grid. One or more agents are also there on the same grid. Now the task is to drag a block and take it to an exit point. The agent can either move carrying block or walk empty handed.

The figure: -6 above is a Sokoban puzzle instance, which provide clearance about the topic. In the figure there are few blocks (brown) and one agents colored yellow. The cross marks are showing the goal position of the blocks that is desirable. the agent has to plan such that all blocks can be pushed at all crossed places with minimum number of moves. It is also noticed that if we increase the number of agents, that may not always be efficient. As each agent would take one place so it can also prove another extra obstacle in the other agent's plan. [15]

It is also an issue in the Sokoban domain that there can a stage where deadlock may occur due to unplanned actions. Thus, if planner is not aware of such cases there may be a chance of no solution at all. In such case if we don`t understand the real cause and simply increase agents in a domain that can make the problem worse.

c. Rover domain:

In the real world, there are many places where a man wants to explore, but it`s not always that favorable environment in all places. Thus a need of intelligent entity comes into the picture that can work on behalf of human with less or no human interaction. There are many volcanos that scientist wants to explore in such case human interaction is available as using remote robots. But as we know that inter planetary distance is very large thus even communication takes an hour or days.

Sometimes an obstacle can break the communication with the rover or what if the rover is on the wrong side of the planet. In such case any emergent situation cannot be handled. So when it comes to interplanetary exploration, we need an intelligent roofer that can take decisions and plan for exploration. [16]

It is simple and better that we understand rover domain by an example. Let’s suppose there is mission launched to explore the surface of planet Mars. Now there are two rovers on the board for exploring. What should be the exploration strategy to explore the maximum area using minimum resources. So two rovers should not explore the same area for maximum coverage.Thus a plan is required to do so.

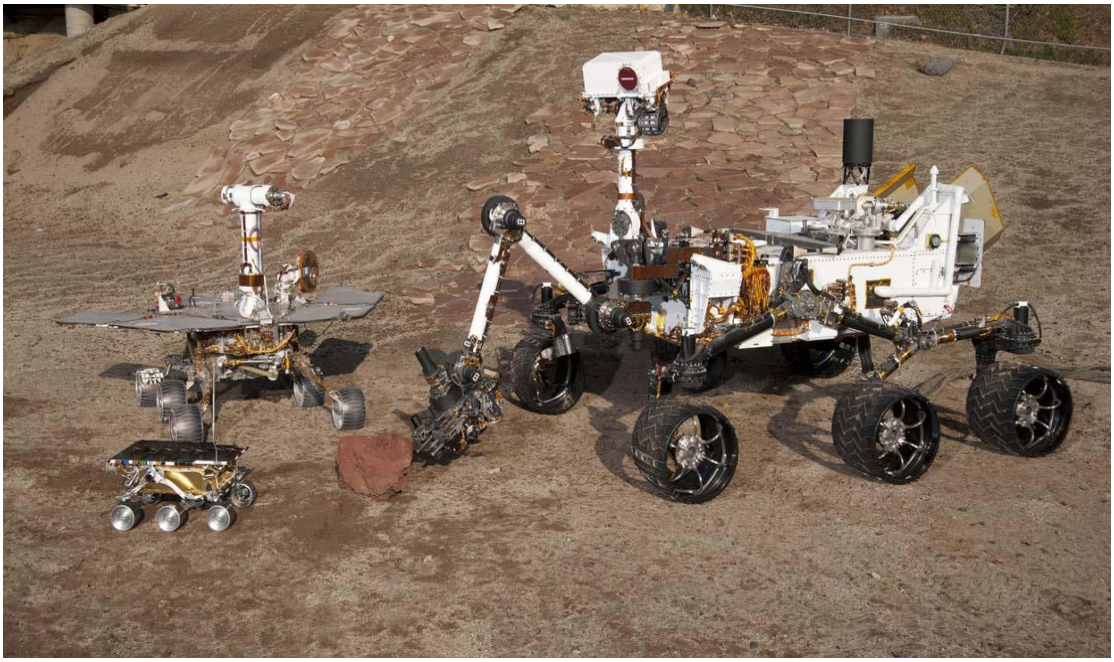

FIGURE 3 : ROVER DOMAIN EXAMPLE

The Rover domain is an exploration problem with minimum usages of resources. There can be multiple Rovers deployed to explore fast and efficiently. Thus, there should be plan that which direction will be explored by a Rover and it also has to be ensured that rover should avoid to explore common area, i.e. The area explored by each agent should be mutually exclusive for maximum usages. In the end a final map will be available as an output.

2.5 Planning Problem:

A problem can be solved only if one can define it. So there is always an effort to formulate a model for any problem. The conceptual model of planning is seen as finite transaction state machine since planning is concerned for choosing actions and change the state of the system.

Formally, the state transition system of planning has 4-tuple $E = (S, A, E, \$)$,

- Where S is a finite set of states;
- A is a finite set of actions;
- $E = \{e1, e2, e3....\}$ is finite set of events;
- $\$: S \times A \times E \longrightarrow 2S$ is state transition function;

A state transition system can also be seen as a directed graph having state as its node. Now if *st` ∈ $(s, p)* where p is a pair *(a, e),* and *a ∈ A* i.e. actions and *e ∈ E* i.e. events. Then in the graph there is an arc from node s to node st`.

Let`s suppose there is a robotic hand that can Pick up or Putdown the boxes fixed on a rover working independently from rover i.e. even robotic hand is fixed on rover still both are separate intelligent entity. The rover can go one place to another and can carry these boxes. But rover is not able to pick up the boxes only robotic hand can. So here state set S is*{s0, s1, s2, s3, s4, s5}* and the actions in the set A are *{Pickup, Load_on_ Rover, Unload _from_Rover,Go_point1, Go_Point2}*. Now Pickup, *Load_on_ Rover* and *Unload _from_Rover* are actions of robotic hand while GO_point1 and GO_point2 are actions of rover that instruct it where to go.The set of E is empty as no event is there. The goal of the problem is to transport all boxes to *point 2.*

Suppose boxes are on *point 1* and rover with robotic hand and is on *point 2*. Now the arc from s0 to s1 will be labelled with action *Go_point1* so rover will reach at *point 1*, Then the next arc will be from s1 to s2 having label as *Pickup* so robotic hand will pick a box.Now robotic hand has the box so next arc will be from s2 to s3 labelled as *Load_on_ Rover* and the box is loaded on rover. Now from s3 to s4 an arc will go labelled as *Go_point2* thus rover will carry the box to point 2.Now from s4 to s5 there is an arc having label *Unload _from_Rover* will effect as unloading the box from rover. This s5 is final state of the machine as the goal is reached.As multiple boxes are there the same process repeats. Thus we can see,each transition is deterministic by nature.[17]

2.6 Multi-agent Planning Problem

An MA-STRIPS problem can be represented as a quadruple $Q = \{ P, \{Ai\}_{i=1}^{k}, I, G\}$, Where,

- P is set of proposition which uses to deterministically define actions and effects.
- For i=1 to k, A_i represent the set of actions that are allowed i.e. an agent can perform action only defined in set A_i. Also each action a∈ A, is $\cup A_i$ just as defined in Strips standards.

- Thus mathematically we can say *a={pre(a), add(a), del(a)}* i.e. there are some *pre(a);* precondition to perform action a, *add(a);* added effect in the environment due to action *a* and *del(a);* Some effects that are deleted from the environment.
- *I* represent the initial condition given in the problem. This is the starting node to solve the problem when we convert problem to graph. $I \subseteq P$, i.e. set of proposition that are given true initially.
- G represent goal state in the problem. i.e. what is the goal of solving the problem. $G \subseteq P$ i.e. proposition should be true in the end of the solution. [21]

2.7 Planning Method:

Blackbox planner feed input in STRIPS or PDDL standard. It solves the problem by first preparing graph of the given problem. Then the graph constructed by the Blackbox later is transformed into CNF wff. Later either the graph planned by planner is used as it is or a SAT solver is used. There are few solver options available in the Blackbox in which Chaff solver is a by default solver. As the problem is converted to the Graphplan than it is easy efficient to simplify the Graphplan than that of SATPLAN. [18]

Both problem file and domain file either feed in PDDL or STRIPS. Only those propositions that mentioned true in the input file, are considered true. Each file has some Precondition and Goal of the problem defined. Also specification of agent is not considered in this planner.

In[18],Fikes and Nilsson shows a new approach in problem solving called STRIPS. Here the effort is used to a real world problem into problem model. The problem is defined as initial state goal state, actions and their effects in term of propositions.

It is a general thinking, that Ai planning is all about how one intelligent entity logically deduct facts and use knowledge to interpret them. STRIPS system *(Fikes and Nilsson 1971)* was proved very convincing about this.

As AI grows researchers started to focuses on the planning algorithm and as in very early phase already existed algorithm in theory are not seemed to be working, researcher started to believe that we need some new practical algorithm. This mindset of special algorithm for planning is countered by *Kautz and Selman* (1992,1996). The result they shown as SATPLAN. Thus SATPLAN shows

that even simple propositional theorem is very useful comparing over many complex planning algorithms**.** [20]

Blackbox = Graphplan + satplan

So if we look closely, we find that Blackbox uses STRIP notation to formulate or better be say define a problem in form of predicates, then it uses SATPLAN to solve the given problem. Each problem is first converted into graph. Each state in the given problem is assigned as the initial state in the graph and later on as the solution started to expand new nodes are created as intermediate state. The process stops when defined Goal state in the problem file is reaches. The path in graph which leads to the goal state is shown as the plan.

CHAPTER 3

RELATED WORK

Multi-agent planning (MAP) can be defined as “the problem of planning by and for a group of agents” [2]. Multi-agent planning (MAP) can be defined as “the problem of planning by and for a group of agents”. Multi-agent planning approaches can be divided into centralized and distributed approaches. In a centralized approach, a centralized planner plans for all the agent in the system. Some centralized approaches have been proposed in [3, 4, 5]. Centralized multi-agent planning are mainly used to increase the performance of the planning.

In some situation, agents are capable of computing the plans. There is no centralized planner exists that has the complete knowledge of the system. In this situation, each agent coordinates and cooperates with other agents via communication. This approach is known as distributed multi-agent planning. In the recent years, some notable works have been done in [6, 7] for distributed multi-agent planning.

In this paper, we emphasis on the heterogeneous agents having different capabilities to perform the actions. There are very few works exist that emphasis on these types of assumptions. Now, we present the works that are closely related to our work.

In [13], A scenario is discussed that if an agent is capable of particular action, then in that case agent can have some different and extra algorithms and resources. Also, if an agent is able not solving the problem itself it can ask another agent on behalf of it i.e. contract based working. But if there is no such communication than how the capability of the agent would be decided.

In [8], An approach from single agent planning to multiple agent planning is discussed. Where they exploit the area of temporal planning. Interestingly, a scenario is briefly discussed that agents can be heterogeneous. Thus, as a part of solution about handling heterogeneous agents, each action has some capability. For Example, if there is an action in a domain called Pickup and an agent is capable of picking up object A then an extra predicate is added in the problem file named cap-Pickup (A) which shows that either agent is capable of picking up or not. Thus, as initial state is now having a predicate of cap-Pickup (A).

3.1 Approaches of multi-agent planning:

There are following two famous approaches of multi-agent planning; 1.Centralized, 2. Decentralized. [7]; Centralized approach is highly identical to the traditional single agent control philosophy where a central station is always there as manager, which makes it simpler to organize. While in distributed approach, there is nothing like central station. This causes higher structural complexity and harder to organize.

In a distributed control group of agents, the main purpose typically to achieve a single or group of tasks cooperatively. Even though both approaches have their pros and cons still distributed approach is more promising in real application because of multiple physical constraints like limited resources and energy, short wireless communication range, limited bandwidth etc. In a multi-agent system, the main objective is to work cooperative fashion which is typically achieved by information sharing.

In Centralized planning policy, the problem solving technique is a mapping of joint actions of the agents to the goal state. Thus first all agents combine their action and pass through as a single joint goal. On bases of thought, a planner decides a Common multi-agent plan for all agents. Later the joint action is split up into agent`s individual goal [8].

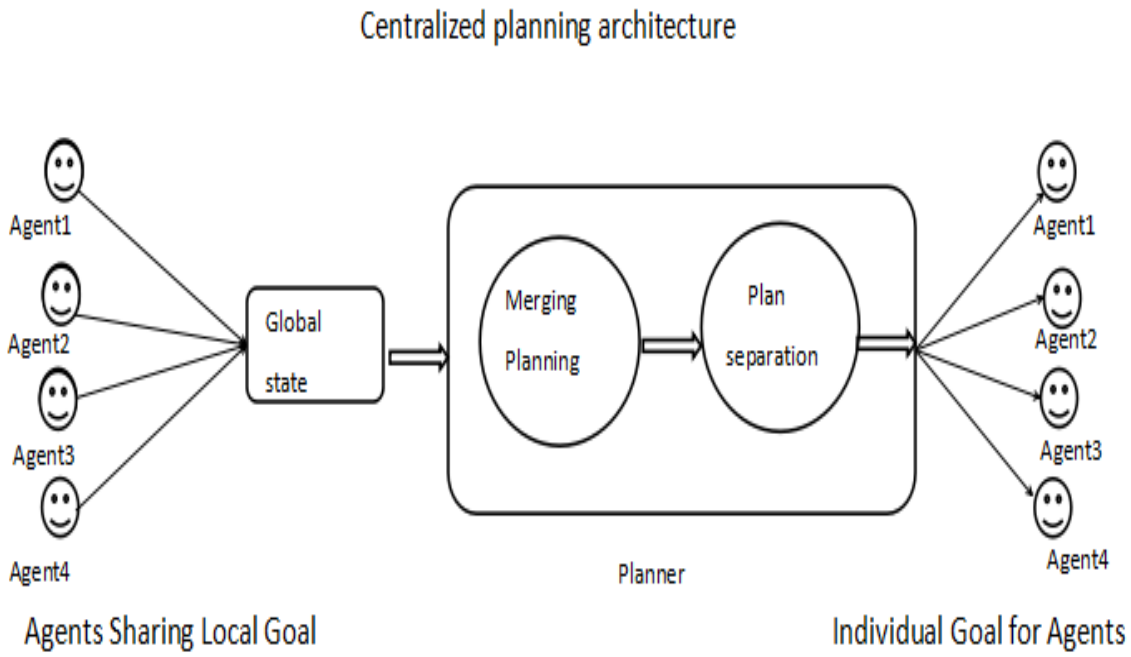

FIGURE 4 : CENTRALIZED PLANNER ARCHITECTURE

In centralized planner when the goal is merged or aggregated there may be a chance of conflicts. Thus, a planner should design in a way that can handle such level conflicts in decision making. Further the plan gets separated and each agent is guided to do some work, i.e. the part of each agent in the plan is distributed among agents so now they can work independently to reach a common goal.

While in real world scenario, agents have to calculate a goal fast, according to need thus the need of decentralized approach emerges. In decentralize approach, the agent first takes action locally, then it analyzes that if there is a need for communication. If such a need is felt by an agent than in that case agents communicate and in the end of communication phase, both are having an update set of their local knowledge.

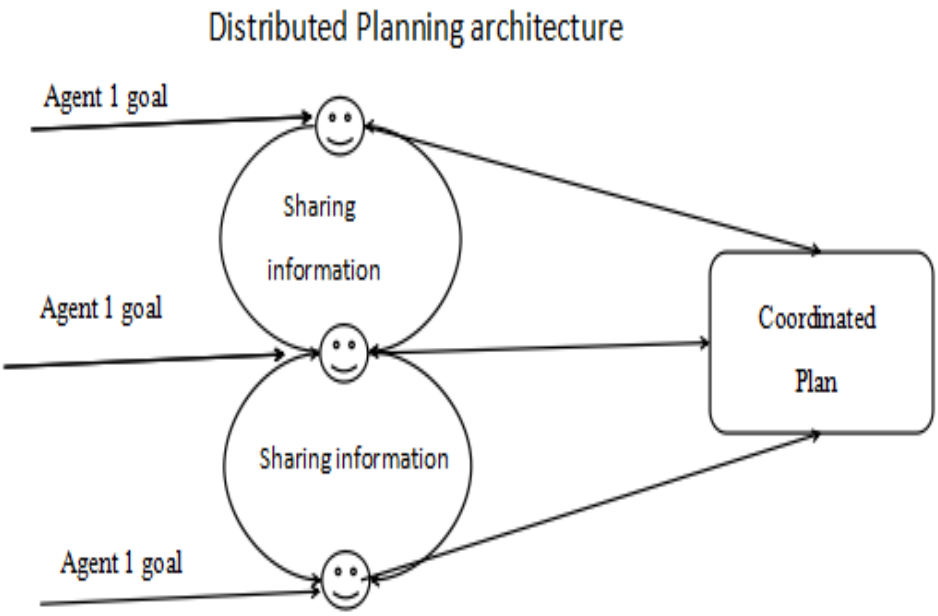

FIGURE 5: DISTRIBUTED PLANNER ARCHITECTURE

The difference in both approaches is that, in centralized approach first a global goal state is found, then the actions are taken by agents, but in decentralized approach, there is a need for external communication. So a planner has to deal with communication, decision and also have to take care of history of agents and their communication. This is why, a centralized approach is called a simple approach of multi-agent planning relative to decentralized one.

CHAPTER 4

PROPOSED WORK

This section will describe the architecture of our approach of multi-agent planning with the help of classical planner. We also describe our classical planner Blackbox later in the section.

In this model, agent's capability is not considered. In our work, we have extended the MA-STRIPS model. Here, each agent has some capabilities and each action a ∈ A requires some capabilities to perform that action.

4.1 Proposed MAP system Architecture and working

Our proposed multi-agent planning system involves heterogeneity at agent level, i.e. the agents involve in the plan may have different capacity or ability to do the same work like most of the real world scenario. We used Blackbox planner which uses PDDL files as input. We have below discussed our system features like input file specification, Domain knowledge and planning method;

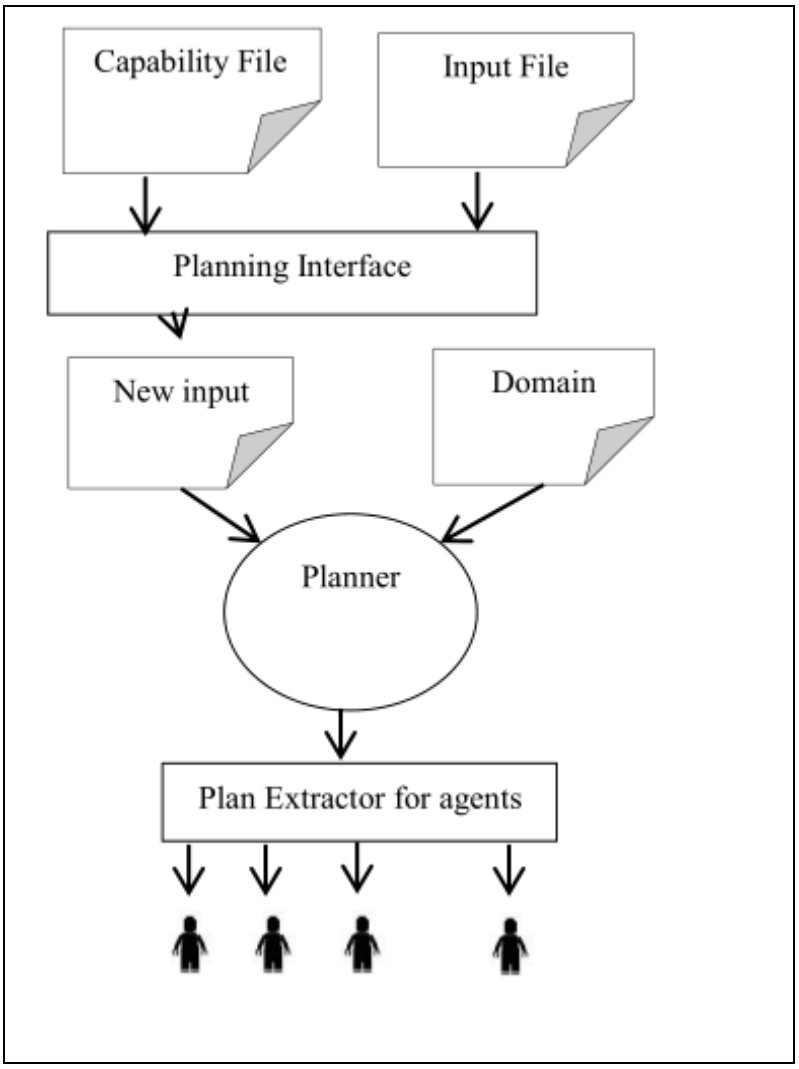

FIGURE 6:ARCHITECTURE OF PROPOSED SYSTEM

In the, above diagram, we Have shown the flow our interface actually works. Only emphasizing on our current problem, we decided to use Blackbox planner which gives concurrent output with timestamp. The advantage of having timestamp in output is that we can easily know that what actions are performed parallel. So as per Blackbox using criteria, we use PDDL format to define one problem. These problem are from three domains; Block world, Sokoban and Rover. Thus we collect the resources about the domain i.e. Domain and input files in PDDL files. For understanding about the problem file one should first study about STRIPS. So we got knows that each condition is in the form of proposition. Only those proposition that are mentioned in the file is considered as true.

4.2 Input file specification

In input file the domain knowledge and set of actions that an agent is allowed to perform, is stored. The capability input file is different from the classical input PDDL. It stores the capacity of an agent to perform a specific action and also capacity required to perform an action on a specific object.

Thus, it stores the agent and object capacity parameters. The initial state is mentioned in the input file along with goal state of the particular planning problem. The image below shows an example of PDDL file. The first line of the PDDL file shows the name of the problem is" simple". The second line defines the domain and objects here as this file is of Sokoban domain, the domain here is Sokoban,and the objects field contain objects and agents. The next line explains *init*: i.e. initial condition for problems. The initial condition shows what the status of agents and objects is. The next line shows the goal of the problem. To achieve this goal planner has to generate a graph plan having this goal as final state.

```
(define (problem simple)
(:domain sokoban-domain)  (:objects s_1_1 s_1_2 s_1_3
s_2_1 s_2_2 s_2_3 s_2_4 s_3_2 s_3_3 s_3_4 s_4_3 s_4_4
s_5_4 s_6_4 ag1 ag2 ag3 ag4 ag5 ag6 ag7 ag8 ag9 ag10
ag11 ag12 ag13 ag14 ag15 ag16 ag17 ag18 ag19 ag20)
 (:init (adjacent s_1_1 s_2_1) (adjacent s_1_1 s_1_2)
(adjacent s_1_2 s_2_2) (adjacent s_1_2 s_1_3)
(adjacent s_1_2 s_1_1) (adjacent s_1_3 s_2_3)
(adjacent s_1_3 s_1_2) (adjacent s_2_1 s_2_2)
(adjacent s_2_1 s_1_1) (adjacent s_2_2 s_3_2)
(adjacent s_2_2 s_2_3) (adjacent s_2_2 s_1_2)
(adjacent s_2_2 s_2_1) (adjacent s_2_3 s_3_3)
(adjacent s_2_3 s_2_4) (adjacent s_2_3 s_1_3)
(adjacent s_2_3 s_2_2) (adjacent s_2_4 s_3_4)
(adjacent s_2_4 s_2_3) (adjacent s_3_2 s_3_3)
(adjacent s_3_2 s_2_2) (adjacent s_3_3 s_4_3)
(adjacent s_3_3 s_3_4) (adjacent s_3_3 s_2_3)
(adjacent s_3_3 s_3_2) (adjacent s_3_4 s_4_4)
(adjacent s_3_4 s_2_4) (adjacent s_3_4 s_3_3)
(adjacent s_4_3 s_4_4) (adjacent s_4_3 s_3_3)
(adjacent s_4_4 s_5_4) (adjacent s_4_4 s_3_4)
(adjacent s_4_4 s_4_3) (adjacent s_5_4 s_6_4)
(adjacent s_5_4 s_4_4) (adjacent s_6_4 s_5_4)
(adjacent_2 s_1_1 s_1_3) (adjacent_2 s_1_2 s_3_2)
(adjacent_2 s_1_3 s_3_3) (adjacent_2 s_1_3 s_1_1)
(adjacent_2 s_2_1 s_2_3) (adjacent_2 s_2_2 s_2_4)
(adjacent_2 s_2_3 s_4_3) (adjacent_2 s_2_3 s_2_1)
(adjacent_2 s_2_4 s_4_4) (adjacent_2 s_2_4 s_2_2)
(adjacent_2 s_3_2 s_3_4) (adjacent_2 s_3_2 s_1_2)
(adjacent_2 s_3_3 s_1_3) (adjacent_2 s_3_4 s_5_4)
(adjacent_2 s_3_4 s_3_2) (adjacent_2 s_4_3 s_2_3)
(adjacent_2 s_4_4 s_6_4) (adjacent_2 s_4_4 s_2_4)
(adjacent_2 s_5_4 s_3_4) (adjacent_2 s_6_4 s_4_4)
(has_player s_2_1 ag12)(has_player s_3_1 ag1)
(has_box s_2_2) (has_box s_2_3))
  (:goal (and (has_box s_1_1) (has_box s_6_4))))
```

FIGURE 7 INPUT FILE IN PDDL FORMAT

4.3 Capability Matrix specification

In input file we describe two types of capability matrix. First matrix is between agents and actions which define the capability of a particular agentto perform a specific action. The second matrix is about objects present in domain and action allowed. This matrix defines the capability required to perform a specific action over a specific object.

TABLE IAGENT CAPABILITY TABLE

Actions	Ag1	Ag2	Ag3
Pickup	5	8	7
Walk_with_package	8	11	4

TABLE IIREQUIRED CAPACITY FOR AN ACTION ON AN OBJECT

Actions	A	B	C	D
Pickup	4	7	3	6
Walk_with_package	9	10	6	11

For example, if there are robotic agents Ag1, Ag2 and Ag3 used to transport things from area 1 to area 2 and there are 4 objects A, B, C and D, to be transported. Now possible actions allowed in the domain are to Pickup and Walk_with_package. Now consider if Ag1, Ag2 and Ag3 can Pick up object not more than 5kg, 3 kg and 7kg respectively, and objects A, B, C and D are weighted as 4kg, 7kg 3kg and 6 kg respectively. Similarly, each agent capacity to Walk_with_package is 8, 11

and 4 however, to Walk_with_package required capacity is 9, 10, 6 and 11 for A, B, C and D respectably.

Now take a look at Table1 and Table II, only Ag2 and Ag3 can Pick up object B but if Ag3 pick up the package then it cannot Walk_with_package. Thus, it has to be done by Ag3.

4.4 Agent's configuration

For planning purpose, Blackbox needed input and domain file as initial input. The input file contains the configuration of agent. The configuration of agent means how many agents do we have or what is the status of the agents as planner should know the status of an agent. An agent may be offline or it may not be capable of some actions that a planner wants to perform through this agent. It is also important to know that whether an agent is free or not. For example, if there is a domain of construction and there are multiple agents in the domain having different capabilities and working skill. Now if planner needs to weld a structure by agent Ag23 but agent Ag23 is involved in another critical welding work at the base of the structure. So now if thee planner assign work without concerning the current work of Ag23, as the result the agent has to leave the work in between. This simple ignorance of one status point may cause the destruction of whole structure and all work vanished in seconds. Thus its recommend to mention the status of intelligent agents we are dealing with.

The capabilities of agents also matter as we are dealing with heterogeneous agents here. The capability of each agent to perform a particular action and the required capability of an action to perform on a particular object, is mentioned in the capability input file which is different from original input file. In capability input file each agent is associated with the capability i.e. heterogeneous agents. An agent cannot perform any action greater than its capability. The input file also defines initial and goal condition of problem in term of true propositions. So each agent can have different initial and goal state.

CHAPTER 5

IMPLEMENTATION AND RESULTS

In this section, we describe our proposed systems implementation. We do a brief analysis of proposed multi-agent planning system. Table III shows the configuration of the environment, which we used to run our proposed system;

TABLE IIISYSTEM CONFIGURATION

Processor	Intel Core™ i5-M460 CPU, 2. 53GHz
RAM	8GB
System type	64-bit operating system
Operating system	Ubuntu 15. 10

In the proposed system, we developed an interface between input and classical planner. The interface is developed in the JAVA language. The developed system is tested over 3 different domain Box-Pushing, Sokoban and Rover domain so that we can show that it's not so domain dependent. This is a general interface which works as a middle layer of implementation. It uses classical planner as well as provide support for heterogeneous agents.

In problem file, there are few things defined firstly initial state in which it is defined that which proposition are true. For example, in a domain of box picking the initial state can be that *agent_hand(empty)* or*agent_hand(A)* i.e. either the agent hand is empty or its holding some object

(here object A). Here in addition the capability is also added as extra proposition as proposed work. We added some capability proposition on the bases of capability matrix given as an initial capability. This whole work is done by the interface developed in JAVA which read Original problem file as well as capability matrix file.

The output of the heterogeneous planning is a sequence of concurrent actions. One can easily see the plan in the figure: -4 below where A problem of Sokoban domain with 12 agents, is tested and plan is shown.

In the Sokoban domain, move-player action is the action to move a player from one place to another. In the output below, we can see that agent *Ag12* is planned to move from block *s-2-1* to place *s-1-1*.

```
blackbox version 43
command line:  ./blackbox -o sokoban-domain.pddl -f new_problem-2.pddl -solver chaff

Begin solver specification
    -maxint        0   -maxsec 0.000000  chaff
End solver specification
Loading domain file: sokoban-domain.pddl
Loading fact file: new_problem-2.pddl
Problem name: simple
Facts loaded.
time: 1, 594 facts and 1 exclusive pairs.
time: 2, 595 facts and 3 exclusive pairs.
time: 3, 599 facts and 24 exclusive pairs.
time: 4, 603 facts and 51 exclusive pairs.
time: 5, 607 facts and 81 exclusive pairs.
time: 6, 610 facts and 104 exclusive pairs.
time: 7, 613 facts and 135 exclusive pairs.
time: 8, 615 facts and 154 exclusive pairs.
time: 9, 616 facts and 149 exclusive pairs.
time: 10, 618 facts and 165 exclusive pairs.
time: 11, 619 facts and 169 exclusive pairs.
time: 12, 620 facts and 176 exclusive pairs.
Goals reachable at 12 steps but mutually exclusive.
time: 13, 620 facts and 162 exclusive pairs.
Goals reachable at 13 steps but mutually exclusive.
time: 14, 620 facts and 149 exclusive pairs.
Goals reachable at 14 steps but mutually exclusive.
time: 15, 620 facts and 140 exclusive pairs.
Goals reachable at 15 steps but mutually exclusive.
time: 16, 620 facts and 131 exclusive pairs.
Goals first reachable in 16 steps.
20750 nodes created.

##################################################
goals at time 17:
  has-box_s-1-1 has-box_s-6-4


--------------------------------------------------
** Turning off completeness check **
Converting graph to wff
number of action variables = 1805
number of fluent variables = 1467
```

FIGURE 8 BLACKBOX PLANNING

```
Begin plan
1 (move-player s-2-1 s-1-1 ag12)
2 (move-player s-1-1 s-1-2 ag12)
3 (move-player s-1-2 s-1-3 ag12)
4 (push-box s-1-3 s-2-3 s-3-3 ag12)
5 (move-player s-2-3 s-1-3 ag12)
6 (move-player s-1-3 s-1-2 ag12)
7 (move-player s-1-2 s-1-1 ag12)
8 (move-player s-1-1 s-2-1 ag12)
9 (push-box s-2-1 s-2-2 s-2-3 ag12)
10 (move-player s-2-2 s-3-2 ag12)
11 (push-box s-3-2 s-3-3 s-3-4 ag12)
12 (move-player s-3-3 s-3-2 ag12)
13 (move-player s-3-2 s-2-2 ag12)
14 (move-player s-2-2 s-1-2 ag12)
15 (move-player s-1-2 s-1-3 ag12)
16 (push-box s-1-3 s-2-3 s-3-3 ag12)
17 (move-player s-2-3 s-2-4 ag12)
18 (push-box s-2-4 s-3-4 s-4-4 ag12)
19 (push-box s-3-4 s-4-4 s-5-4 ag12)
20 (push-box s-4-4 s-5-4 s-6-4 ag12)
21 (move-player s-5-4 s-4-4 ag12)
22 (move-player s-4-4 s-4-3 ag12)
23 (push-box s-4-3 s-3-3 s-2-3 ag12)
24 (move-player s-3-3 s-3-4 ag12)
25 (move-player s-3-4 s-2-4 ag12)
26 (push-box s-2-4 s-2-3 s-2-2 ag12)
27 (move-player s-2-3 s-3-3 ag12)
28 (move-player s-3-3 s-3-2 ag12)
29 (push-box s-3-2 s-2-2 s-1-2 ag12)
30 (move-player s-2-2 s-2-3 ag12)
31 (move-player s-2-3 s-1-3 ag12)
32 (push-box s-1-3 s-1-2 s-1-1 ag12)
End plan
------------------------------------------------------

32 total actions in plan
0 entries in hash table,
31 total set-creation steps (entries + hits + plan length - 1)
0 actions tried

######################################################
Total elapsed time:   6.61 seconds
Time in milliseconds: 6611
```

FIGURE 9: OUTPUT PLAN USING PROPOSED SYSTEM

After first phase execution of the interface, the capabilities are added into the problem file and a new problem file is generated as the result. This new file will have capabilities as propositions. These proposition will allow or block any agent to do particular action. In the same input file, the goal of the problem is also defined. Thus the new input file will also be containing the goal as previous. Now new input file and domain file both are fed to the Blackbox planner. The Blackbox

planner first convert the problem into graph as the problem is defined well in STRIPS format, then as it began to solve the problem the graph started to increase and each state become a node in the graph.

The Blackbox planner used with different heuristics solver. For faster result we use it best solver called as *chaff*. [19] The interface internally calls Blackbox to plan for the new input. So in the end of the execution of interface the plan by Blackbox is saved into a file from where one can easily retrieve the plan.

TABLE IVBLOCK WORLD DOMAIN RESULTS

Tuple (agent, objects)	Plan Length	Node Created	No. of actions in plan	Total Time elapsed
2,7	8	1276	10	0.11 sec
3,8	5	1339	10	0.050 sec
4,9	4	2283	12	0.083 sec
5,10	5	2846	14	0.339 sec
6,11	6	4002	15	0.589 sec
7,12	6	6292	16	0.983 sec
8,13	6	7724	18	1.62 sec
9,14	6	7155	16	0.96sec
10,15	6	8895	18	1.54 sec
11,16	7	8985	20	2.45 sec

Table-IV, V and VI are for Block World Domain, Sokoban Domain and Rover Domain respectively. The first column in each table is a tuple of agents and objects used in the problem.

For example, if an entry is 3, 10 means there are 3 agents and 10 objects are used in a given problem. The next column is named as “Plan length” shows the number of actions that will not execute concurrently. Thus “Plan length” will always be less or equal than the total action in the plan. The third column “Total Time elapsed” is the time elapsed in solving problems for that tuple and it is an average of time elapsed for 4 different problems with same tuple. Next Column is the number of nodes created in the graph to solve this multi-agent planning problem. Than Total actions planned is in column “No. Of actions in plan”.

TABLE V SOKOBAN WORLD DOMAIN RESULTS

Tuple (agent, objects)	Plan Length	Node Created	No. of actions in plan	Total Time elapsed
2,7	8	2112	8	0.06 sec
3,8	5	2074	8	0.12 sec
4,9	4	16429	10	0.32 sec
5,10	5	8253	10	0.85 sec
6,11	6	2134	10	1.89 sec
7,12	6	229336	10	1 min 42 sec
8,13	6	212773	12	1 min 5 sec
9,14	9	1555850	16	3 min 4 sec
10,15	6	3574	12	2.57 sec
11,16	7	162429	12	45.01 sec

TABLE VIROVER DOMAIN RESULTS

Tuple (agent, objects)	Plan Length	Node Created	No. of actions in plan	Total Time elapsed (sec)
2,7	9	233	12	0.77 sec
3,8	6	432	12	0.84 sec
4,9	6	7623	8	2.76 sec
5,10	7	8834	9	5.85 sec
6,11	6	7162	10	1.45 sec
7,12	5	32748	11	3 min 23 sec
8,13	6	23847	7	2 min 25 sec
9,14	11	42344	15	2 min 6 sec
10,15	10	84393	17	2 min 34 sec
11,16	8	152323	11	4 min 12 sec

CHAPTER 6

CONCLUSION AND FUTURE WORK

In this paper, we proposed an approach for multi-agent planning with heterogeneous agents having different capabilities. We have shown that two agents may have same action set, but have different capabilities to perform the action. We have used the Blackbox planner for the planning purpose.Results also shows how concurrent plan can we retrieve from classical planner even if we have heterogeneous agents. By seeing results one can easily conclude, that even using such a system will not take much time to solve the problems. Thus efficiency of planner is not affected and now planner also supports heterogeneous agents.

In the future work, we would like to extend the proposed approach to include joint actions. Joint actions are required there are places when one agent alone is not capable to take particular action on an object (e.g. Picking up a heavy box by 4 agents). We would also like to examine the proposed work on various complex domains. Since we tested on three domains here, there future work is to make a more generalized system that can support more domains.

REFERENCE

[1] Russell, S. and Norvig, P., “Artificial Intelligence: A Modern Approach, ” Prentice Hall, 2010.

[2] deWeerdt, M. and Clement, B. , “Introduction to planning in multi-agent systems, ” In journal of Multi-agent and Grid Systems, 2009, pages 345–355.

[3] Borrajo, D.: Multi-agent planning by plan reuse. In: Proceedings of the international conference on Autonomous agents and multi-agent systems(AAMAS). pp. 1141-1142 (2013)

[4] Katz, M.J.: The generation and execution of plans for multiple agents. Computers and Artificial Intelligence 12(1), 5-35 (1993)

[5] Luis, N., Borrajo, D.: Plan merging by reuse for multi-agent planning. Distributed and Multi-Agent Planning p. 38 (2014)

[6] Torreno, A., Onaindia, E., Sapena,O’.: An approach to multi-agent planning with incomplete information.In20th European Conference on Artificial Intelligence (ECAI). pp.762-767 (2012)

[7] Torreno, A., Onaindia, E., Sapena, O’: Fmap: Distributed cooperative multi-agent planning.Applied Intelligence 41(2), 606-626 (2014)

[8] Crosby, Matthew, Anders Jonsson, and Michael Rovatsos. "A Single-Agent Approach to Multiagent Planning." ECAI. 2014.

[9] Xuan, Ping, and Victor Lesser. "Multi-agent policies: from centralized ones to decentralized ones." *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 3*. ACM, 2002.

[10] Durfee, Edmund H. "Distributed problem solving and planning." *Multi-agent systems and applications*. Springer Berlin Heidelberg, 2001. 118-149.

[11] Chouhan, Satyendra Singh, and RajdeepNiyogi. "A Multiagent Planning Algorithm with Joint Actions." *Proceedings of the 4th International Conference on Frontiers in Intelligent Computing: Theory and Applications (FICTA) 2015*. Springer India, 2016.

[12] Kitano, Hiroaki. "Robocup rescue: A grand challenge for multi-agent systems." *MultiAgent Systems, 2000. Proceedings. Fourth International Conference on*. IEEE, 2000.

[13] Gurnell, David John. *A Comparative Study of Approaches to Multi Agent Plannig*. Diss. University of Birmingham, School of Computer Science, 2006.

[14] Gupta, Naresh, and Dana S. Nau. "On the complexity of blocks-world planning." *Artificial Intelligence* 56.2 (1992): 223-254.

[15] Junghanns, Andreas, and Jonathan Schaeffer. "Sokoban: Enhancing general single-agent search methods using domain knowledge." *Artificial Intelligence*129.1 (2001): 219-251.

[16] Marecki, Janusz, Sven Koenig, and MilindTambe. "A Fast Analytical Algorithm for Solving Markov Decision Processes with Real-Valued Resources." *IJCAI*. 2007.

[17] Ghallab, Malik, Dana Nau, and Paolo Traverso. *Automated planning: theory & practice*. Elsevier, 2004.

[18] Cao, Yongcan, et al. "An overview of recent progress in the study of distributed multi-agent coordination." *Industrial Informatics, IEEE Transactions on* 9.1 (2013): 427-438.

[19] Kautz, Henry, and Bart Selman. "BLACKBOX: A new approach to the application of theorem proving to problem solving." *AIPS98 Workshop on Planning as Combinatorial Search*. Vol. 58260. 1998.

[20] Kautz, Henry, and Bart Selman. "Unifying SAT-based and graph-based planning." *IJCAI*. Vol. 99. 1999.

[21] Brafman, Ronen I., and Carmel Domshlak. "From One to Many: Planning for Loosely Coupled Multi-Agent Systems." *ICAPS*. 2008.