CS297 Report
Online Collaborative Time Management System using Artificial Intelligence.

Anand Sivaramakrishnan
anandsk123@gmail.com

Advisor: Dr. Chris Pollett

Department of Computer Science

San Jose State University

Spring 2008

Abstract

Online Collaborative Time Management System is a system that will plan events and help achieve goals intelligently using Partial Order Planning. This system is similar to a social networking site, which plans events for collaborative set of people. For this masters' project, we will develop such a system. Our site will allow multiple people to add items to a collaborative To-Do list. The user interface for our site will allow people to add actions and preconditions to the existing system, enabling the system to grow constantly. In order to generate a sequence of actions to the most complex problems, we will implement partial order generators at the back end. One of the algorithms has already been implemented, which is explained later in the paper. These Planning algorithms will have to be designed or chosen such that they will resolve conflict by implementing backtracking at whatever stage it has to be done. Since, this system is collaborative, it can have many conflicts during the generation of large plans, during which backtracking will have to be implemented.

Table of Contents

List of Figures

1. Introduction

Online Collaborative Time Management System project will implement an intelligent online collaborative system for scheduling different people's TO DO lists according to their daily routine. This system will differ from all other existing planners such as Google, Planware and GradeMate, as it will not stress on planning with respect to time, but will stress on planning logically to solve complex problems and tasks. It will be collaborative in nature to enable planning between groups of people. Partial Order Planning will be the backbone behind successful implementation of the system. The most special thing about this system is that it is dynamic in nature, as it will be constantly growing.

The problem is that currently there is no tool available online to make plans for the most complex situations, which needs intelligence. For any event or goal, there are many important issues to deal with, and it is not possible to remember all of them. Furthermore, it is further difficult to solve all the issues intelligently. The existing planners online only have solved the first problem of remembering all the issues remaining to be solved. A system that helps solving all these issues intelligently is not yet available online.

My goal is to solve the above problem, and make the system accessible to many people where they can share problems and solutions.

2. Building a Planner

The primary goal of my project is to build a planner that will generate the plans and the partial orders and present it to the people who want their problems solved or events planned.

A planner operates by searching through a world of states. These states are fundamentally situations, which are the possibilities while travelling from the current state to the goal state. The initial state is the problem state. This is why it is called **situation space** planner, as it is going through all the situations. Every node represents a state/situation. There are two types of planners. Progression Planner searches forward from the initial state (situation) to the goal state (situation). Regression Planner searches backward from the goal state (situation) to the initial state (situation). Progression Planner searches forward from the initial state (situation) to the goal state (situation). Regression Planner searches backward from the goal state (situation) to the initial state (situation). In my project, I will be using backward search techniques, which means I will be traversing from the goal state to the initial state to get the sequence of actions.

During the current implementation of the back end, I used STRIPS language to represent the initial state, the goal state, actions and preconditions.

3. Deliverable 1: Total Order Planning

A Partial Order Planner is an extension of a Total Order Planner. This is why in the very first deliverable; I implemented the total order planner. Total order planning refers to a strict sequence of steps that has to be followed in order to obtain a solution for a problem or a query. It is a planner, which is inflexible and rigid in nature. It generates a plan to solve a problem, which may not be the best solution to solve the problem. Total order planner is also called linear planner, because of its narrow choice of steps to be followed to solve the problem in hand. This is a single solution from beginning to end,

where in there are specified places where we have to enter data. In my implementation, I have implemented the following algorithm for my backend:

TO (P, G)

1. Termination check: If G is empty, report success and stop.
2. Goal selection: Let c be a goal in G, and let Oneed be the plan step for which c is a precondition.
3. Operator selection: Let Oadd be an operator in the library that adds c.

   If there is no such Oadd, then terminate and report failure.

   Backtracking point - all such operators must be considered.
4. Ordering selection: Let Odel be the last deleter of c.

   Insert Oadd somewhere between Odel and Oneed, call the resulting plan P'.
5. Update goal set: Let G' be the set of preconditions in P' that are not true.
6. Recursive invocation: TO(P',G') [7]

Figure 1: Algorithm - Total Order Planning

The input to the above algorithm is a text file, which contains information such as the initial state, which is the actual state in which the person trying to achieve a goal is actually at. He needs to reach the goal state from this state. The goal state is specified in the text file. The text file also has the total number of actions that are available to choose from when we decide how to traverse from the initial state to the goal state.

When I run the above algorithm with the specified text file as the input to the same, it will generate the plan. There will be just one plan that will be generated. Following this plan will take you from the initial state to the goal state. This plan takes into account all the preconditions for all the actions that are involved in the sequence of actions. This

means that it will take into account all the constraints that could come in the way realistically whilst traversing from the initial state to the goal state.

When we implement such planning algorithms, the program written also needs to implement the backtracking function, which when called resolves conflicts that arises when we design the sequence of actions in the plan. This means during the traversal of different states, when we find it is not possible to implement the next action because of its precondition that cannot be satisfied. At such a stage, we need to go back and change one of the actions that we had chosen and choose a different action that has the same effect.

4. Deliverable 2: Partial Order Planning

In the second deliverable, I implemented a partial order planner. The partial-order planner is a planner that generates a solution, which is a set of open conditions that is empty, and there is no conflict. It is a planner, which is flexible and resourceful in nature. It generates many plans to solve a problem. Therefore, we can choose the best solution out of it. A planner that can represent plans in which some steps are ordered (before or after) with respect to each other and other steps are unordered.

Since partial numbers of steps are ordered, it is called Partial Order Planner. It is said that the Partial Order Planner follows the principle of Least Commitment, which says, ‘One should make choices only about things that you currently care about, leaving the others to be worked out later’. Mountain Climbing is analogous to this principle.

UA(P,G)

1. Termination check: If G is empty, report success and stop.

2. Goal selection: Let c be a goal in G, and let Oneed be the plan step for which c is a

   precondition.

3. Operator selection: Let Oadd be an operator in the library that adds c.

   If there is no such Oadd, then terminate and report failure.

   Backtracking point - all such operators must be considered.

4. Ordering selection: Let Odel be the last deleter of c.

   Order Oadd after Odel and before Oneed

   Repeat until there are no interactions:

   - Select a step Oint that interacts with Oadd.
   - Order Oint either before or after Oadd.

   Backtracking point - both orders must be considered for completeness.

   b Let P' be the resulting plan.

5. Update goal set: Let G' be the set of preconditions in P' that are not true.

6. Recursive invocation: TO(P'.G') [7]

Figure 2: Algorithm - Partial Order Planning

As implemented in the Total Order Planning Algorithm, the input to the above Partial Order Planning algorithm is again a text file, which has the format of actions, preconditions and effects as the earlier text files of the Total Order Planning Algorithm.

The implementation of the Partial Order Algorithm was an extension of the Total Order Planning Algorithm. It had to implement Partial Orders. The generation of partial orders is done by a separate function that is called by this same algorithm when it is generating the sequence of actions. This separate function generates partial orders, which will tell

the user of this planner, the sections of the actions that it can implement in parallel and other sections, which have to be implemented strictly sequentially. The below figure demonstrates the pictorial behavior of the above planners. The Total Order Planner will generate only one of the two outputs that are shown in the figure. The goal that is being planned in the figure below is that of replacing a flat tire on the axle with the spare tire, which is on the trunk.

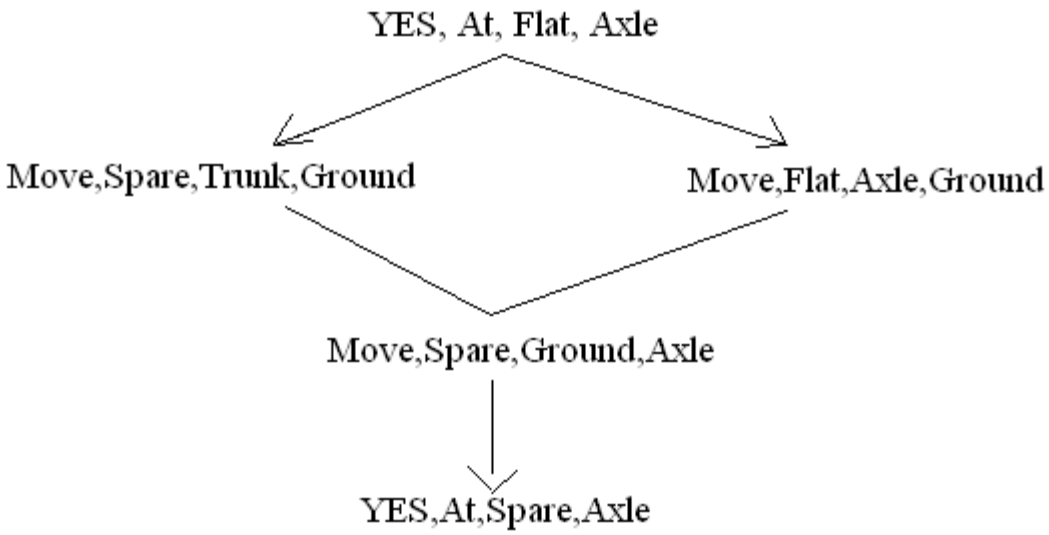

Figure 3: Total Planner and Partial Order Planner outputs for the goal of replacing flat tire on the axle

5. Deliverable 3: Object - Oriented Partial Order Planner

In the third deliverable, I converted the above coded Partial Order Planner in Object Oriented Programming by making a class outside the working functions.

Object Oriented Programming is the type of programming where we convert all the code into an object. This means that we are combining both the state and behavior into an object. Therefore, now we have packaged an object consisting of data as well as the properties. That is the beauty of Object Oriented Programming.

6. Deliverable 4: Graphical User Interface

The fourth deliverable was to create a front end or graphical user interface (GUI) for the Online Collaborative Time Management System. The front end will enable the users to interact with the system effectively. The users can add their actions and goals to the system to make the system a growing collaborative planner to work with, so that it can build convenient plans, and traverse solutions for complex problems. However, for the users to be willing to give input to the system, the system must have a very attractive user interface. It has to have minimum typing required to give the input. It should have many checkboxes and drop down menus.

As this system is a collaborative system, it needs to show interactions between different people of the same group. To design the user interface for a collaborative system is difficult and a specimen for such design is helpful. Therefore, I am using Drupal for the same.

Drupal is a content management system that is free to download from the following site:

http://www.drupal.org/

I have downloaded the latest version that is Drupal 6.2 [6]

Using the above framework, I made the GUI in HTML (Hyper Text Markup Language), and CSS (Cascading Style Sheet).

HTML files have simple markup tags which directs the browser in representing data in between the tags. HTML files have a .htm or .html extension.

Cascading Style Sheet is the technology that is used to make the GUI as attractive as possible. You need to know HTML before learning CSS. They can be used using either internal or external style sheets.

7. Future Work & Conclusion

My future work in this project implementation will be to connect the front end to the back end. The back end has been successfully implemented with respect to the examples from the book, and is yet to be tested in real world collaborative scenarios and complex problem situations. In CS 298 there are a few challenging tasks that I am planning to do. One of them is to co-ordinate the same event or goal in between many people. This is why the system is going to be collaborative. The second challenging task is to minimize the number of conflicts while creating these plans, so that we have to backtrack lesser number of times and therefore the control flow is more efficient. The third challenging task is to accommodate and re-plan existing plans in the system after new additions of actions are made to the system by the collaborative set of people.

8. References

1. Peter Norvig, Stuart Russell. (1995). Artificial Intelligence: A Modern Approach. Prentice Hall Series.

2. Sussanne Biundo, Maria Fox. (1999). Recent Advances in AI Planning. ECP, Springer.

3. Craig Knoblock, Qiang Yang. (1997). Relating the Performance of Partial Order Planning Algorithms to Domain Features. SIGART Bulletin, Vol. 6, No. 1, 8-15

4. Edwin P.D Pednault. (1988). Synthesizing plans that contain actions with context dependent effects. Computational Intelligence, 356-372

5. Stuart J. Russell. (1992). Efficient memory-bounded search algorithms. Proceedings of the Tenth European Conference

6. Drupal 6.2. (2008, April 9).

   Drupal 6.2 released, fixing security issues.

   http://drupal.org/drupal-6.2

7. Charles. F. Schmidt

   Rutgers Institute, Notes on Planning

   http://www.rci.rutgers.edu/~cfs/472_html/Planning/PlanAlg_472.html