Software Agents: Project 1

Joshua Torrance
267920

1 Introduction

The blocks world domain is a well known problem domain which due to its initial simplicity has been used frequently in the study of AI and planning. Simple blocks world consists of a set of blocks and a robots. The aim is to get the robot to stack the blocks in a given order as fast as possible.

2 Extensions to Blocks World

This project had several goals all of which extend the simple blocks world domain in various ways.

2.1 Multiple Agents

Adding multiple agents is a common addition to blocks world since this allows interesting examination of multi-agent interactions and cooperation. One way of implementing multiple agents is simply to get the agents to take turns doing actions and interleave those actions in the action sequence. This simulates simultaneous actions but it is difficult to get the robots to cooperate. A better way is to get the robots to actually take their actions simultaneously but this requires careful handling of which actions are permitted to the robots.

2.2 Block Weights

Complexity is added to the simple blocks world domain by giving all of the blocks weights and the robots strength such that a robot is unable to lift a block if the block's weight is greater than the robot's strength. This means that in order to lift some heavy blocks the robots must cooperate.

2.3 Height

Another simple extension is to give each of the robots a height such that they are unable to stack block above their height. For example a robot with a height of 1 would be unable to stack any blocks on top of other blocks.

2.4 Agent Cooperation

As mentioned above in order to lift heavy blocks the agents must cooperate. The creates extra complexity for the problem in order to get the robots to cooperate effectively.

2.5 Dynamic Goals

Instead of the user specifying the exact sequence of blocks they would like it is possible to specify a set of rules that make a desirable outcome such as minimising the number of stacks or putting all the heavy blocks on the bottom.

3 Programs

The results of this project consist of two programs which implement an extended blocks world in two different ways. The frst program implement the domain with a original implementation of the situation calculus with successful use of simultaneous actions. The second program is built on ConGolog and generates plans with dynamic goals but without true simultaneous actions. Unfortunately both programs suffer from poor performance when given any scenario with a moderate number of entities.

3.1 sitcalc

Sitcalc has extended blocks world to deal with:

- ÂĹ Multiple agents.
- ÂĹ Block weight and robot strength.
- ÂĹ Robot height.
- ÂĹ True simultaneous actions.
- ÂĹ Agent cooperation.
- ÂĹ Simply dynamic goals.

In order to implement simultaneous, cooperative actions in stead of the standard do(A,S) style of situations sitcalc uses a list of actions; do([A,B,...],S) combined with a set of rules (the simultaneous_actions predicates) which define the combinations of actions that are not permitted. This, along with the standard situation calculus, allows the agents to simultaneous perform cooperative actions.

The new situation calculus has also been implemented independent of the domain.

3.2 ConGolog

The ConGolog program implements similar features to the sitcalc program, albeit slightly more polished, but is using Ryan Kelly's implementation of ConGolog for the situation calculus. This program implements:

- ÂĹ Multiple agents with interleaved actions.
- ÂĹ Block weight and robot strength.
- ÂĹ Robot height.
- ÂĹ Complicated dynamic goals.

The programs goals are generated during run time in order to be able to create a reasonable goal for any initial state. A number of arbitrary rules are used to generate possible goals:

- ÂĹ All blocks must be part of the solution.
- ÂĹ Minimise the number of stacks.
- ÂĹ Place the heavier blocks at the bottom of the stacks.

The last component of goal generation is minimising the time required to access each type of block. Each block is has a type (such as productA or productB) and the goal with the lowest average access time for each type of block is chosen as the final goal. This could be useful in a warehouse simulation where quick access to types of good is required.

4 Further Work

There is obviously a huge amount of scope for the blocks world domain but the main things that could benefit these program involve the execution times for complicated situations (which are currently woeful for both programs). Fortunately there is a large amount of literature dealing with this issue.

There are many other interesting extensions that could be implemented for these programs:

- ÂĹ Heuristic action choice.
- ÂĹ An ongoing warehouse problem where new blocks are added to the system and certain blocks must be removed from the system at times (these would be exogenous actions).
- ÂĹ More complicated handling of robot-block interactions (such as some robots being able to handle certain types of blocks or robot movement around blocks).
- ÂĹ Knowledge about block attributes and locations (imagine a warehouse with multiple rooms and robots have limited knowledge about rooms they aren't in).