CS 686 Project Deliverable #2:

Heuristic Methods for Planning Problems

By: Steve Engels, 94006290

For: Dr. Brendan Frey

Date Due: Thursday, Dec 9, 1999

CS 686 Project Deliverable #2

By: Steve Engels, ID 94006290

Contents

1 Introduction

Planning problems are a common and practical application of artificial intelligence. They occur in programs related to scheduling, navigation, construction and other applications that seek to achieve a goal state from a start state, given a set of actions. To solve these problems, software planners will search the space of possible actions, until they find a series of actions that collectively make the transition from the start to the goal. However, searching for a plan simply with depth- or breadth- first searches in not a very effective approach to the problem. For large action spaces, simply searching to find the goal can be inefficient and computationally expensive, and may not even find a plan at all.

The focus of this project is to develop a new heuristic that would help solve planning problems more effectively. If successful, this heuristic would give a better indication of the actions that should be explored first in order to reach the goal. The path from the start state to the goal state can thus be found more easily by applying this heuristic at each successive stage in the path.

2 Problem Description

The specific problem being addressed in this project is whether the new heuristic being developed will be effective in resolving a planning problem in the Blocks World domain. The success of this heuristic is determined by comparing two planning algorithms: one using the heuristic, and one not using the heuristic (breadth-first search). The technique that solves the most planning problems and solves them in the least number of steps will be considered the more effective technique. The planning algorithms will also be applied in three different directions: from initial state to goal state, from goal state to start state, and from a middle state in both directions, to see if the direction of search would favour one technique over the other.

The result of this investigation will be an illustration of how successful each heuristic is in the Blocks World domain, and which direction each heuristic should be applied in to achieve optimal performance. The reason why this is important is that if the new heuristic is successful, it will be a breakthrough in the area of planning problem research in that it will provide insights into how to represent and approach planning problems in the future. Even if the heuristic is not a success, the manner in which is fails will illustrate the shortcomings to be avoided in future attempts.

2.1 Blocks World Problem

The Blocks World domain is the specific type of planning problem being used in this project. This domain consists of a world with a table, upon which rest a certain number of blocks. The blocks can rest on the table or on each other, but multiple blocks can never rest on a single block, nor can one block rest on more than one block. To move from an original configuration of blocks to a goal configuration of blocks, the blocks can be picked up and moved, but only the blocks with nothing on them can be moved, and blocks can only be picked up and put down one at a time. Examples of legal and illegal block configurations are shown below in Figure 1.

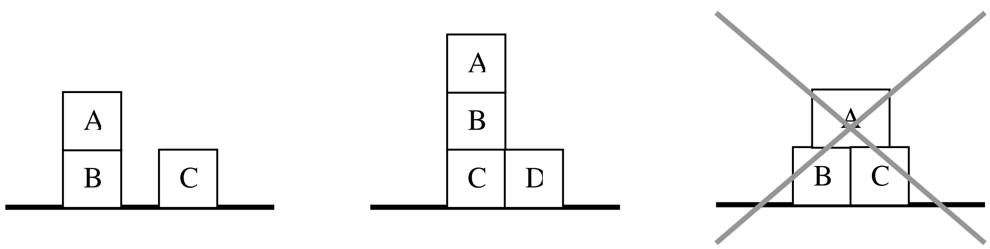

Figure 1: Two legal and one illegal block configuration

This domain has certain advantages in that the possible configurations and actions are very structured and easy to represent. Possible heuristics for this domain are also easily to visualize and implement. However, since any heuristic that is developed is meant to be general and not restricted to this domain, characteristics specific to Blocks World must be set aside when devising the heuristic.

3 Review of Techniques

Traditionally, planning algorithms have been thought of in the STRIPS notation. This notation consists of a set of states and actions, where the states describe certain characteristics of the world, and the actions are defined by a precondition/postcondition pair. Finding the solution to the planning problem with the STRIPS notations involved finding actions whose preconditions matched the initial state and whose postconditions represented the successor state. By performing this action repeatedly on successor states, the algorithm would eventually find a series of actions that would lead from the initial state to the final state, forming a plan.

The problem with this however, is that the set of characteristics that defined a state was never fully complete, in that it would only specify the necessary conditions or characteristics of the state or action. If an action could lead to the goal state, but required a precondition that didn't already exist, the algorithm would have to search backwards to find the actions that would cause those preconditions to take place. This could sometimes involve changing the plan entirely, and would cause unwanted overhead.

Another time and space problem involved with STRIPS is the fact that a search with STRIPS is no better than a breadth or depth-first search. Considering that representing the states and actions of this method can sometimes be cumbersome, it would help to be able to use the most efficient method available. This prompted researchers like Geffner to devise planners with heuristics that would help to navigate the state space as efficiently as possible.

4 Algorithm Description

4.1 Data Representation

Since the configuration of the blocks in this domain can be described in terms of their positions relative to the table and the other blocks, a natural representation of the data is as a set of descriptive characteristics. The characteristics describe the state of each block (clear, on table), each block relative to other blocks (on A, on B, on C) and the "hand" (hand empty, holding A).

If each characteristic is represented as a column in an array, two rows can be used to represent the start and goal configurations of the system. In order to describe a problem to solve, the columns that correspond to desired start or goal characteristics can be set to be either TRUE (1) or FALSE (-1). Any characteristics that are not relevant to the problem to be solved can be left as UNKNOWN (0).

For example, a block problem where the initial state has block B on block A, and must be changed to have block A on block B would be represented in Figure 2 (with the column titles added for clarity). These problem sets were generated randomly by Fahiem Bacchus, from which they were converted into this array representation. Other columns would exist for problems with more than two blocks, but this subset shows how columns relating to A and B would be stored:

	Hand empty	Holding (A)	Holding (B)	Clear (A)	Clear (B)	On Table (A)	On Table (B)	On (A,B)	On (B,A)
Start	[0	-1	-1	-1	0	0	-1	-1	1
Goal	0	-1	-1	0	-1	-1	0	1	-1]

Figure 2: Data representation of start and goal states of plan

The actions that make the transition from state to state are represented in a similar way. The same columns exist for the actions, but the top row represents the required state characteristics that must exist for the action to be legal, and the second row represents the resulting change to the state after the action occurs. An example of an action to pick up block B is shown in Figure 3:

	Hand empty	Holding (A)	Holding (B)	Clear (A)	Clear (B)	On Table (A)	On Table (B)	On (A,B)	On (B,A)
State	[1	-1	-1	0	0	0	0	-1	0
Result	-1	-1	1	0	-1	0	0	-1	-1]

Figure 3: Data representation of input state and result of pickup action

4.2 Heuristic Description

The heuristic to be used is based on that of Blai Bonet and Hector Geffner[1] (referred to after this as the Geffner heuristic). The Geffner heuristic begins by eliminating the delete lists from the problem descriptions (i.e. any instances of -1in the action results). It then compares the start and goal rows in the problem description, and treats each instance where the start and goal characteristics were different as individual subgoals to achieve. In order to calculate the heuristic, smaller, shallower searches are made to achieve each of the individual subgoals, and the results of each of the searches are added together to provide the heuristic value from one state to the next.

Geffner applies this idea by calculating the heuristic to change each characteristic in the goal state of the problem to its opposite value in the start state, a technique known as regression search. Once this is complete, it results in a table of values that represents the cost incurred in changing each characteristic, assuming that it did not match the goal. Then to evaluate the heuristic cost from any state to the goal, the mismatched characteristics in the state are found, and the corresponding entries in the heuristic table are added together to obtain the heuristic value. The attractiveness of one action over another is thus influenced by the heuristic value for each of the possible successor states.

The use of the heuristic for this project is similar, but adaptations are being made to allow it to operate in both the forward and backward directions. Another adaptation is to make the heuristic admissible, by taking the maximum of the heuristic costs instead of the sum. Since the steps needed to simply change one of the mismatched characteristics will always be less than or equal to the steps needed to change all of the mismatched characteristics, the maximum of the heuristic costs will always underestimate the real cost, and will therefore be admissible.

[1] Blai Bonet, Hector Geffner, Planning as Heuristic Search: New Results, *submitted to ECP-99*

4.3 Algorithm Description

Before putting the heuristic into consideration, a breadth-first search of the legal actions is used to explore the state space of block configurations. Once the heuristic is introduced, a greedy search algorithm is used to test the validity of the heuristic. The search then continues either until a path from the start to the goal node is found, or until the search explores a set maximum limit, which prevents the system from searching forever.

A difference between the approach used by Geffner and the approach used in this adaptation is the bi-directional search mentioned above. This means that the initial state in the search is neither the start nor the goal state, but rather an intermediate state on which actions can be applied in either direction. It is uncertain whether this technique would be valid or not, but this also will be illustrated once the performance of this approach has been fully explored.

5 Results

After running experiments on the planning program in its different configurations, the results obtained were very limited. The limiting factor in this case is that as the number of blocks increased, the number of possible states and actions increased exponentially, causing processing failures beyond situations with more than six blocks. The results that could be obtained are illustrated in the following tables:

Blocks	Average No. of Actions Considered with Heuristic	Average No. of Actions Considered without Heuristic
2	70.8	70.8
4	2008	2661.2
6	8528	19555

Table 1 : Results from Forward Search Trials

Blocks	Average No. of Actions Considered with Heuristic	Average No. of Actions Considered without Heuristic
2	70.8	70.8
4	1739.2	2182.4
6	7214.5	14134

Table 2 : Results from Backwards Search Trials

Blocks	Average No. of Actions Considered with Heuristic	Average No. of Actions Considered without Heuristic
2	70.8	70.8
4	2084	2617.6
6	7832	23535

Table 3 : Results from Bi-directional Search Trials

More detailed results are available in Appendix A.

6 Discussion

There are three main patterns that can be seen in the data. The first is that the heuristic has very little impact on the smallest block cases. This is not surprising, since the number of possible actions and the length of the solution cannot be very large for the two-block case.

The second pattern is that as the number of blocks being considered gets higher, the heuristic helps to improve the performance of the search by reducing the number of actions that need to be expanded to find the goal. This is also not surprising, for even though greedy searches have a tendency to get caught up in local minima in certain circumstances, the Blocks World problem would not have many situations like that, and in problems with fewer blocks would typically perform comparably with the breadth-first search.

The final pattern that can be seen is that the performance of the algorithms are ranked according to efficiency in the following way:

1. Backwards search with heuristic
2. Bi-directional search with heuristic
3. Forward search with heuristic
4. Backwards search without heuristic
5. Forward search without heuristic
6. Bi-directional search without heuristic

The fact that the backwards search is the more efficient algorithm is not very surprising, since the algorithm proposed by Geffner worked by underspecifying the goal and performing the search in the backwards direction. Implementing an equivalent heuristic in the forward direction was not easily done, since the operation was not symmetrical, and the performance of the heuristic suffered as a result. However, the unusual behaviour was the bi-directional search, which performed better than the forward search in the heuristic case, but not in the breadth-first case. The main explanation for this is that the bi-directional case is not as efficient as tying down one end of the search to the initial state or goal state. However, when enhanced by the heuristic, it was able to surpass the forward search, probably because of the efficiency of Geffner's heuristic in the backwards direction.

7 Proposed Future Direction

So far, the best use of this new heuristic is practically identical to the method that Geffner first proposed. This is reassuring in that his results can at least be replicated, if not improved upon. In order to improve the search, the next step will be to not only calculate the steps needed to change one differing characteristic, but also to change the other differing characteristics successively after that, until all of them have been explored. This is more computationally intense, and was thus not ready to be integrated into the project, but will be attempted within the near future.

Another proposed enhancement will be to loosen the constraints on preconditions and postconditions, to move towards more flexible plan searches. It is uncertain what form this will take, or whether this will actually have the intended positive effect, but if possible search paths are currently being left out, then by loosening the constraints the heuristic will be able to find the optimal path more easily.

Appendix A: Tables of Simulation Results

Blocks	Iteration	Success?	Actions	Expansions	Alistsize	Aheapsize	Adictsize	Total
2	0	Y	8	0	8	6	14	64
2	1	Y	8	0	8	6	14	64
2	2	Y	8	2	10	6	18	98
2	3	Y	8	0	8	6	14	64
2	4	Y	8	0	8	6	14	64
				0.4	8.4	6	14.8	70.8
4	0	Y	32	0	32	132	164	1024
4	1	Y	32	12	44	170	226	1924
4	2	Y	32	7	39	164	210	1514
4	3	Y	32	32	64	201	297	4064
4	4	Y	32	7	39	180	226	1514
				11.6	43.6	169.4	224.6	2008
6	0	Y	72	16	88	952	1056	7728
6	1	Y	72	20	92	1189	1301	8444
6	2	Y	72	39	111	1105	1255	12282
6	3	Y	72	0	72	810	882	5184
6	4	Y	72	23	95	1109	1227	9002
				19.6	91.6	1033	1144.2	8528

Table 4: Results from Forward Search with Heuristic

Blocks	Iteration	Success?	Actions	Expansions	Alistsize	Aheapsize	Adictsize	Total
2	0	Y	8	0	8	6	14	64
2	1	Y	8	0	8	6	14	64
2	2	Y	8	2	10	6	18	98
2	3	Y	8	0	8	6	14	64
2	4	Y	8	0	8	6	14	64
				0.4	8.4	6	14.8	70.8
4	0	Y	32	4	36	161	201	1292
4	1	Y	32	12	44	170	226	1924
4	2	Y	32	6	38	154	198	1438
4	3	Y	32	32	64	201	297	4064
4	4	Y	32	36	68	280	384	4588
				18	50	193.2	261.2	2661.2
6	0	Y	72	25	97	1083	1205	9384
6	1	Y	72	14	86	949	1049	7382
6	2	N						
6	3	Y	72	109	181	1613	1904	32652
6	4	Y	72	98	170	1318	1591	28802
				61.5	133.5	1240.75	1437.25	19555

Table 5: Results from Forward Search without Heuristic

Blocks	Iteration	Success?	Actions	Expansions	Alistsize	Aheapsize	Adictsize	Total
2	0	Y	8	0	8	6	14	64
2	1	Y	8	0	8	6	14	64
2	2	Y	8	2	10	6	18	98
2	3	Y	8	0	8	6	14	64
2	4	Y	8	0	8	6	14	64
				0.4	8.4	6	14.8	70.8
4	0	Y	32	0	32	132	164	1024
4	1	Y	32	10	42	165	217	1754
4	2	Y	32	4	36	153	193	1292
4	3	Y	32	24	56	196	276	3112
4	4	Y	32	7	39	163	209	1514
				9	41	161.8	211.8	1739.2
6	0	Y	72	10	82	920	1012	6714
6	1	Y	72	14	86	1010	1110	7382
6	2	N						
6	3	Y	72	0	72	810	882	5184
6	4	Y	72	26	98	1138	1262	9578
				12.5	84.5	969.5	1066.5	7214.5

Table 6: Results from Backward Search with Heuristic

Blocks	Iteration	Success?	Actions	Expansions	Alistsize	Aheapsize	Adictsize	Total
2	0	Y	8	0	8	6	14	64
2	1	Y	8	0	8	6	14	64
2	2	Y	8	2	10	6	18	98
2	3	Y	8	0	8	6	14	64
2	4	Y	8	0	8	6	14	64
				0.4	8.4	6	14.8	70.8
4	0	Y	32	0	32	132	164	1024
4	1	Y	32	12	44	168	224	1924
4	2	Y	32	5	37	157	199	1364
4	3	Y	32	36	68	209	313	4588
4	4	Y	32	13	45	181	239	2012
				13.2	45.2	169.4	227.8	2182.4
6	0	Y	72	18	90	1013	1121	8082
6	1	Y	72	26	98	1131	1255	9578
6	2	Y	72	118	190	1231	1542	35982
6	3	Y	72	0	72	810	882	5184
6	4	Y	72	37	109	1190	1336	11844
				39.8	111.8	1075	1227.2	14134

Table 7: Results from Backward Search without Heuristic

Blocks	Iteration	Success?	Actions	Expansions	Alistsize	Aheapsize	Adictsize	Total
2	0	Y	8	0	8	6	14	64
2	1	Y	8	0	8	6	14	64
2	2	Y	8	2	10	6	18	98
2	3	Y	8	0	8	6	14	64
2	4	Y	8	0	8	6	14	64
				0.4	8.4	6	14.8	70.8
4	0	Y	32	0	32	132	164	1024
4	1	Y	32	10	42	172	224	1754
4	2	Y	32	6	38	181	225	1438
4	3	Y	32	33	65	252	350	4192
4	4	Y	32	13	45	194	252	2012
				12.4	44.4	186.2	243	2084
6	0	Y	72	16	88	953	1057	7728
6	1	Y	72	20	92	1208	1320	8444
6	2	N						
6	3	Y	72	0	72	810	882	5184
6	4	Y	72	28	100	1170	1298	9972
				16	88	1035.25	1139.25	7832

Table 8: Results from Bi-directional Search with Heuristic

Blocks	Iteration	Success?	Actions	Expansions	Alistsize	Aheapsize	Adictsize	Total
2	0	Y	8	0	8	6	14	64
2	1	Y	8	0	8	6	14	64
2	2	Y	8	2	10	6	18	98
2	3	Y	8	0	8	6	14	64
2	4	Y	8	0	8	6	14	64
				0.4	8.4	6	14.8	70.8
4	0	Y	32	4	36	161	201	1292
4	1	Y	32	12	44	170	226	1924
4	2	Y	32	6	38	154	198	1438
4	3	Y	32	38	70	213	321	4862
4	4	Y	32	28	60	254	342	3572
				17.6	49.6	190.4	257.6	2617.6
6	0	Y	72	101	173	1377	1651	29828
6	1	Y	72	14	86	949	1049	7382
6	2	N						
6	3	Y	72	96	168	1505	1770	28128
6	4	Y	72	98	170	1318	1591	28802
				77.25	149.25	1287.25	1515.25	23535

Table 9: Result from Bi-directional Search without Heuristic