RIP HW 1

Inder Dhir, Ajmal Kunnummal, Sagar Laud, James Liu, Richard Stauffer

October 5, 2014

1 Problem 1

(a) **Compare the Methods of the Two Planners:**
Blackbox Planner
This planner operated by turning the given problem into a set of boolean satisfiability problems through two mechanisms. The front end is a graphplan technique where instead of state nodes and edges representing possible traversal, we have nodes of actions/facts and edges from facts -¿ actions or from actions -¿ facts affected by said action. Arranged in an alternating fashion: Facts, to possible actions, back to facts. It also uses backward chaining and iterative depth probing to keep from exploring too many extraneous nodes.

VHPOP Planner
VHPOP is a partial-order planner. This means it generates a lists of actions necessary to get to the goal, only constraining their order when absolutely necessary (One has preconditions, an earlier action must synthesize) Based off POCL it eliminates flaws in a partial plan until all preconditions for each action are satisfied. Its given list of features (From the VHPOP website) are as follows:

Can plan with either ground or lifted actions.

Can enforce joint parameter domain constraints when using lifted actions.

Has several different plan ranking heuristics to choose from that can be combined into complex plan ranking functions.

Efficiently implements all common flaw selection strategies, including LCFR and ZLIFO, and many novel ones, and allows flaw selection strategies to be specified using a concise notation.

Several flaw selection strategies can be used simultaneously.

Implements A*, IDA*, and hill climbing search.

(b) **Which Planner was Fastest:**
Both took very little time, as this is a relatively small problem in terms of search space, but the Blackbox planner was still undoubtedly faster at 12 milliseconds versus the 5160 milliseconds of the VHPOP planner.

(c) **Why Might this Planner be Faster?**
Much of A*'s speed relies on finding a good heuristic. Since VHPOP uses several generalizable heuristics it might have been able to save time if given a problem specific heuristic.

2 Problem 2

(a) **Solutions for Sokoban Problems:**
Solutions given by FF Planner

2.1
step 0: MOVE SOUTH BOT B4 B3
1: MOVE EAST BOT B3 C3
2: MOVE EAST BOT C3 D3
3: MOVE SOUTH BOT D3 D2
4: PUSH WEST BOT BLOCK1 D2 C2 B2
5: MOVE NORTH BOT C2 C3
6: MOVE WEST BOT C3 B3
7: MOVE WEST BOT B3 A3
8: MOVE SOUTH BOT A3 A2
9: MOVE SOUTH BOT A2 A1
10: MOVE EAST BOT A1 B1
11: PUSH NORTH BOT BLOCK1 B1 B2 B3
12: PUSH NORTH BOT BLOCK1 B2 B3 B4
13: PUSH NORTH BOT BLOCK1 B3 B4 B5

2.2
step 0: MOVE NORTH BOT A5 A6
1: MOVE EAST BOT A6 B6
2: MOVE EAST BOT B6 C6
3: PUSH SOUTH BOT BLOCKB C6 C5 C4
4: MOVE NORTH BOT C5 C6
5: MOVE WEST BOT C6 B6
6: MOVE WEST BOT B6 A6
7: MOVE SOUTH BOT A6 A5
8: PUSH EAST BOT BLOCKA A5 B5 C5
9: MOVE SOUTH BOT B5 B4

10: PUSH EAST BOT BLOCKB B4 C4 D4
11: MOVE WEST BOT C4 B4
12: MOVE NORTH BOT B4 B5
13: MOVE NORTH BOT B5 B6
14: MOVE EAST BOT B6 C6
15: PUSH SOUTH BOT BLOCKA C6 C5 C4
16: MOVE EAST BOT C5 D5
17: PUSH SOUTH BOT BLOCKB D5 D4 D3
18: PUSH SOUTH BOT BLOCKB D4 D3 D2
19: PUSH SOUTH BOT BLOCKB D3 D2 D1
20: MOVE NORTH BOT D2 D3
21: MOVE WEST BOT D3 C3
22: PUSH NORTH BOT BLOCKA C3 C4 C5
23: MOVE EAST BOT C4 D4
24: MOVE NORTH BOT D4 D5
25: PUSH WEST BOT BLOCKA D5 C5 B5
26: MOVE SOUTH BOT C5 C4
27: MOVE WEST BOT C4 B4
28: PUSH NORTH BOT BLOCKA B4 B5 B6
29: MOVE EAST BOT B5 C5
30: MOVE NORTH BOT C5 C6
31: PUSH WEST BOT BLOCKA C6 B6 A6

2.3
step 0: MOVE EAST BOT B3 C3
1: MOVE SOUTH BOT C3 C2
2: MOVE SOUTH BOT C2 C1
3: MOVE EAST BOT C1 D1
4: MOVE EAST BOT D1 E1
5: MOVE NORTH BOT E1 E2
6: MOVE NORTH BOT E2 E3
7: PUSH WEST BOT BLOCK3 E3 D3 C3
8: PUSH WEST BOT BLOCK3 D3 C3 B3
9: MOVE EAST BOT C3 D3
10: MOVE EAST BOT D3 E3
11: MOVE SOUTH BOT E3 E2
12: MOVE SOUTH BOT E2 E1
13: MOVE EAST BOT E1 F1
14: MOVE EAST BOT F1 G1
15: MOVE NORTH BOT G1 G2
16: MOVE NORTH BOT G2 G3
17: MOVE NORTH BOT G3 G4
18: MOVE NORTH BOT G4 G5
19: MOVE NORTH BOT G5 G6
20: MOVE EAST BOT G6 H6

21: MOVE EAST BOT H6 I6
22: MOVE SOUTH BOT I6 I5
23: MOVE SOUTH BOT I5 I4
24: MOVE SOUTH BOT I4 I3
25: PUSH WEST BOT BLOCK1 I3 H3 G3
26: MOVE EAST BOT H3 I3
27: MOVE NORTH BOT I3 I4
28: MOVE NORTH BOT I4 I5
29: MOVE NORTH BOT I5 I6
30: MOVE WEST BOT I6 H6
31: MOVE WEST BOT H6 G6
32: MOVE SOUTH BOT G6 G5
33: MOVE SOUTH BOT G5 G4
34: PUSH SOUTH BOT BLOCK1 G4 G3 G2
35: PUSH WEST BOT BLOCK2 G3 F3 E3
36: PUSH WEST BOT BLOCK2 F3 E3 D3
37: MOVE SOUTH BOT E3 E2
38: MOVE SOUTH BOT E2 E1
39: MOVE EAST BOT E1 F1
40: MOVE EAST BOT F1 G1
41: PUSH NORTH BOT BLOCK1 G1 G2 G3
42: PUSH NORTH BOT BLOCK1 G2 G3 G4
43: PUSH NORTH BOT BLOCK1 G3 G4 G5
44: PUSH NORTH BOT BLOCK1 G4 G5 G6
45: MOVE SOUTH BOT G5 G4
46: MOVE SOUTH BOT G4 G3
47: MOVE WEST BOT G3 F3
48: MOVE WEST BOT F3 E3
49: MOVE SOUTH BOT E3 E2
50: MOVE SOUTH BOT E2 E1
51: MOVE WEST BOT E1 D1
52: MOVE WEST BOT D1 C1
53: MOVE NORTH BOT C1 C2
54: MOVE NORTH BOT C2 C3
55: PUSH EAST BOT BLOCK2 C3 D3 E3
56: PUSH EAST BOT BLOCK2 D3 E3 F3
57: PUSH EAST BOT BLOCK2 E3 F3 G3
58: MOVE WEST BOT F3 E3
59: MOVE SOUTH BOT E3 E2
60: MOVE SOUTH BOT E2 E1
61: MOVE EAST BOT E1 F1
62: MOVE EAST BOT F1 G1
63: MOVE NORTH BOT G1 G2
64: PUSH NORTH BOT BLOCK2 G2 G3 G4
65: PUSH NORTH BOT BLOCK2 G3 G4 G5
66: MOVE SOUTH BOT G4 G3

67: MOVE WEST BOT G3 F3
68: MOVE WEST BOT F3 E3
69: MOVE WEST BOT E3 D3
70: MOVE WEST BOT D3 C3
71: MOVE SOUTH BOT C3 C2
72: MOVE SOUTH BOT C2 C1
73: MOVE WEST BOT C1 B1
74: MOVE WEST BOT B1 A1
75: MOVE NORTH BOT A1 A2
76: MOVE NORTH BOT A2 A3
77: PUSH EAST BOT BLOCK3 A3 B3 C3
78: PUSH EAST BOT BLOCK3 B3 C3 D3
79: PUSH EAST BOT BLOCK3 C3 D3 E3
80: PUSH EAST BOT BLOCK3 D3 E3 F3
81: PUSH EAST BOT BLOCK3 E3 F3 G3
82: MOVE WEST BOT F3 E3
83: MOVE SOUTH BOT E3 E2
84: MOVE SOUTH BOT E2 E1
85: MOVE EAST BOT E1 F1
86: MOVE EAST BOT F1 G1
87: MOVE NORTH BOT G1 G2
88: PUSH NORTH BOT BLOCK3 G2 G3 G4

(b) **Compare two Planners:**
The FF planner always rounded down to 0.0 seconds for these problems but blackbox took .05 seconds, 12 minutes, and would not solve the 3rd one without manually allowing for larger step size solutions. FF might have performed better due to a breadth first search approach to enforced hill climbing. They both used variations of graphplan so it is safe to assume that blackbox's issues arose with its other half: the boolean constraint satisfiability solver. This is a rather large search space to be defining boolean constraints between objects and that might have been its hang up, it also searches all plans of each length before finding the correct one whereas depth first search has a (small) chance of finding a correct path in its first try.

(c) **Challenges of Expressing in PDDL:**
Semantic (or rather; logical/numerical based) planning require carefully spelled out relationships between items. In a graph we might have a relationship where if the x coordinate of one item is the same as the x coordinate of another, they are on the same column. Meanwhile geometric constraints are much trickier to define. For instance, parallelism: To do this we have to express each line in formula and determine the slope. A much more involved process.

PDDL (more importantly in the domain file) contains actions with a precondition and an effect. If we're manipulating geometric objects in a "real" world it quickly becomes difficult to cover all the factors changed in the state. After the action, are all previous lines that were parallel still parallel? Are we occupying the

same space as another object? Are triangles still congruent to one another?

(d) **Describe a Problem to be used in Semantic Planing:**
Say we had a word with a lot of propositions.$P \rightarrow Q, Q \rightarrow R, \bar{Q} \rightarrow S$ A Forward chaining approach to determine if we can get from X→Y would be faster than considering the domain in a physical space and figuring out connections therein.

3 Problem 3

(a) **Solutions for Sokoban Problems:**

(b) **Compare our Planner vs Previous Planners:**

(c) **Proof of our Planner's Completeness:**

(d) **How did we Speed up our Planner?:**

4 Problem 4

(a) **Solutions for Large Hanoi Problems:**
Solutions given by FF Planner
Since the solutions for tower of hanoi problems are $2^n - 1$ length, where n is the number of discs, it didn't seem prudent to include a text form in this document. Please instead refer to the problem 4 folder of our code, it contains a hanoi10output.txt and a hanoi12output.txt with the solutions enclosed.

(b) **Notes on the Structure of the Plan:**
The Tower of Hanoi plan is actually rather simple. And it is very very repetitive. For instance: imagine a tower n high in position 3 and a goal of a tower n high in position 1. Then, logically due to the rule of only smaller numbers on larger numbers. In order to move disc n to 1, 1 has to be empty and there can't be any discs on top of disc n. Therefore there is a stack of n-1 in position 2. So there exists some number of moves to stack n-1 in position 2, then a move of disc n from position 3 to 1, followed by the the same number again of moves to get n-1 from position 2 onto disc n.

In short if we consider our position numbers flexible we repeat our pattern constantly. Even with n=5. We move 4 discs onto position 2, which requires moving 3 discs onto position 1, which requires moving 2 discs onto position 2, and moving 1 disc onto position 3. Using these precondition states for actions we could then repeat our as-of-yet partially discovered plan and swap in the correct position locations.

(c) **General Planning Strategy using Problem Structure:**
As already explained there is a lot of repetition in Tower of Hanoi. If we saved our states in sort of a dynamic programming solution we could save a lot of time by comparing previous found solutions. So we save our states and our minimum cost paths for all explored states in a plan. But, especially in this instance, we save them in a generalizable manner that we can plug in variable names at the end. So we might consider the state 001 and 010 to be the same state with the same sequence of actions to get to it. However in this sequence of actions the position names might be left blank and we can fill those in as needed in our plan.

We do need to be careful to consider the relative pattern compared to the start space, because in that instance we cannot consider the moves to get from all discs on p3 to p1 (many moves)as the same as getting all discs on p3 onto p3 (0 moves). Anyways, we would take our goal state, consider the n-1 disc problem and assemble our solution as Solve(N)=Solve(n-1) + move disc n + Solve(n-1).

This should be complete because it returns a solution unless one of the subproblems is impossible. And if we can't complete a precondition for some necessary action in the final solution, then there exists no solution.

5 Problem 5

(a) **HTN Planning Problem for Hanoi:**

(b) **Describe Encoded Domain Knowledge:**

(c) **Solve HTN for 3 12 discs:**

(d) **Compare with non HTN Planner:**

(e) **Observations:**