Matthew Ragoza
mtr22@pitt.edu

Assignment 8

ISSP 2160
November 4, 2021

Problem 1. Block world planning with STRIPS

a. Let us define the following two STRIPS operators:

- **put-on**(x, y)
  - preconditions: $Clear(x), Clear(y)$
  - add list: $On(x, y)$
  - remove list: $On(x, z), Clear(y)$
- **put-table**(x)
  - preconditions: $Clear(x)$
  - add list: $On(x, Table)$
  - remove list: $On(x, y)$

b. State after applying **put-table**(B) at initial state:

$$On(A, C), On(C, Table), On(D, Table), Clear(B), Clear(D), On(B, Table)$$

c. New goal from applying **put-on**(C, B) before goal state:

$$On(C, z), On(B, A), On(A, D), Clear(B), Clear(C)$$

d. The uninformed search algorithm I would use for the block world problem is iterative deepening because it is optimal, complete, and has linear space complexity in the depth of the solution. The block world problem can have cyclic and non-cyclic state repeats, so depth first search has the potential to enter infinite loops. Depth first search could also return suboptimal solutions if it happened to explore a branch on a suboptimal solution path before one that leads to the optimal solution. Breadth first search explores the search space level by level, so it always finds the optimal solution, but it has exponential space complexity. Therefore, iterative deepening is the most efficient optimal and complete uninformed search algorithm for the problem.

e. The search tree and solution path that are found by evaluation function-driven search using the remaining sub-goals heuristic can be seen in Figure 1. The algorithm finds the optimal solution, which involves making four moves from the initial state. Evaluation function-driven search is optimal in general when the heuristic is admissible. In this particular block world problem, the remaining sub-goals heuristic is admissible because it never overestimates that number of remaining moves.

f. The remaining sub-goals heuristic would not necessarily be admissible on other planning problems. Specifically, it would not be admissible in a planning problem where more than one sub-goal could be satisfied in a single move, because in that case the heuristic would overestimate the number of moves needed to satisfy those sub-goals. This could lead to the search algorithm returning sub-optimal solutions.

Matthew Ragoza
mtr22@pitt.edu

Assignment 8

ISSP 2160
November 4, 2021

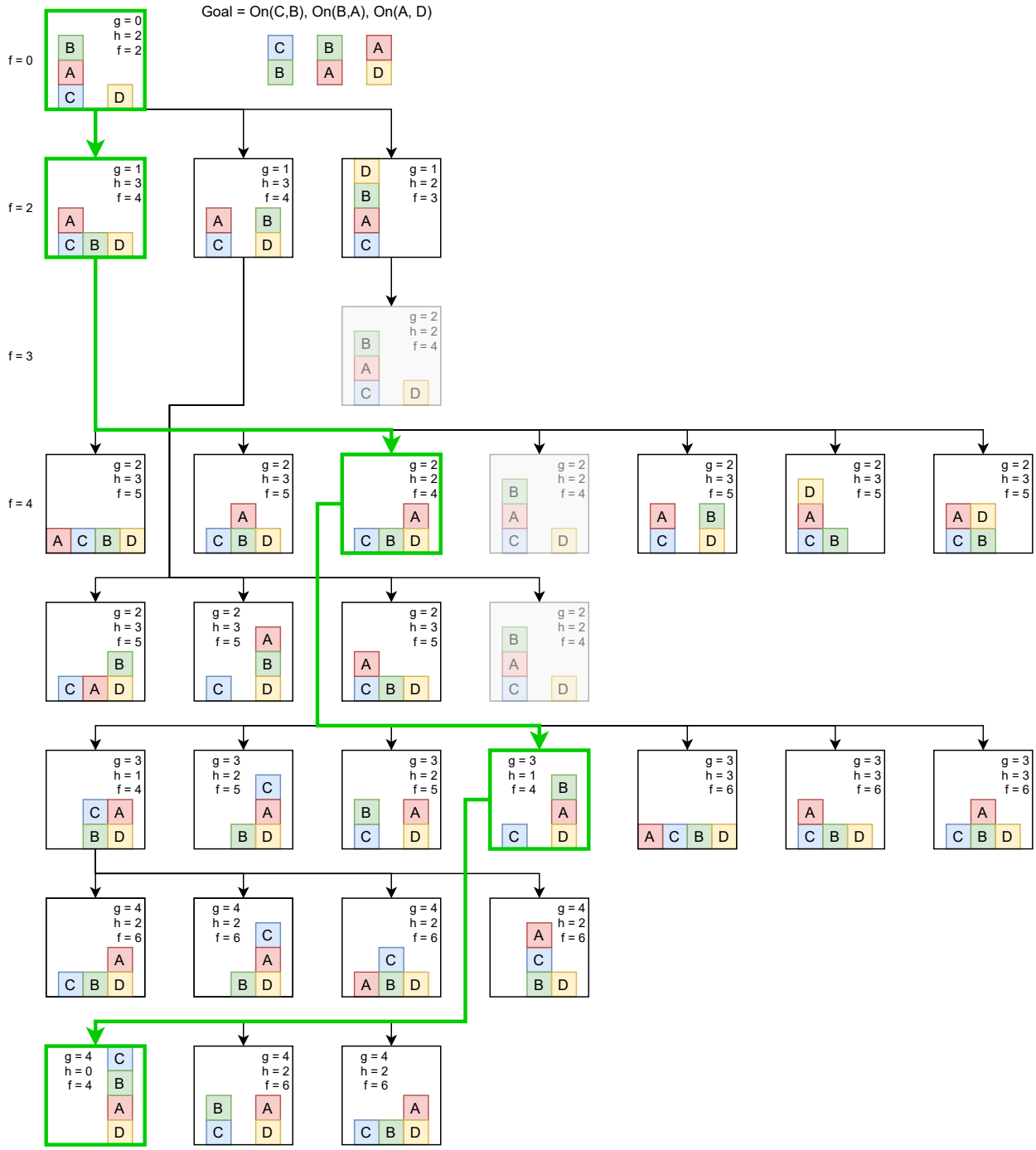

Figure 1: The search tree generated by evaluation function-driven search on the block world problem using the number of remaining sub-goals heuristic. The order of the rows is the order that the nodes were expanded. The solution path is highlighted in green. Grayed-out nodes were visited but not expanded because they contain repeated states. Each node is labelled with the depth (g), heuristic value (h), and evaluation function value (f = g+h).

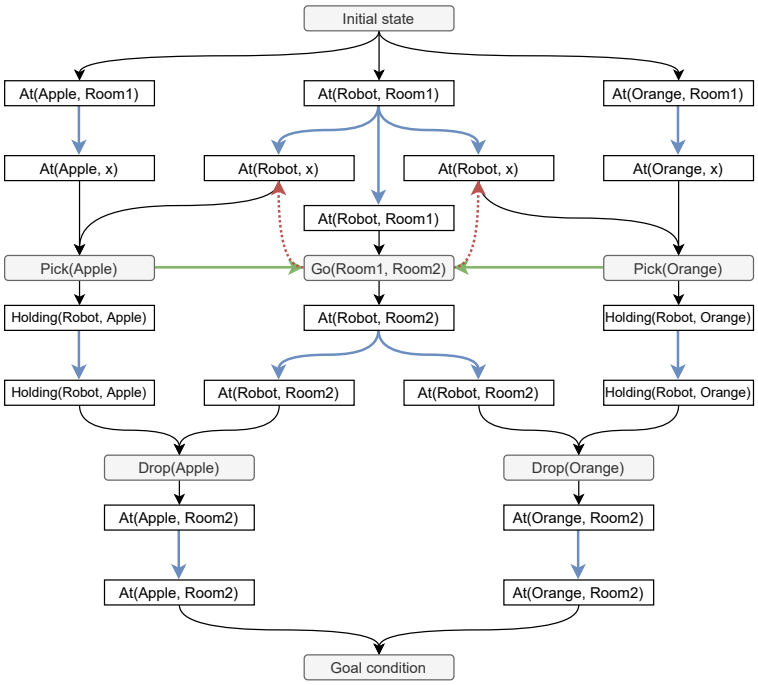

Figure 2: The complete partial-order plan for the robot planning problem. Conditions are represented as white rectangles. Actions are shown as gray rounded rectangles. Blue edges are causal links between action effects and preconditions of other actions that they satisfy. Green edges are ordering links between actions that resolve threats. Dotted red edges represent threats between actions and preconditions of other actions that were resolved by ordering.

Problem 2. Robot partial-order planning

a. The partial order plan for the robot is depicted in Figure 2. Two threats were resolved by adding ordering links. When the robot moves from room 1 to room 2, this is a threat to the precondition that the robot must be in room 1 in order to pick up the apple. Therefore, an ordering link was added so that the robot must pick up the apple before moving from room 1 to room 2. Likewise, moving from room 1 to room 2 is a threat to the precondition that the robot must be in room 1 to pick up the orange. Thus, another ordering link was added so that the robot picks up the orange before it moves to room 2.

b. Each of the following solution plans is consistent with the partial order plan:

(a) $Pick(Apple) \rightarrow Pick(Orange) \rightarrow Go(Room1, Room2) \rightarrow Drop(Apple) \rightarrow Drop(Orange)$

(b) $Pick(Apple) \rightarrow Pick(Orange) \rightarrow Go(Room1, Room2) \rightarrow Drop(Orange) \rightarrow Drop(Apple)$

(c) $Pick(Orange) \rightarrow Pick(Apple) \rightarrow Go(Room1, Room2) \rightarrow Drop(Apple) \rightarrow Drop(Orange)$

(d) $Pick(Orange) \rightarrow Pick(Apple) \rightarrow Go(Room1, Room2) \rightarrow Drop(Orange) \rightarrow Drop(Apple)$