Assignment #4

Planning and Scheduling

COMP 307 Introduction to Artificial Intelligence

Qiangqiang Li(Aaron)

ID: 300422249

Part 1: Classical Planning – Monkey-and-Bananas

1. ***(5 marks) Write the description for the initial and goal states using PDDL.***

Reply:

Init(At(Monkey, A)^At(Box, B)^At(Bananas, C)^Height(Monkey, Low)^ Height(Box, Low)^ Height(Bananas, High))

Goal(Have(Monkey, Bananas))

2. ***(10 marks) Write all the action PDDL schemas (there are 6 actions in total). Each action should include a name, variables, precondition and effect. The precondition of each action should be set in a way that self-connection is avoided, i.e. no action connects any state to itself.***

Reply:

Action(GO(x, y),
  Precond: At(Monkey, x) ∧ ¬ Height(Monkey, High),
  Effect: At(Monkey, y) ∧ ¬ At(Monkey, x))

Action(Push(Box, x, y),
  Precond: At(Monkey, x) ∧ At(Box, x),
  Effect: At(Monkey, y) ∧ At(Box, y) ∧ ¬ At(Monkey, x) ∧ ¬ At(Box, y))

Action(ClimbUp(Box, x),
  Precond: At(Monkey, x) ∧ At(Box, x),
  Effect: Height(Monkey, High) ∧ ¬ Height(Monkey, Low))

Action(ClimbDown(Box, x, y),
  Precond: At(Monkey, x) ∧ At(Box, x) ∧ Height(Monkey, High)
  Effect: Height(Monkey, Low) ∧ ¬ Height(Monkey, High))

Action(Grasp(Monkey, x, h),
  Precond: At(Monkey, x) ∧ At(Box, x) ∧ Height(Monkey, h) ∧ Height(Banana, h)
  Effect: Have(Monkey, Banana))

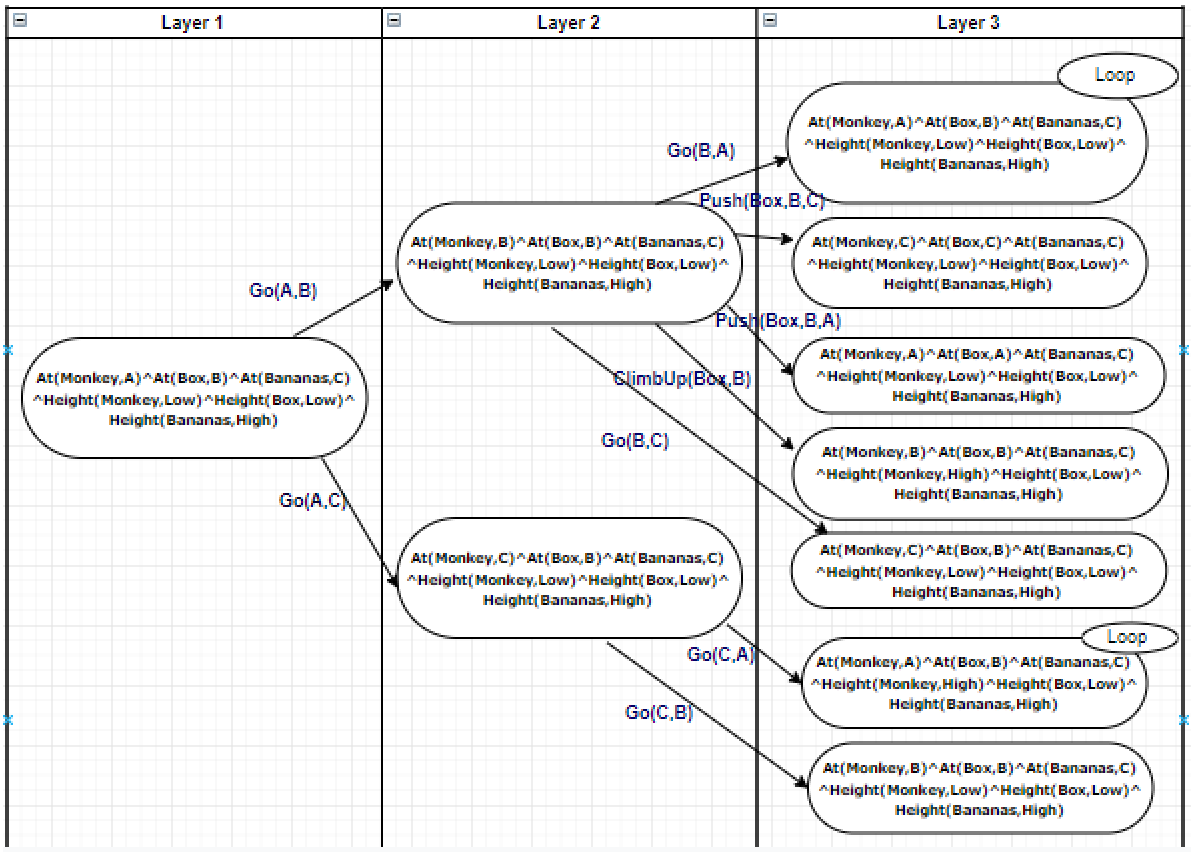
Action(UnGrasp(Banana),
  Precond: Have(Monkey, Banana)
  Effect: ¬ Have(Monkey, Banana)

3. ***(15 marks) A plan to achieve the goal state from the initial state can be found by using forward state-space search. Based on the PDDL in Questions 1 and 2, draw the first three layers of the***

corresponding state-space search graph to demonstrate the search process.

- ***Each node (state) in the graph is represented by a conjunction of fluents.***
- ***Each edge is associated with an action.***
- ***Each leaf node is either a goal state or a loop.***

Reply:



4. ***(5 marks) Write a plan to achieve the goal state from the initial state.***

Reply:

Initial state:

At(Monkey,A) ∧ At(Box,B) ∧ At(Bananas,C) ∧ Height(Monkey,Low) ∧ Height(Box,Low) ∧ Height(Banana, High)

Action1：GO (A, B))

State1:

At(Monkey,B) ∧ At(Box,B) ∧ At(Bananas,C) ∧ Height(Monkey,Low) ∧ Height(Box,Low) ∧ Height(Banana, High)

Action2：Push (Box, B, C)

State2:

At(Monkey,C) ∧ At(Box,C) ∧ At(Bananas,C) ∧ Height(Monkey,Low) ∧ Height(Box,Low) ∧ Height(Banana, High)

Action3： ClimbUp (Box, x))
State3:
At(Monkey,C)∧At(Box,C)∧At(Bananas,C)∧Height(Monkey,High)∧Height(Box,Low)∧Height(Banana, High)

Action4:Grasp (Monkey, C, High)
State4 (goal state):
At(Monkey,C)∧At(Box,C)∧At(Bananas,C)∧Height(Monkey,High)∧Height(Box,Low)∧Height(Banana, High)∧Have(Banana)

Part 2: Job Shop Scheduling

1. Given a schedule whose action sequence is as follows: Process(O11, M1, t1) → Process(O21, M2, t2) → Process(O31, M1, t3) → Process(O12, M2, t4) → Process(O22, M1, t5) → Process(O32, M2, t6). Since the sequence is sorted in the increasing order of starting time, we know that t1 ≤ t2 ≤ t3 ≤ t4 ≤ t5 ≤ t6. Calculate the earliest starting time (t1 to t6) of each action:

Reply:

According to this problem, the action sequence is as follows: Process(O11, M1, t1) → Process(O21, M2, t2) → Process(O31,M1,t3) → Process(O12,M2,t4) → Process(O22,M1,t5) → Process(O32,M2,t6). Since the sequence is sorted in the increasing order of starting time, we can draw the gantt chart.

Initial:
Earliest idle time: M1=0, M2=0
Earliest ready time:
O11=0,O12=∞
O21=10,O22=∞
O31=20,O32=∞

- ⇨ **t1=max(0,0)=0**

Process(O11;M1; 0):
Earliest idle time: M1=50, M2=0
Earliest ready time:
O12=50
O21=10,O22=∞
O31=20,O32=∞

- ⇨ **t2= max(10,0)=10**

Process(O21;M2; 10):
Earliest idle time: M1=50, M2=40
Earliest ready time:
O12=50

O22=40
O31=20,O32= ∞

⇨ **t3=max(20,50)=50**

Process(O31;M1; 50):
Earliest idle time: M1=90, M2=40
Earliest ready time:
O12=50
O22=40
O32=90

⇨ **t4=max(50,40)=50**

Process(O12;M2; 50):
Earliest idle time: M1=90, M2=40
Earliest ready time:
O22=50
O32=90

⇨ **t5=max(50,90)=90**

Process(O22;M1; 90):
Earliest idle time: M1=125, M2=75
Earliest ready time:
O32=90

⇨ **t6=max(90,75)=90**

Process(O32;M2; 90):
Earliest idle time: M1=125, M2=110

Hint: the earliest starting time of an action is the later time between the earliest ready time of the operation and the earliest idle time of the machine, **t1=0, t2= 10, t3=50, t4=50, t5=90, t6=90**

2. For the solution given in Question 1, find the completion time of each job, which is the finishing time of its last operation. Then, calculate the makespan of the solution, which is defined as the maximum completion time of all the jobs..

Reply:

The completion time of J1 = t4+ ProcTime(O12) = 50 + 25 =75
The completion time of J2 = t5+ ProcTime(O22) = 90 + 35 =125
The completion time of J3 = t6+ ProcTime(O32) = 90 + 20 =110
So, the makespan of the solution is 125.

3. Write the state from step 1 to step 3, and the final solution when applying the Shortest Processing time (SPT) dispatching rule to the problem. At each step, the representation of a state is composed

of (1) a partial solution, (2) the earliest idle time of each machine and (3) the earliest ready time of each unprocessed operation. The initial state (step 0) is given below for your reference.

Reply:

Step 1:
Partial solution: (Process(O11,M1, 0))
earliestIdleTime(M1) = 50, earliestIdleTime(M2) = 0
earliestReadyTime(O12) = 50
earliestReadyTime(O21) = 10, earliestReadyTime(O22) = ∞
earliestReadyTime(O31) = 20, earliestReadyTime(O32) = ∞

Step 2:
Partial solution: (Process(O11,M1, 0)→Process(O21,M2, 10))
earliestIdleTime(M1) = 50, earliestIdleTime(M2) = 40
earliestReadyTime(O12) = 50
earliestReadyTime(O22) = 40
earliestReadyTime(O31) = 20, earliestReadyTime(O32) = ∞

Step 3:
Partial solution: (Process(O11,M1, 0)→Process(O21,M2,10) →Process(O12,M2,50))
earliestIdleTime(M1) = 50, earliestIdleTime(M2) = 75
earliestReadyTime(O22) = 40
earliestReadyTime(O31) = 20, earliestReadyTime(O32) = ∞

Final solution:
Process(O11,M1,0)→Process(O21,M2,10)→Process(O12,M2,50)→Process(O22,M1,50)→Process(O31,M1, 85)→Process(O32,M2, 125)

4. For the solution obtained by the SPT rule, calculate the completion time of each job and the makespan. Compare the makespan between this solution with that obtained in Question 1 to find out which solution is better in makespan. Note: the solution in Question 1 is obtained by the First-Come-First-Serve (FCFS) rule.

Reply:According to SPT rule, the makespan of the solution like the below:

The completion time of J1 = 50+ ProcTime(O12) = 50 + 25 =75
The completion time of J2 = 50+ ProcTime(O22) = 50 + 35 =85
The completion time of J3 = 125+ ProcTime(O32) = 125 + 20 =145
So, the makespan of the solution is 145.

According to FCFS rule, the makespan of the solution(in Question 1) like the below:

The completion time of J1 = t4+ ProcTime(O12) = 50 + 25 =75

The completion time of J2 = t5+ ProcTime(O22) = 90 + 35 =125
The completion time of J3 = t6+ ProcTime(O32) = 90 + 20 =110
So, the makespan of the solution is 125.

So, the makespan obtained in Question 1 (using **FCFS rule**) is shorter than obtained by the SPT rule, the solution obtained in Question 1 (using **FCFS rule**) is better.

5. The two compared solutions are obtained by the SPT and FCFS rules, respectively. If one solution is better then the other, does it mean that the rule that generates the better solution is better than the other rule? Why

Reply:

No, it doesn't mean the FCFS rule is always better than the SPT rule. That's because the performance of the rules is problem dependent. Different rules have different performance because of their own pros and cons in varied circumstance. And since SPT and FCFS are both Forward State-Space Search, they both can't obtain the global optimum. Without global optimum, it's hard to determine which one is better or worse. In this case, SPT rule is doing better than FCFS rules in job 2.

Part 3: Vehicle Routing

1. Find a solution by the nearest neighbor heuristic. Write the solution as a set of node sequences starting and ending at the depot node (node 1). It should look like R1 = (1……1), R2 = (1……1), ….

Reply:

R1 = (1,2,3,5,1)
R2 = (1,6,8,4,1)
R3 = (1,7,9,10,1)

2. Calculate the total length (Euclidean distance) for the obtained solution.

Reply:

L(R1)=L(1,2)+L(2,3)+L(3,5)+L(5,1)=1.00+1.00+1.00+2.24=5.24
L(R2)=L(1,6)+L(6,8)+L(8,4)+L(4,1)=1.41+1.41+1.41+3.16=7.39
L(R3)=L(1,7)+L(7,9)+L(9,10)+L(10,1)=2.23+3.16+2.00+5.39=12.78
L(Sum)= L(R1)+L(R2)+L(R3)=5.24+7.39+12.78=25.41

3. (10 marks) When developing GPHH to evolve heuristic for this routing problem, a priority function is represented as a GP tree. We typically use the standard GP initialisation, selection and evolutionary operators. However, the fitness function, terminal and function sets are problem specific.
For each of them, give your design choices of the following aspects, and briefly justify your choices.

- **Terminal set.**

- **Function set.**
- **Fitness function。**

Reply:

Terminal set： Capacity of truck,
The demand of each node,
The distance to Depot node,
The distance to the nearest neighbor,
Current number of the truck.

Function set： {+, -, *, /, sqrt, square}

Fitness function: The fitness function should be designed to calculate the total distances. Using these current features, compared the total distance form Depot node to each node, and chooses the least distances as the best route. The less Distances, the better routing.