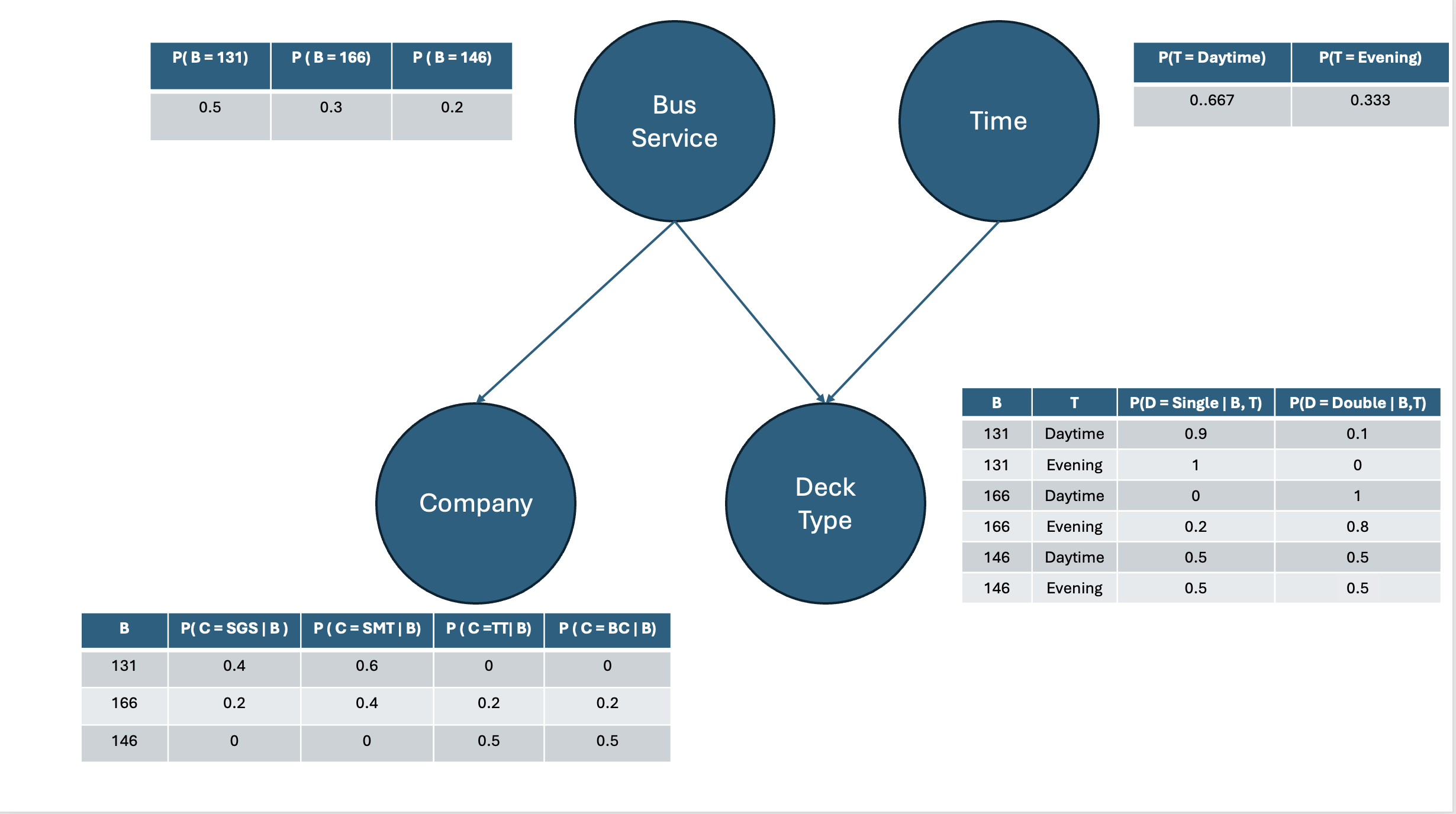
**Question-1**

(a)



(b)

P (D = Single, C = SMT | T = Evening)

= Σ P (D = Single, C =SMT, B| T = Evening)

Since B and T are independent without giving D

= Σ P (D = Single, C = SMT | T = Evening, B) P(B)

Since C and D are independent when given B,

=Σ P (D = Single | T = Evening, B) P (C = SMT | T = Evening, B) P(B)

=P (D = Single | T=Evening, B = 131) P (C = SMT | B =131) P (B =131) +

P (D = Single | T = Evening, B =166) P (C = SMT | B = 166) P(B=166) +

P (D = Single | T = Evening, B =146) P (C = SMT | B =146) P(B=146)

=1\*0.6\*0.5 + 0.2\*0.4\*0.3 + 0.5\*0\*0.2

=0.324

(c)

P (C =SMT | D = Single, T = Daytime)

= P (C = SMT, D = Single | T = Daytime) /P (D =Single | T = Daytime)

=Σ P (C = SMT, D = Single, B | T = Daytime) /Σ P (D = Single, B | T = Daytime)

=Σ P (B) P (C = SMT | B) P (D = Single | Daytime, B)/Σ P (D = Single | B, T = Daytime) P (B)

= (0.5\* 0.6\* 0.9 + 0.3\*0.4\*0 + 0.2\*0\*0.5) /(0.5\*0.9 + 0.3\*0 + 0.2\*0.5)

= (0.27 + 0 + 0) /(0.45 + 0 + 0.1)

= 0.491

(d)

P (D = Double | B = 166)

= Σ P (D = Double, T | B =166)

= Σ P (D = Double | B = 166, T) P(T)

= 0.667\*1 + 0.333\*0.8

= 0.933

**Question-2**

(a)

A screenshot of a computer program

Description automatically generated

The TE does not make sense because the total number of people do not get treated is twice as people who get treated.

But the number of smokers get treated is even larger than smokers do not get treated.

And we all known that smoking can significantly increase the chance of dead.

A screenshot of a computer program

Description automatically generated

(b)

A screenshot of a computer program

Description automatically generated

Which is the same as question(a)

(c)

A screenshot of a computer program

Description automatically generated

(d)

The fundamental difference between Bob and Alan’s Bayesian Network is:

In Bob’s BN, smoking habits will influence the probability of treatment, which means the but in Alan’s BN, smoking and treatment are conditional independent.

Bob’s BN should be more accurate because when we are calculating TE, we use treatment and dead only.

**Question-3**

(a)

Code for X-distance:

A computer screen shot of a program

Description automatically generated

Code for A\* Search:

A screen shot of a computer program

Description automatically generated

Case1:

Hamming distance:

A black background with white text

Description automatically generated

Manhattan distance:

A black background with white text

Description automatically generated

X-distance:

A screenshot of a black background

Description automatically generated

Case 2:

Hamming distance:

A black background with white text

Description automatically generated

Manhattan distance

A black background with white text

Description automatically generated

X-distance:

A black background with white numbers

Description automatically generated

Case 3:

Hamming distance:

A black background with white text

Description automatically generated

Manhattan distance:

A black background with white text

Description automatically generated

X-distance:

A screenshot of a phone

Description automatically generated

(b)

Manhattan Distance: Often the best heuristic due to its balance between informativeness and efficiency.

X-Distance: A good middle ground, better than Hamming but not as effective as Manhattan.

Hamming Distance: The simplest but least effective, leading to higher costs and more node reopens.

**Question-4**

(a)

Since it’s a 5 \* 5 grid, and here are two walls occupy 3 grid and 2 grid respectively.

5 \* 5 – 3 – 2 = 20 states.

(b)

*Q*(*S*,*A*)=*R*(*S*,*A*,*S*′)+*γV*(*S*′) { *R*(*S*,*A*,*S*′) = 1while *S*′= (5,3) . *R*(*S*,*A*,*S*′) = -1while *S*′≠ (5,3)}

(c)

Iteration 0:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 0 |
| 0 |  | 0 |  | 0 |
| 0 |  | 0 |  | 0 |
| 0 |  | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 |

Iteration 1:

Since all values are initialized to 0:

*V*(*s*)=−1+max{0.9⋅0+0.05⋅0+0.05⋅0}=−1

However, for the cell with the reward:

*V*(3,5)=1+max{0.9⋅0+0.05⋅0+0.05⋅0}=1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| -1 | -1 | -1 | -1 | -1 |
| -1 |  | -1 |  | -1 |
| -1 |  | -1 |  | 1 |
| -1 |  | -1 | -1 | -1 |
| -1 | -1 | -1 | -1 | -1 |

Iteration 2:

*V*(2,5)=−1+max{0.9⋅1+0.05⋅(−1)+0.05⋅(−1)} =−0.2

*V*(4,5)=−1+max{0.9⋅1+0.05⋅(−1)+0.05⋅(−1)} =-0.2

For other states:

*V*(*s*)=−1+max{0.9⋅(−1)+0.05⋅(−1)+0.05⋅(−1)}=−2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| -2 | -2 | -2 | -2 | -2 |
| -2 |  | -2 |  | -0.2 |
| -2 |  | -2 |  | 1 |
| -2 |  | -2 | -2 | -0.2 |
| -2 | -2 | -2 | -2 | -2 |

(d)

States is determined by the positions of agent and pizza.

And here are 20 positions for agent and pizza, 20 \* 20 =400 states in all.

(e)

The Bellman Equation accounts for the probabilistic nature of state transitions, which fits the dynamic environment where the agent might not always land in the desired grid.

Since we got 20 \* 20 =400 states according to the question(d), and agent can choose any grid to jump which means he have 20 choices of actions, so here will be 20 \* 20 \* 20 = 8000 entries.