MDL Assignment 2 Part 2 and Part 3 Report

Team Name: Trees.Love

Team Members:

- Pulak Malhotra 2019101050
- Ashwin Rao 2019101049

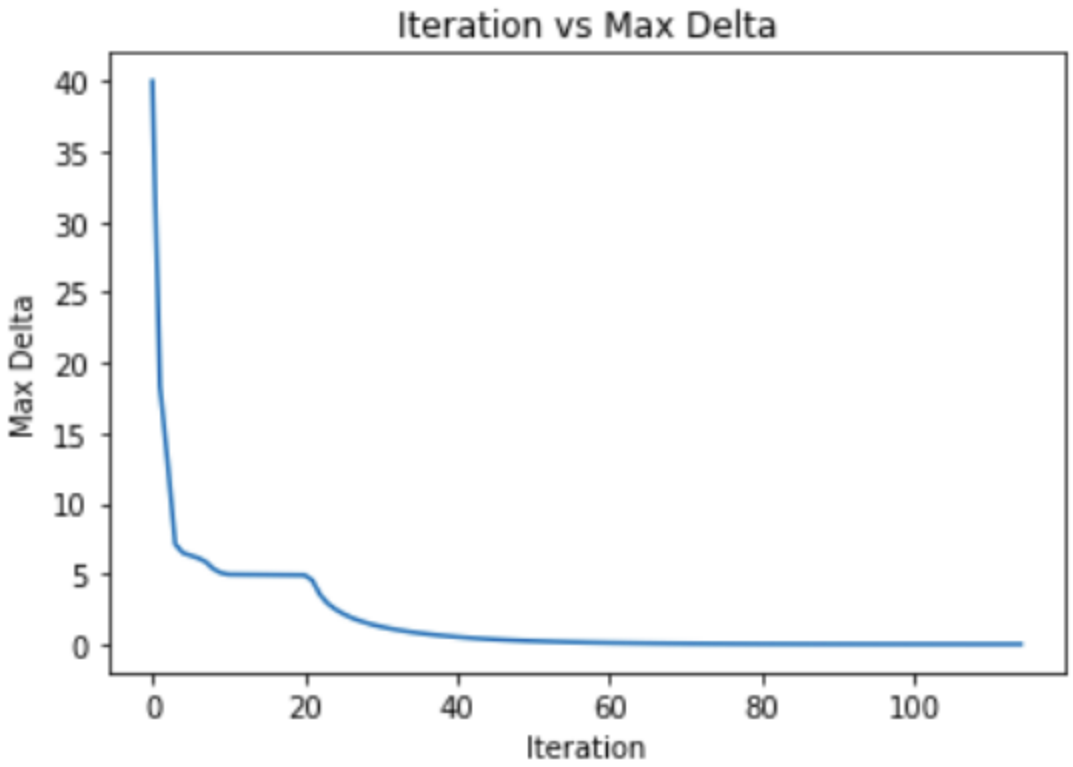
PART 2: VI

TASK 1

Analysis of Trace

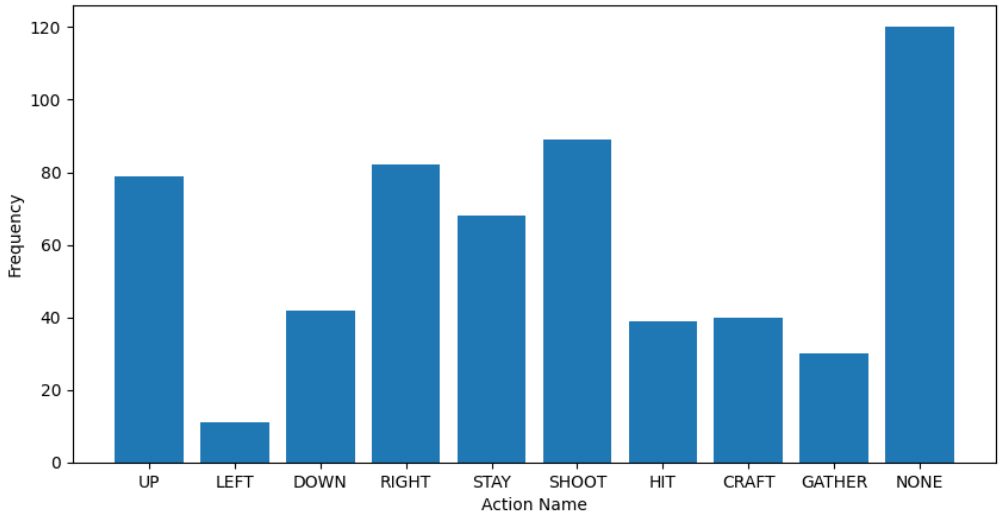
Stats:

- Step cost = -5.0
- VI took 114 iterations to converge.
- Convergence Visualization



Analysis of trace:

- States with the highest value [50] are the terminal states



- Action frequency `{'UP': 79, 'LEFT': 11, 'DOWN': 42, 'RIGHT': 82, 'STAY': 68, 'SHOOT': 89, 'HIT': 39, 'CRAFT': 40, 'GATHER': 30, 'NONE': 120}`

North:

- `(N,2,3,D,25)` have the highest value of `9.424323225891255` with the most favoured action being to go down to `C`
- If `MM` is ready and you have material most likely action is `CRAFT` with a few exceptions. If you don't have material then the most likely action is to `STAY` in the north state. This shows that the model has learned that going to `C` when MM is ready is not good action.
- If `MM` is dormant, then action is `DOWN` or `CRAFT` depending upon the number of arrows already and the health of `MM` . `DOWN` is favoured if you already have 3 arrows or no materials. This shows the model has learnt crafting arrows is useless when you already have arrows.

South:

- `(S,2,3,D,25`) have the highest value of `9.424323225891255`
- If `MM` is ready then action is between `GATHER` and `STAY` depending upon the existing material and the health of MM. This shows that the model has learned that going to `C` when MM is ready is not good action.
- If `MM` is dormant then action is between `UP` in most cases and `GATHER` only in a couple of cases. This shows the willingness to not stay in south if `MM` is dormant since you cant attack from here.

East:

- `(E,2,3,D,25)` have the highest value of `42.99477645858277` . This is the **best** position to be in compared to all other positions.
- If `MM` is ready then the mostly likely action is `HIT` if MM has health `100` else the action is to `SHOOT` him if you have arrow.
- If `MM` is dormant then the most favoured action is almost the same as if `MM` was ready. This shows that in `E` state there is no point of `STAY` or going to `C` since this has best probability of landing hits and even if MM is ready you can do nothing about it.

West:

- `(W,2,3,D,25)` has the highest value of `12.294799337626952` .
- If `MM` is ready then actions is among `SHOOT` and `STAY` Again this shows that IJ does not want to go to `C` when `MM` is ready.
- if `MM` is dormant then action is to shoot if MM is at health `25` else the action is to go `RIGHT` to centre

Center

- `(C,2,3,D,25)` have the highest value of `25.35106780906009` after the terminal states
- If `MM` is ready then most actions are trying to get out of `C` and go `LEFT, UP, DOWN` depending upon current material, health and arrows. This shows `IJ` wants to go out of attack range asap.
- If `MM` is in a dormant state most favored action is to go `RIGHT` to the east but if you don't have arrows and have materials then it is favored to go `UP`

Simulation

I have simulated both of the events two times:

(W, 0, 0, D, 100)

Simulation result 1:

```
Now: (W,0,0,D,100) Actions.RIGHT
Possible outcomes
0.800 (C,0,0,D,100)
0.200 (C,0,0,R,100)
Selected Outcome: 0 Rolled 0.008
(C,0,0,D,100) Actions.RIGHT
Possible outcomes
0.120 (E,0,0,D,100)
0.030 (E,0,0,R,100)
0.680 (E,0,0,D,100)
0.170 (E,0,0,R,100)
Selected Outcome: 2 Rolled 0.280
(E,0,0,D,100) Actions.HIT
Possible outcomes
0.640 (E,0,0,D,100)
0.160 (E,0,0,R,100)
```

```
0.160 (E,0,0,D,50)
0.040 (E,0,0,R,50)
Selected Outcome: 0 Rolled 0.156
(E,0,0,D,100) Actions.HIT
Possible outcomes
0.640 (E,0,0,D,100)
0.160 (E,0,0,R,100)
0.160 (E,0,0,D,50)
0.040 (E,0,0,R,50)
Selected Outcome: 0 Rolled 0.005
(E,0,0,D,100) Actions.HIT
Possible outcomes
0.640 (E,0,0,D,100)
0.160 (E,0,0,R,100)
0.160 (E,0,0,D,50)
0.040 (E,0,0,R,50)
Selected Outcome: 0 Rolled 0.549
(E,0,0,D,100) Actions.HIT
Possible outcomes
0.640 (E,0,0,D,100)
0.160 (E,0,0,R,100)
0.160 (E,0,0,D,50)
0.040 (E,0,0,R,50)
Selected Outcome: 2 Rolled 0.913
(E,0,0,D,50) Actions.HIT
Possible outcomes
0.640 (E,0,0,D,50)
0.160 (E,0,0,R,50)
0.160 (E,0,0,D,0)
0.040 (E,0,0,R,0)
Selected Outcome: 0 Rolled 0.563
(E,0,0,D,50) Actions.HIT
Possible outcomes
0.640 (E,0,0,D,50)
0.160 (E,0,0,R,50)
0.160 (E,0,0,D,0)
0.040 (E,0,0,R,0)
Selected Outcome: 2 Rolled 0.876
(E,0,0,D,0) Actions.NONE
```

Simulation result 2:

```
Now: (W,0,0,D,100) Actions.RIGHT
Possible outcomes
0.800 (C,0,0,D,100)
0.200 (C,0,0,R,100)
Selected Outcome: 0 Rolled 0.635
(C,0,0,D,100) Actions.RIGHT
Possible outcomes
0.120 (E,0,0,D,100)
0.030 (E,0,0,R,100)
```

```
0.680 (E,0,0,D,100)
0.170 (E,0,0,R,100)
Selected Outcome: 2 Rolled 0.750
(E,0,0,D,100) Actions.HIT
Possible outcomes
0.640 (E,0,0,D,100)
0.160 (E,0,0,R,100)
0.160 (E,0,0,D,50)
0.040 (E,0,0,R,50)
Selected Outcome: 0 Rolled 0.332
(E,0,0,D,100) Actions.HIT
Possible outcomes
0.640 (E,0,0,D,100)
0.160 (E,0,0,R,100)
0.160 (E,0,0,D,50)
0.040 (E,0,0,R,50)
Selected Outcome: 1 Rolled 0.719
(E,0,0,R,100) Actions.HIT
Possible outcomes
0.400 (E,0,0,R,100)
0.100 (E,0,0,R,50)
0.500 (E,0,0,D,100)
Selected Outcome: 0 Rolled 0.242
(E,0,0,R,100) Actions.HIT
Possible outcomes
0.400 (E,0,0,R,100)
0.100 (E,0,0,R,50)
0.500 (E,0,0,D,100)
Selected Outcome: 2 Rolled 0.970
(E,0,0,D,100) Actions.HIT
Possible outcomes
0.640 (E,0,0,D,100)
0.160 (E,0,0,R,100)
0.160 (E,0,0,D,50)
0.040 (E,0,0,R,50)
Selected Outcome: 0 Rolled 0.591
(E,0,0,D,100) Actions.HIT
Possible outcomes
0.640 (E,0,0,D,100)
0.160 (E,0,0,R,100)
0.160 (E,0,0,D,50)
0.040 (E,0,0,R,50)
Selected Outcome: 0 Rolled 0.210
(E,0,0,D,100) Actions.HIT
Possible outcomes
0.640 (E,0,0,D,100)
0.160 (E,0,0,R,100)
0.160 (E,0,0,D,50)
0.040 (E,0,0,R,50)
Selected Outcome: 2 Rolled 0.947
(E,0,0,D,50) Actions.HIT
Possible outcomes
0.640 (E,0,0,D,50)
0.160 (E,0,0,R,50)
```

```
0.160 (E,0,0,D,0)
0.040 (E,0,0,R,0)
Selected Outcome: 0 Rolled 0.506
(E,0,0,D,50) Actions.HIT
Possible outcomes
0.640 (E,0,0,D,50)
0.160 (E,0,0,R,50)
0.160 (E,0,0,D,0)
0.040 (E,0,0,R,0)
Selected Outcome: 0 Rolled 0.199
(E,0,0,D,50) Actions.HIT
Possible outcomes
0.640 (E,0,0,D,50)
0.160 (E,0,0,R,50)
0.160 (E,0,0,D,0)
0.040 (E,0,0,R,0)
Selected Outcome: 0 Rolled 0.595
(E,0,0,D,50) Actions.HIT
Possible outcomes
0.640 (E,0,0,D,50)
0.160 (E,0,0,R,50)
0.160 (E,0,0,D,0)
0.040 (E,0,0,R,0)
Selected Outcome: 0 Rolled 0.589
(E,0,0,D,50) Actions.HIT
Possible outcomes
0.640 (E,0,0,D,50)
0.160 (E,0,0,R,50)
0.160 (E,0,0,D,0)
0.040 (E,0,0,R,0)
Selected Outcome: 0 Rolled 0.153
(E,0,0,D,50) Actions.HIT
Possible outcomes
0.640 (E,0,0,D,50)
0.160 (E,0,0,R,50)
0.160 (E,0,0,D,0)
0.040 (E,0,0,R,0)
Selected Outcome: 0 Rolled 0.071
(E,0,0,D,50) Actions.HIT
Possible outcomes
0.640 (E,0,0,D,50)
0.160 (E,0,0,R,50)
0.160 (E,0,0,D,0)
0.040 (E,0,0,R,0)
Selected Outcome: 0 Rolled 0.016
(E,0,0,D,50) Actions.HIT
Possible outcomes
0.640 (E,0,0,D,50)
0.160 (E,0,0,R,50)
0.160 (E,0,0,D,0)
0.040 (E,0,0,R,0)
Selected Outcome: 0 Rolled 0.210
(E,0,0,D,50) Actions.HIT
Possible outcomes
```

```
0.640 (E,0,0,D,50)
0.160 (E,0,0,R,50)
0.160 (E,0,0,D,0)
0.040 (E,0,0,R,0)
Selected Outcome: 3 Rolled 0.990
(E,0,0,R,0) Actions.NONE
```

So we see in both the simulations that the strategy that IJ follows is to go right to reach `E` asap and then just `HIT` with blade since he doesn't have arrows. `IJ` does not bother with crafting or gathering since `C` and `E` are higher value in general than `S` and `N`

(C,2,0,R,100)

Simulation 1:

```
Now: (C,2,0,R,100) Actions.UP
Possible outcomes
0.075 (E,2,0,R,100)
0.425 (N,2,0,R,100)
0.500 (C,2,0,D,100)
Selected Outcome: 2 Rolled 0.736
(C,2,0,D,100) Actions.RIGHT
Possible outcomes
0.120 (E,2,0,D,100)
0.030 (E,2,0,R,100)
0.680 (E,2,0,D,100)
0.170 (E,2,0,R,100)
Selected Outcome: 2 Rolled 0.647
(E,2,0,D,100) Actions.HIT
Possible outcomes
0.640 (E,2,0,D,100)
0.160 (E,2,0,R,100)
0.160 (E,2,0,D,50)
0.040 (E,2,0,R,50)
Selected Outcome: 0 Rolled 0.582
(E,2,0,D,100) Actions.HIT
Possible outcomes
0.640 (E,2,0,D,100)
0.160 (E,2,0,R,100)
0.160 (E,2,0,D,50)
0.040 (E,2,0,R,50)
Selected Outcome: 1 Rolled 0.715
(E,2,0,R,100) Actions.HIT
Possible outcomes
0.400 (E,2,0,R,100)
0.100 (E,2,0,R,50)
0.500 (E,2,0,D,100)
Selected Outcome: 0 Rolled 0.035
```

```
(E,2,0,R,100) Actions.HIT
Possible outcomes
0.400 (E,2,0,R,100)
0.100 (E,2,0,R,50)
0.500 (E,2,0,D,100)
Selected Outcome: 2 Rolled 0.913
(E,2,0,D,100) Actions.HIT
Possible outcomes
0.640 (E,2,0,D,100)
0.160 (E,2,0,R,100)
0.160 (E,2,0,D,50)
0.040 (E,2,0,R,50)
Selected Outcome: 0 Rolled 0.608
(E,2,0,D,100) Actions.HIT
Possible outcomes
0.640 (E,2,0,D,100)
0.160 (E,2,0,R,100)
0.160 (E,2,0,D,50)
0.040 (E,2,0,R,50)
Selected Outcome: 0 Rolled 0.063
(E,2,0,D,100) Actions.HIT
Possible outcomes
0.640 (E,2,0,D,100)
0.160 (E,2,0,R,100)
0.160 (E,2,0,D,50)
0.040 (E,2,0,R,50)
Selected Outcome: 1 Rolled 0.645
(E,2,0,R,100) Actions.HIT
Possible outcomes
0.400 (E,2,0,R,100)
0.100 (E,2,0,R,50)
0.500 (E,2,0,D,100)
Selected Outcome: 2 Rolled 0.732
(E,2,0,D,100) Actions.HIT
Possible outcomes
0.640 (E,2,0,D,100)
0.160 (E,2,0,R,100)
0.160 (E,2,0,D,50)
0.040 (E,2,0,R,50)
Selected Outcome: 0 Rolled 0.223
(E,2,0,D,100) Actions.HIT
Possible outcomes
0.640 (E,2,0,D,100)
0.160 (E,2,0,R,100)
0.160 (E,2,0,D,50)
0.040 (E,2,0,R,50)
Selected Outcome: 0 Rolled 0.551
(E,2,0,D,100) Actions.HIT
Possible outcomes
0.640 (E,2,0,D,100)
0.160 (E,2,0,R,100)
0.160 (E,2,0,D,50)
0.040 (E,2,0,R,50)
Selected Outcome: 2 Rolled 0.813
```

```
(E,2,0,D,50) Actions.HIT
Possible outcomes
0.640 (E,2,0,D,50)
0.160 (E,2,0,R,50)
0.160 (E,2,0,D,0)
0.040 (E,2,0,R,0)
Selected Outcome: 0 Rolled 0.604
(E,2,0,D,50) Actions.HIT
Possible outcomes
0.640 (E,2,0,D,50)
0.160 (E,2,0,R,50)
0.160 (E,2,0,D,0)
0.040 (E,2,0,R,0)
Selected Outcome: 2 Rolled 0.953
(E,2,0,D,0) Actions.NONE
```

simulation 2

```
Now: (C,2,0,R,100) Actions.UP
Possible outcomes
0.075 (E,2,0,R,100)
0.425 (N,2,0,R,100)
0.500 (C,2,0,D,100)
Selected Outcome: 1 Rolled 0.355
(N,2,0,R,100) Actions.CRAFT
Possible outcomes
0.250 (N,2,1,R,100)
0.175 (N,2,2,R,100)
0.075 (N,2,3,R,100)
0.250 (N,2,1,D,100)
0.175 (N,2,2,D,100)
0.075 (N,2,3,D,100)
Selected Outcome: 5 Rolled 0.985
(N,2,3,D,100) Actions.DOWN
Possible outcomes
0.120 (E,2,3,D,100)
0.030 (E,2,3,R,100)
0.680 (C,2,3,D,100)
0.170 (C,2,3,R,100)
Selected Outcome: 1 Rolled 0.136
(E,2,3,R,100) Actions.HIT
Possible outcomes
0.400 (E,2,3,R,100)
0.100 (E,2,3,R,50)
0.500 (E,2,0,D,100)
Selected Outcome: 2 Rolled 0.587
(E,2,0,D,100) Actions.HIT
Possible outcomes
0.640 (E,2,0,D,100)
0.160 (E,2,0,R,100)
0.160 (E,2,0,D,50)
```

```
0.040 (E,2,0,R,50)
Selected Outcome: 0 Rolled 0.202
(E,2,0,D,100) Actions.HIT
Possible outcomes
0.640 (E,2,0,D,100)
0.160 (E,2,0,R,100)
0.160 (E,2,0,D,50)
0.040 (E,2,0,R,50)
Selected Outcome: 0 Rolled 0.552
(E,2,0,D,100) Actions.HIT
Possible outcomes
0.640 (E,2,0,D,100)
0.160 (E,2,0,R,100)
0.160 (E,2,0,D,50)
0.040 (E,2,0,R,50)
Selected Outcome: 0 Rolled 0.082
(E,2,0,D,100) Actions.HIT
Possible outcomes
0.640 (E,2,0,D,100)
0.160 (E,2,0,R,100)
0.160 (E,2,0,D,50)
0.040 (E,2,0,R,50)
Selected Outcome: 0 Rolled 0.287
(E,2,0,D,100) Actions.HIT
Possible outcomes
0.640 (E,2,0,D,100)
0.160 (E,2,0,R,100)
0.160 (E,2,0,D,50)
0.040 (E,2,0,R,50)
Selected Outcome: 3 Rolled 0.985
(E,2,0,R,50) Actions.HIT
Possible outcomes
0.400 (E,2,0,R,50)
0.100 (E,2,0,R,0)
0.500 (E,2,0,D,75)
Selected Outcome: 0 Rolled 0.113
(E,2,0,R,50) Actions.HIT
Possible outcomes
0.400 (E,2,0,R,50)
0.100 (E,2,0,R,0)
0.500 (E,2,0,D,75)
Selected Outcome: 1 Rolled 0.425
(E,2,0,R,0) Actions.NONE
```

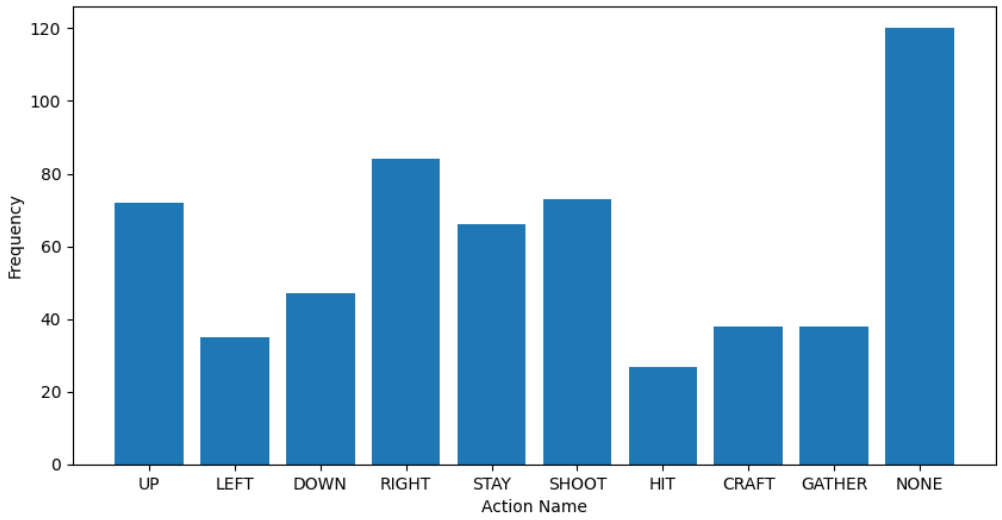
So we see in the simulation IJ tries to go to North for crafting initially to get arrows. Then he goes back down to center and then towards the east. At `100` health he tries to HIT `MM` but then `MM` attacks him and he loses his arrows. This leads to the same result as first simulation and `IJ` does not bother with crafting or gathering anymore and starts hitting `MM` with his blade.

TASK 2

CASE 1

Indiana now on the LEFT action at East Square will go to the West Square.

- Algorithm now converges in **128** iterations
- When `MM` is dormant the algorithm does not favor `LEFT` action in East state at all since `W` is of much lower value than `C`
- When `MM` is ready then the algorithm favours `LEFT` action and `SHOOT` / `HIT` . We noticed that before making this change the algorithm would never take `LEFT` action. This makes sense since `C` state is a worse state than E if `MM` is `R` or `D` but `W` is a more favourable than `E` if MM is `R`

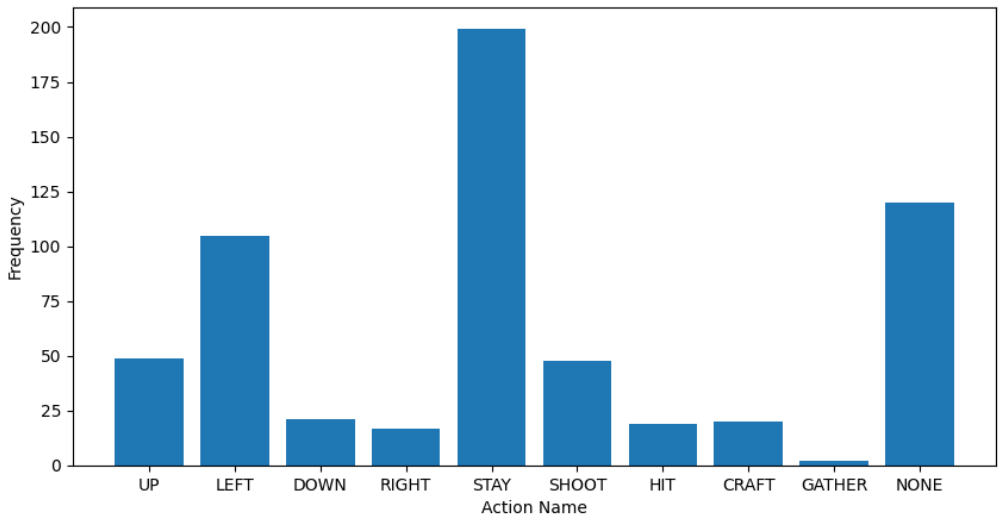


- `{'UP': 72, 'LEFT': 35, 'DOWN': 47, 'RIGHT': 84, 'STAY': 66, 'SHOOT': 73, 'HIT': 27, 'CRAFT': 38, 'GATHER': 38, 'NONE': 120}` is the new action frequency. We clearly see that left has gone from a frequency of 11 → 35 compared to base case which is a 350% increase.

CASE 2

The step cost of the STAY action is now zero.

- Algorithm now converges in **103** iterations. This means now the actions are more decisive.
- Policy has undergone a lot of changes
- In general now `STAY` action is lot more effective, this has lead to inflation in value of the states since all states have higher value than before.
- Now stay action has abundant use in `N` regardless of `MM` being dormant or not. This is due to the probability of `STAY` teleporting to the `E` state which is better than `C` when MM is dormant or ready and in best case IJ remains at `N` at no risk of MM.
- Now stay action has abundant use in `S` regardless of `MM` being dormant or not. This is due to the probability of `STAY` teleporting to the `E` state which is better than `C` when MM is dormant or ready and in best case IJ remains at `S` at no risk of MM.
- Now stay action is has abundant use in `W` regardless of `MM` being dormant or not. This is due to the probability of `STAY` teleporting to the `E` state which is better than `C` when MM is dormant or ready and in best case IJ remains at `W` at no risk of MM.

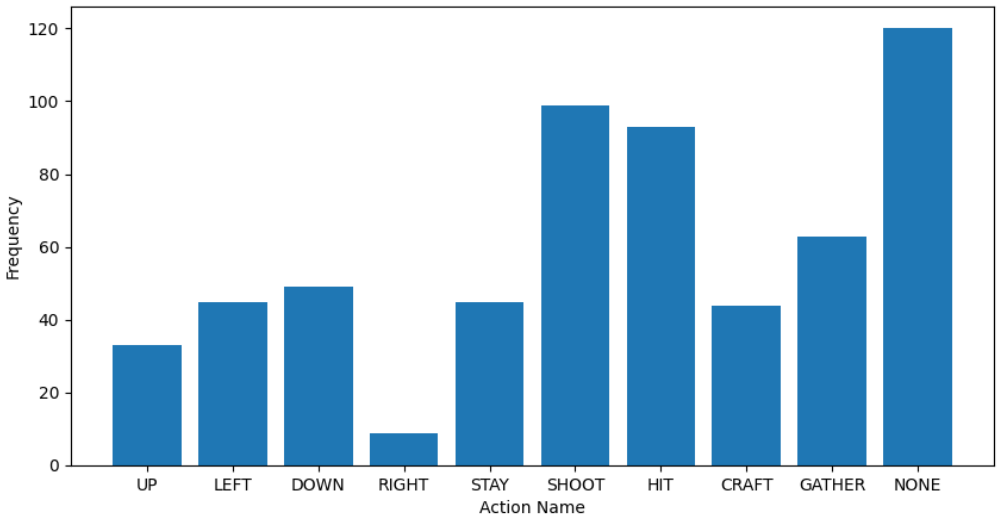


- `{'UP': 49, 'LEFT': 105, 'DOWN': 21, 'RIGHT': 17, 'STAY': 199, 'SHOOT': 48, 'HIT': 19, 'CRAFT': 20, 'GATHER': 2, 'NONE': 120}` is the new action frequency. We see that frequency of `STAY` has increased from 68→199 which is again over a 150% increase.

CASE 3

Change the value of gamma to 0.25

- It took **7** iterations to converge. Bellman error fell rapidly.
- Variance of values of state is much lower now. Almost all states except the terminal states have negative value. This is justified since gamma has been greatly reduced so the nearer states to terminal states extract less positive value from terminal state.
- Also now the action `RIGHT` has `9` uses. This is due to the fact even `E` does not have high enough value to draw IJ towards it
- These actions often end up in infinite loops for simulation. This shows this value of gamma is too low and IJ is not able to find a good path to end
- If `IJ` has `SHOOT` action available and the MM health is 25 he shoots and if `IJ` has `HIT` action and MM health is 50 he hits.



- `{'UP': 33, 'LEFT': 45, 'DOWN': 49, 'RIGHT': 9, 'STAY': 45, 'SHOOT': 99, 'HIT': 93, 'CRAFT': 44, 'GATHER': 63, 'NONE': 120}` is the new action frequency.

PART 3 LP

Procedure for making matrix A

Dimensions: `(Num_states, dim)`

Here `dim` is the number of valid `(state, action)` pairs and `Num_states` is the total number of states.

For every (state, action) we have added the outflow probability and subtracted the inflow probability. For `NONE` actions we have only added the outflow probability. Here is the relevant code snippet.

```
def initialize_a(self):
      a = np.zeros((self.num_states, self.dim))
      action_no = 0
      for state_no, cur_state in enumerate(self.states):
          for action in cur_state.actions:
              got_hit, results = action_value(action, cur_state)
              if len(results) <= 0:
                  print(cur_state, action)
              assert len(results) > 0
              for idx, (pr, next_state) in enumerate(results):
                  a[state_no][action_no] += pr  # outflow
                  if action != Actions.NONE:
                      a[self.getState(next_state).get_number()][action_no] -= pr  # inflow
              action_no += 1
              assert state_no == cur_state.get_number()
      self.a = a
```

Procedure for finding the optimal policy

As discussed in class given α, r, x, A vectors:

Optimal policy is given by:

$$max(rx)|Ax = \alpha, x \geq 0$$

This is directly translated to code as:

```
x = cp.Variable((self.dim, 1), 'x')
constraints = [
    cp.matmul(self.a, x) == self.alpha,
    x >= 0
]
objective = cp.Maximize(cp.matmul(self.r, x))
problem = cp.Problem(objective, constraints)
solution = problem.solve(verbose=True)
```

After running LP we get X vector which has expected value for each (state, action). So for a given state we select the **maximum expected value action** as the chosen action. Here is the relevant code:

```
def get_solution(self):
  xs = self.x.value.tolist()
  count = 0
  self.policy = []
  for state in self.states:
      action_len = len(state.actions)
      options = xs[count: count + action_len]
      max_arg = np.argmax(np.array(options))
      state.favoured_action = state.actions[max_arg]
      self.policy.append([str(state), state.favoured_action.name])
      count += action_len
```

Multiple policies

Yes there can be multiple policies:

- If two or more actions have same expected value for the given state then we can chose any of these actions. This can give rise to multiple policies.
- Changing the distribution of α vector to change the start state probabilities can result in different policies.
- Changing the reward and step cost can result in a different. If step cost is higher the algorithm will try to minimize the number of steps

- If any changes are made to `A` vector this would obviously result in a new policy.