CPSC 4610 – Artificial Intelligence

Homework 2

Seattle University

Jakob Balkovec

02/04/2025

Q1 (15 points) – CSP

Given Constraint Satisfaction Problem (CSP): FIVE + THREE = EIGHT

a. **Sketch the most efficient way to find a solution, if one exists, or prove that the constraints prohibit a solution**
b. **Using standard terminology (most/least constrained/constraining), explain your approach**

A1

Part of the answer is in the PDF (CSP.pdf)

Setup

- Each letter represents a unique digit $(0 \leftrightarrow 9)$
  - $\{F, I, V, E, T, H, R, G\}$

Constraints

- Each letter must have a different (unique) digit $(0 \leftrightarrow 9)$
- Leading digits cannot be zero (F, T, E)
- Arithmetic constraint (**PDF**)

- Since E appears in both numbers, we have to analyze the last digit rule
  - $E + E = T$ or $(E + E)\ \%\ 10 = T$

Table

E	**2E**	**Possible T**
0	0	Invalid, $E = 0$
1	2	2
2	4	4
3	6	6
4	8	8
5	10	Invalid, (Two digits)
6	12	2 (Already used)
7	14	4 (Already used)
8	16	6 (Already used)
9	18	8 (Already used)

Since T needs to be unique, we check all possible valid pairs:

- $\{(\boldsymbol{E = 1, T = 2}), (\boldsymbol{E = 2, T = 4}), (\boldsymbol{E = 3, T = 6}), (\boldsymbol{E = 4, T = 8})\}$

Check the carry from $E + E$:

- For $F + H \geq 10$ for the leftmost column. Checking possible values, we find that there is no "valid" to maintain uniqueness while keeping the sum constraint valid. Every assignment either violates uniqueness or fails to create a consistent sum.

Conclusion

- Since no valid digit assignments satisfy all constraints, we can conclude that no solution exists for this CSP.

Files:

CSP.pdf
CSP_py.pdf

Q2 (15 points) – Expectimax & MDP

Identify and explain the key similarities, and differences, if any between Expectimax and Markovian Decision Processes(MDPs).

A2

Expectimax

- Expectimax is most commonly found in adversarial games with some chance elements (e.g. dice rolls, board games, poker ...). Expectimax works like Minimax but instead of a minimizer we have chance nodes that take the expected value (probability p has to add up to 1). The goal is to maximize the expected utility.

MDP (Markov Decision Process)

- MDP is most commonly used in decision-making with stochastic transitions (e.g. finance, robotics). It is defined by: states (S), actions (A), transition probabilities ($P[s'|s,a]$), reward function ($R[s,a,s']$), and an optional discount factor (γ)

Similarities

- They are similar in the fact that they both does with uncertainty in transitions, they both optimize "long-term" rewards and they both use Bellman's equations (value of a state based on the rewards from taking actions and the expected value of future states)

Differences

- They are different in the fact that Expectimax is mostly used in games, where outcomes depend on both player actions and chance, while MDPs are better suited for decision-making tasks, where outcomes follow probabilistic rules. Expectimax builds a game tree with chance nodes and evaluates it recursively, whereas MDPs rely on state-action transitions and solve for optimal policies using techniques like policy iteration or value iteration. The key difference is how they handle uncertainty, Expectimax deals with random chance events, while MDPs focus on transition probabilities between states.

Q3 (25 points) – MicroBlackJack

Review the MicroBlackJack problem covered in class and then extended in the 1/29/25 in-class design exercise. Show your work (tables, etc.).

a. **Complete the in-class design exercise, solving the MDP with Value Iteration**
b. **Compare intuitive expectations for gameplay (e.g. stop with value 4) to your results**

A3

Part of the answer is in the PDF (MicroBlackJack.pdf)

Policy

State ***s***	Optimal Action
0	Draw
2	Draw
3	Stop
4	Stop
5	Stop

Comparison with Intuition

- Intuition says to stop early (e.g. Stop at 3):
  - The MDP confirms this, since the optimal action is to **stop** at 3, 4, and 5

- Optimal policy suggests drawing at state 2:
  - This makes sense, because drawing at 2 has a good chance of leading to states 4 or 5. Both of those give off a better value compared to stopping at 2

- State 0 should always draw:
  - We should always draw at 0, since there is no obtainable reward if we stop at 0. Drawing is "preferable"

Conclusion

- The extracted policy aligns well with intuitive play, except for state 2, where drawing provides better long-term rewards. Value iteration converges in 4 iterations and confirms that stopping at state 3 or higher is optimal. I think the takeaway here is that MDP solutions can provide "surprising" but optimal decision that will be better than naive/greedy strategies in most cases.

Files:

MicroBlackJack.pdf

Q4 (35 points) MicroBlackJack

Modify the MicroBlack in-class design exercise, noted above, so that gameplay is extended until the sum of card values exceeds 6 (rather than 5).

a. **Solve this MDP with Policy Iteration**
b. **Discuss differences between this process & solution compared to #3**
c. **What is the effect of extending gameplay until the sum exceeds 7?**

A4

Part of the answer is in the PDF (MicroBlackJack2.pdf)

Setup

- The problem setup remain the same as in Q3, except:
  - States $S = \{0, 2, 3, 4, 5, 6, Done\}$
  - Terminal State **(Done)** is reached when the sum exceeds 6
  - Reward Function remains the same (sum changes from **5** to **6**)

Using a different table approach (simpler + less work)

Policy

State ***s***	Optimal Action
0	Draw
2	Draw
3	Draw
4	Stop
5	Stop
6	Stop

Comparison to Q3

- **Q3:**
  - The max sum was 5 ($sum = 5$), we stopped at 3
- **Q4:**
  - The max sum is 6 ($sum = 6$), we are now drawing at 3 (optimal) because it leads to 5 or 6, both of which are rewarding.

- **Main Difference:**
  - More opportunities for higher rewards encourage riskier play

Extending the game to 7:

- If the game continues until sum exceeds 7, the stopping threshold will most likely shift to 5, because drawing at 4 would be safer. The optimal policy would likely be something along the lines of this:

State ***s***	Optimal Action
0	Draw
2	Draw
3	Draw
4	Draw
5	Stop
6	Stop

Files:

MicroBlackJack2.pdf

Q5 (10 points) – POMDPs

***Briefly* describe a card or video game that would be modelled by a POMDP and explain why a MDP would be insufficient.**

A5

Poker

Why a POMDP Works:

- Poker is a great example of a POMDP because players never have all the information they need. Unlike chess for example, where every piece is visible, poker involves hidden information, specifically, your opponent's cards. Since you can't see their hand, you have to rely on belief states, which are your best guesses based on available information. These beliefs are updated by analyzing observations like betting patterns, previous moves, and potential bluffs. Every action an opponent takes provides clues, but there's always uncertainty, making a POMDP the right framework for modeling decision-making in poker.

Why an MDP Fails:

- A standard Markov Decision Process (MDP) assumes full observability, meaning the agent would need access to all relevant game information, including the opponent's hand. In poker, this isn't really realistic because players must make decisions based on incomplete information. MDPs work well in environments where all states and transitions are known, but poker requires reasoning under uncertainty.

Q1 ⇒ CSP

Define variables:

```
    F I V E
+ T H R E E
-----------
  E I G H T
```

↳ {F, I, V, E, T, H, R, G, H, T}

⇒ Arithmetic constraint:

FIVE + THREE = EIGHT

= (1000F + 100I + 10V + 1E) + (10000T + 1000H + 100R + 10E + 1E) =

(10000E + 10001 + 100G + 10H + 1T)

⇒ F, T, E ≠ 0

MCV: [E]; since it appears 3 times

MRV: [T]; assign early, since it's the leading digit and part of a 5-digit num

Solve:

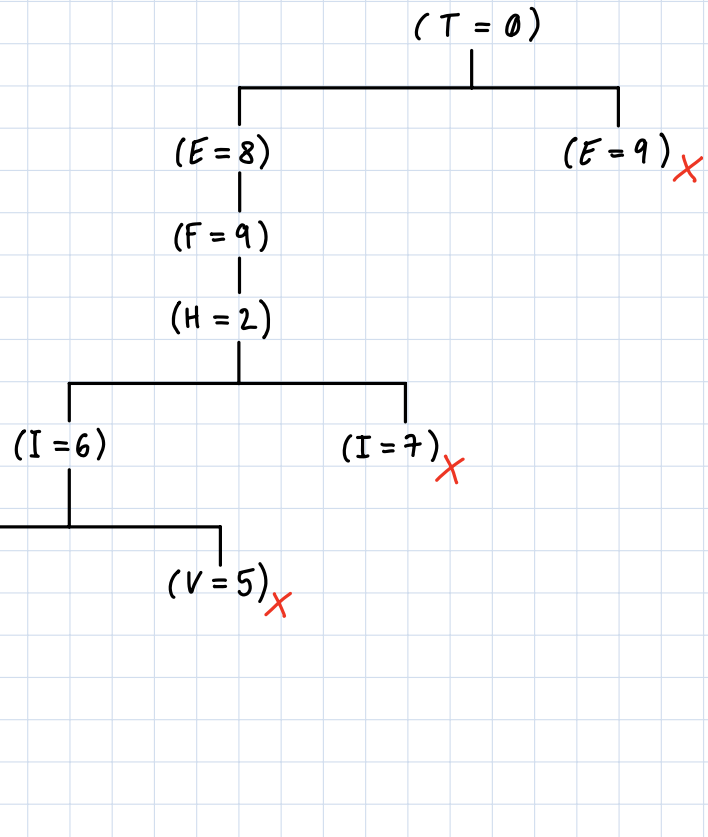
```
    F I V E
+ T H R E E
-----------
  E I G H T
```

↓

```
    9 1 7 4
  8 5 2 4 4
-----------
  4 1 3 5 8
```

✗ WRONG

Tree:



Trial & Error

~ T = 0, ∄F, H > 10

~ E + E = T

↳ or E + E % 10 = T

{0, 1, 2, 3, 4, 5, 6, 7, 8}

```
"""CSP script

  Runs through all possible permutations of the digits 0-9 and assigns them to the l
etters
  in the words FIVE, THREE, and EIGHT. The script then checks if the assignment is a
 valid
  solution to the cryptarithmetic puzzle. If a valid solution is found, the script p
rints
  the solution. If no solution is found, the script prints that no solution exists.
"""

from itertools import permutations

def verify_solution(mapping):
    # convert the words to numbers using the mapping
    FIVE = mapping['F'] * 1000
    + mapping['I'] * 100
    + mapping['V'] * 10
    + mapping['E']

    THREE = mapping['T'] * 10000
    + mapping['H'] * 1000
    + mapping['R'] * 100
    + mapping['E'] * 10
    + mapping['E']

    EIGHT = mapping['E'] * 10000
    + mapping['I'] * 1000
    + mapping['G'] * 100
    + mapping['H'] * 10
    + mapping['T']

    return FIVE + THREE == EIGHT

def solve_csp():
    letters = ['F', 'I', 'V', 'E', 'T', 'H', 'R', 'G']
    digits = range(10) # yields 0-9

    for perm in permutations(digits, len(letters)):
        mapping = dict(zip(letters, perm))

        # F, T, and E must not be 0
        if mapping['F'] == 0 or mapping['T'] == 0 or mapping['E'] == 0:
            continue

        # check if valid
        if verify_solution(mapping):
            return mapping  # return first valid solution

    return None  # no solution


solution = solve_csp()
if solution:
    print("Solution found:")
    for letter, digit in sorted(solution.items()):
        print(f"{letter} = {digit}")
else:
    print("No solution exists.")
```

Q3 ~ Micro Black Jack

States: $S = \{0, 2, 3, 4, 5, \text{Done}\}$

Actions: $A = \{\text{Draw}, \text{Stop}\}$

Transition P: $P(s'|s,a)$ [MDP]

⇒ if "draw" is chosen
- ~ $P(s+2|s, \text{draw}) = 1/3$
- ~ $P(s+3|s, \text{draw}) = 1/3$
- ~ $P(s+4|s, \text{draw}) = 1/3$

⇒ if "stop" is chosen
- ~ game moves to DONE (terminal)

Reward Function: $R(s, a, s')$

⇒ if "stop" is chosen
- ~ $R(s, \text{stop}, \text{done}) = s$; where s is the score @ stopping point

⇒ if "draw" is chosen
- ~ $R(s, \text{Draw}, s') = 0$; no immediate reward

Discount Factor: $\gamma = 1$ (game ends quickly, no γ needed)

Value Iteration

⇒ set $V(s) = 0$; $\forall s$

⇒ use Bellmans equation until values converge

$$V(s) = \max\left(R(s, \text{stop}, \text{done}) \sum_{s'} P(s'|s, \text{draw})\, V(s')\right)$$

Tables + Iterations

Computing $V_1(s)$

State s	Action	Value Calculation	$V_1(s)$
0	Stop	0	0
	Draw	$\frac{1}{3}(0 + 1V_0(2)) + \frac{1}{3}(0 + 1V_0(3)) + \frac{1}{3}(0 + 1V_0(4)) = 0$	0
2	Stop	2	2
	Draw	$\frac{1}{3}(0 + 1V_0(4)) + \frac{1}{3}(0 + 1V_0(5)) + \frac{1}{3}(0 + 1V_0(0)) = 0$	2
3	Stop	3	3
	Draw	$\frac{1}{3}(0 + 1V_0(5)) + \frac{2}{3}(0 + 1V_0(0)) = 0$	3
4	Stop	4	4
	Draw	0 (because score ≥ 6 is a loss)	4
5	Stop	5	5
	Draw	0 (loss)	5

$\Rightarrow$ Observations: S_4 & S_5 have max values when stopping $\Rightarrow$ Drawing = loss

Computing $V_2(s)$

State s	Action	Value Calculation	$V_1(s)$
0	Stop	0	0
	Draw	$\frac{1}{3}(2) + \frac{1}{3}(3) + \frac{1}{3}(4) = 3$	3
2	Stop	2	2
	Draw	$\frac{1}{3}(4) + \frac{1}{3}(5) + \frac{1}{3}(0) = 3$	3
3	Stop	3	3
	Draw	$\frac{1}{3}(5) + \frac{2}{3}(0) = \frac{5}{3} + 0 = \frac{5}{3}$	3

$\Rightarrow$ Observations: S_0 has an expected value of 3 if drawing is chosen

Computing $V_3(s)$

State s	Action	Value Calculation	$V_1(s)$
0	Stop	0	0
	Draw	$\frac{1}{3}(3) + \frac{1}{3}(3) + \frac{1}{3}(4) = \frac{10}{3}$	$\frac{10}{3}$

$\Rightarrow$ Observations: S_0 reaches a stable expected value when drawing

Final Values + Policy Extraction

State, Action

$T = \{(0, \text{Draw}), (2, \text{Draw}), (3, \text{Stop}), (4, \text{Stop}), (5, \text{Stop}); (S_n, A)\}$

Q4 - Micro Black Jack

Value Iteration

⇒ set $V(s) = 0$; $\forall s$

⇒ use Bellmans equation until values converge

$$V(s) = \max\left(R(s, \text{stop}, \text{done}) \sum_{s'} P(s' | s, \text{draw})\, V(s')\right)$$

Iteration 1; Initial guess $[V(s) = 0]$

State (s)	Stop Value R(s, stop)	Draw Value $[\frac{1}{3}V(s+2) + \frac{1}{3}V(s+3) + \frac{1}{3}V(s+4)$	V(s)*
0	0	$\frac{1}{3}(2+3+4) = 3$	3
2	2	$\frac{1}{3}(4+5+6) = 5$	5
3	3	$\frac{1}{3}(5+6+0) = 3.66\overline{6} \approx 3.67$	3.67
4	4	$\frac{1}{3}(6+0+0) = 2$	4
5	5	0	5
6	6	0	6

Update Policy

V(s)

↳ if $V(s, \text{draw}) > V(s, \text{stop}) =$ Draw

else stop

Old policy

(state, policy(d/s), V(s, draw), V(s, stop))

{(0, d, 3, 0),
(2, d, 5, 2),
(3, d, 3.67, 3),
(4, d, 2, 4),
(5, s, 0, 5),
(6, s, 0, 6)}

New policy

(state, policy(d/s), V(s, draw), V(s, stop))

{(0, d, 3, 0),
(2, d, 5, 2),
(3, d, 3.67, 3),
(4, s, 2, 4),
(5, s, 0, 5),
(6, s, 0, 6)}

Iterate until convergence

State (s)	Stop Value R(s, stop)	Draw Value $[\frac{1}{3}V(s+2) + \frac{1}{3}V(s+3) + \frac{1}{3}V(s+4)$	V(s)*
0	0	$\frac{1}{3}(5+3.67+4) = 4.22$	4.22
2	2	$\frac{1}{3}(4+5+6) = 5$	5
3	3	$\frac{1}{3}(5+6+0) = \frac{11}{3} = 3.67$	3.67
4	4	2	4
5	5	0	5
6	6	0	6