Solving the Chomp MDP:
A Comparative Analysis of SARSA and Heuristic Strategies

Introduction and Problem Description: The Game of Chomp

The subject of this project is Chomp, a two-player impartial strategy game played on a rectangular grid, conceptualized as a chocolate bar. In this project, we aim to model the game as a Sequential Decision-Making problem to analyse optimal strategies and train a Reinforcement Learning agent to master the game.

The Rules and Mechanisms of Chomp

The game begins with a rectangular grid of dimensions (Rows X Columns). For this specific implementation, we utilize a starting state of a 4×5 grid. The top-left square, located at coordinate (0,0), is designated as the "poisoned" piece.

The game play proceeds according to the following rules:

- Turn-Based Play: Two players take turns selecting a remain square on the grid.
- The “Chomp” Mechanism: When a player selects a square of chocolate on the grid at coordinate (R,C) they “eat” (remove) that square along with all the remaining squares that are located below and to the right of it.
- Constraints: A player cannot select a square that has already been removed from the board.
- Terminal state: The game ends when the board is empty.

The Objective

The objective of chomp is to avoid selecting the designated poison square at the top left with the coordinates (0,0). The last player to move loses.

- Loss condition: A player loses when they are forced to select the poison square
- Win condition: A player wins if they make a move that leaves only the poison square remaining, thereby forcing the opposing player to “eat” it on the subsequent turn.

As Chomp is a finite, deterministic, perfect-information game with no possibility of a draw one player must have a winning strategy from the starting position. In this project, we will formulate these mechanics as a Markov Decision Process (MDP) to computationally solve for this strategy.

1 State Space

The game is played on a grid of size R×C (where R=4, C=5). Rather than representing the board as a raw binary matrix, we represent the state as a tuple of row lengths, as the rules of Chomp imply that if a square at column c in row r exists, all squares to its left (<c) **must also exist**. **Furthermore, row r cannot be longer than row r−1.**

Definition: $S = \{(x_0, x_{1,} x_2 \dots, x_{R-1}) \mid C \geq x_0 \geq x_1 \geq x_2 \geq \cdots \geq x_{R-1} \geq 0, x_i \in Z\}$

The state is defined as a list (tuple) of numbers
R: This is the number of rows 4
x_i: This represents the number of chocolate squares remaining in row i.
C: This is the maximum number of columns (5). So, no row can have more than 5 chocolate squares.
≥: We use greater than here because it describes the physical property of the game.

In Chomp, since you "eat" everything to the right and below of the square you choose, **a lower row can never be longer (greater than) the row above it.** The chocolate bar can have a staircase shape going down, but we can not have a staircase shape where the bottom is longer than the top.

Initial State: (S_0): (5,5,5,5)

This represents a full 4 x 5 grid. There are 4 numbers in the list (4 row) and every number is 5 (every row is filled by 5 columns). Visually this can be represented as:

$$R = 0: [P][X][X][X][X]$$
$$R = 1: [X][X][X][X][X]$$
$$R = 2: [X][X][X][X][X]$$
$$R = 3: [X][X][X][X][X]$$

P: The poison piece at (0,0)
X: Safe chocolate Pieces.

Terminal State $(S_{terminal})$: The state where the poison block at (0,0) is chosen. In implementation, we may treat the state where *only* the poison remains, (1,0,0,0,0,0), as the effective terminal state for the winner. This matters because the opponent must eat the poison on their next tern. Therefore, it will be easier to treat this state as the goal or found item, rather than simulating the opponent eating it.

Calculating the State Space:

Due to the rules and constraints of Chomp, a valid state cannot contain any 'floating' pieces of chocolate; if a square exists at least one other square must also exist either to its left or above, or to its left and above – with the exception of the terminal state. Consequently, the boundary between the "eaten" and "uneaten" segments forms a "staircase" line from the bottom-left to the top-right of the grid.
Mathematically, every unique state corresponds to a unique path along the grid lines consisting of exactly vertical steps and horizontal steps. The total number of valid states is therefore equivalent to the number of ways to arrange these steps, calculated using the binomial coefficient formula, where (total steps) and (vertical steps)

$$Number\ of\ states = \frac{(R+C)}{R} = \frac{(4+5)}{4} = \binom{9}{4}$$

$$\frac{n!}{k!\,(n-k)!} = \frac{9!}{4!\,(9-4)!} = \frac{9!}{4!\,(5)!} = 126$$

This means that we have a state space consisting of 126 valid states (including the terminal state). This compact state space allows for efficient training of the SARSA.

2. Actions

The set of actions – This is the specific coordinates that we choose to chomp

- In the game in an action is picking a square (r,c) (row, column) - minus the coordinate of the poison piece.
- The effect of picking (r,c) is that you remove the square and all the squares that are both below it and to the right of it.

1. There is one constraint: you can't pick a square that has already been eaten.

- The terminal action is choosing the poison chocolate.

3. The Transition Probability Function - P(s′|s,a). If you take action A, what state do you end up in?

- The transition represents the opponent's turn

1. State_t(Your turn) --> Action (chomp) ---> Opponent moves ---> State_t+1 (Your trun again)
2. P depends on your opponent's strategy.

4. The reward function – the feedback for taking an action.

- The reward in the game is delayed until the end of the game.
- Win (+1): You force the opponent to eat the poison square (1,1)
- Loss (-1): You eat the poison square
- Step (0): All other moves possible

5. The Discount factor

- Not applicable for our game.