Assignment 1

AL3021 | Search Methods in AI

Submitted By - Tushar Goyal

UG20210087

Problem Description

Bacteria vs. Neutrophils - A Game in the Microbiological World

In this state space search problem, we model a simplified biological scenario where bacteria and neutrophils interact. The goal is to control the population of bacteria using neutrophils while adhering to certain constraints. The state space represents the populations of bacteria and neutrophils, the level of bacterial resistance, and the constraints involve the dynamics of their interactions.

Some bacteria can develop resistance to neutrophil attacks, making them harder to eliminate. We will represent bacterial resistance as a parameter R, where R is an integer between 0 and 5, with 0 indicating no resistance and 5 indicating maximum resistance.

Domain Representation

- **State Representation:** A state is represented as a tuple (B, N, R), where:
    - B: The number of bacteria.
    - N: The number of neutrophils.
    - R: The level of bacterial resistance (an integer from 0 to 5).
- **Start State:** Users can specify the initial state by providing values for B, N, and R.
- **Goal State:** The goal is to eliminate all bacteria, so the goal state is defined as (0, N, R) for any valid values of N and R.

MoveGen (Neighborhood Function)

The move generation function generates valid neighboring states based on the following rules:

1. **Bacteria Reproduction:** Bacteria can reproduce at a certain rate, so a valid neighbor state can be (B+1, N, R).
2. **Neutrophil Attack:** Neutrophils can attack and eliminate bacteria, but the effectiveness of the attack depends on bacterial resistance. A valid neighbor state can be (B-1, N-1, R), where this transition is only allowed if B > 0, N > 0, and R < 5 (indicating that bacteria are not fully resistant).
3. **Neutrophil Migration:** Neutrophils can migrate to the environment without attacking bacteria. A valid neighbor state can be (B, N-1, R), where N > 0.
4. **Bacteria Die:** Bacteria can die due to environmental factors or antibiotics. A valid neighbor state can be (B-1, N, R), where B > 0.
5. **Bacterial Resistance Development:** Bacteria can develop resistance over time. A valid neighbor state can be (B, N, R+1), where this transition is allowed as long as R < 5 (indicating that bacteria can further develop resistance) and B > 0.

Pseudocode for MoveGen Function

```
# Define the state representation as a tuple (B, N, R)
# B: Number of bacteria
# N: Number of neutrophils
# R: Level of bacterial resistance (an integer from 0 to 5)

# Define the move generation (neighborhood) function
def movegen(state):
    neighbors = []

    # Rule 1: Bacteria Reproduction
    if state[0] > 0:
        new_state = (state[0] + 1, state[1], state[2])
        neighbors.append(new_state)

    # Rule 2: Neutrophil Attack
    if state[0] > 0 and state[1] > 0 and state[2] < 5:
        new_state = (state[0] - 1, state[1] - 1, state[2])
        neighbors.append(new_state)

    # Rule 3: Neutrophil Migration
    if state[1] > 0:
        new_state = (state[0], state[1] - 1, state[2])
        neighbors.append(new_state)

    # Rule 4: Bacteria Die
    if state[0] > 0:
        new_state = (state[0] - 1, state[1], state[2])
        neighbors.append(new_state)
    # Rule 5: Bacterial Resistance Development
    if state[2] < 5 and state[0] > 0:
        new_state = (state[0], state[1], state[2] + 1)
        neighbors.append(new_state)

    return neighbors

# Example usage:
initial_state = (0, 0, 0)
print("Initial State:", initial_state)
print("Neighbors:", movegen(initial_state))
```

GoalTest Function

The goal test function checks if the current state is a goal state. It returns true if the state is (0, N, R) for any valid values of N and R, where all bacteria are eliminated.

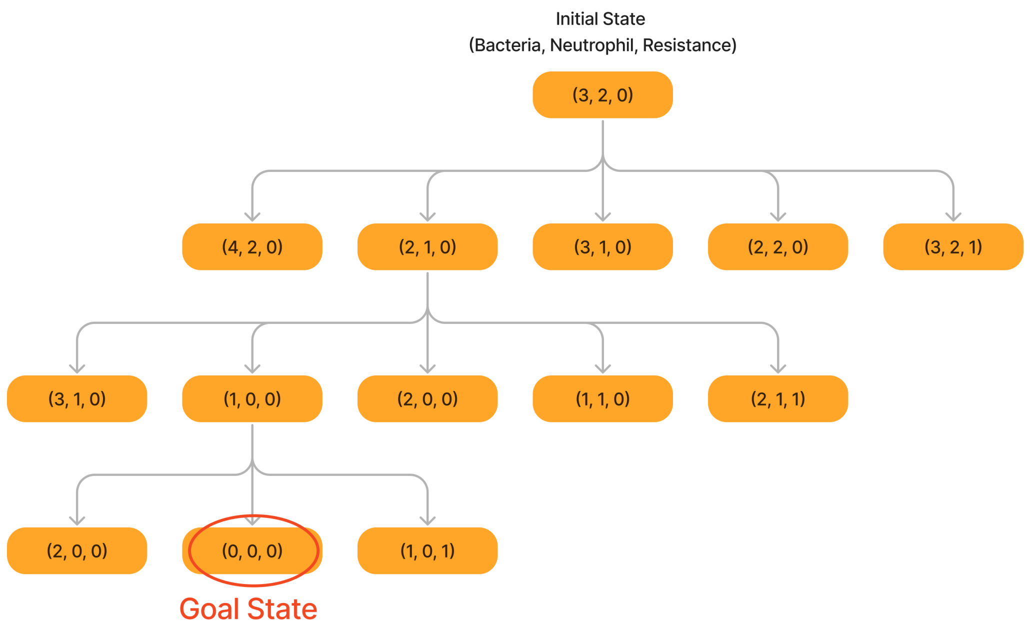
Pseudocode for GoalTest Function

```
# Define the state representation as a tuple (B, N, R)
# B: Number of bacteria
# N: Number of neutrophils
# R: Level of bacterial resistance (an integer from 0 to 5)

# Define the goal test function
def goaltest(state):
    # Check if all bacteria are eliminated (B == 0)
    return state[0] == 0

# Example usage:
initial_state = (0, 0, 0)
print("Initial State:", initial_state)
print("Goal Test Result:", goaltest(initial_state))
```

Example: Limited State Space Graph for (3, 2, 0)



The search algorithm will explore the state space by applying the move generation function to find a sequence of actions that lead to the goal state, i.e., the complete elimination of bacteria while considering bacterial resistance. Players must strategize and manage bacterial populations while taking into account evolving resistance levels, adding an extra layer of complexity and decision-making to the game.