**Комп’ютерний практикум №4**

**Методи пошуку в умовах протидії**

**ПІБ: Головня Олександр**

**Група: ІП-11**

**Мета роботи:** ознайомитись з методами пошуку в умовах протидії та дослідити їх використання для інтелектуального агента в типовому ігровому середовищі.

***З*авдання:** обрати середовище, що моделює гру з нульовою сумою, та задачу, що містить декілька агентів, які протидіють один одному. В обраному середовищі вирішити поставлену задачу, реалізувавши один з методів пошуку в умовах протидії. Реалізувати власну функцію оцінки станів. Виконати дослідження впливу деякого фактору середовища.

**Номер варіанту: 15**

**Завдання для варіанту:**

|  |  |  |
| --- | --- | --- |
| **Номер студента/ бригади** | **Метод** | **Задача дослідження** |
| 15 | Expectimax | Вплив перешкод на карті |

**Середовище і задача:** коротко опишіть обране середовище. Формалізуйте середовище в термінах агентів, станів, можливих дій агентів, переходів і винагород

В якості середовища було обрано середовище для гри Pacman.

**Агент**: Представляє гравця або контролер, який керує Pac-Man.

**Стани**: Стан гри включає в себе положення Pac-Man на карті, положення ворогів (привидів), розміщення точок для збору, а також можливі зміни у станах об'єктів, таких як вороги або точки після взаємодії з Pac-Man.

**Можливі дії агентів**:

Рух Pac-Man або Привидів вгору, вниз, вліво або вправо.

**Переходи**: Переходи відбуваються при виконанні дій агентів. Наприклад, якщо Pac-Man рухається до точки, точка зникає, а винагорода збільшується. Якщо Pac-Man зіштовхується з ворогом, гра може закінчитися.

**Винагороди**:

За кожну зібрану точку Pac-Man отримує певну кількість балів.

За з'їдання спеціальної точки Pac-Man може тимчасово з'їсти ворогів.

За зіткнення з ворогом Pac-Man може втратити життя і/або почати гру з початку.

**Метод вирішення задачі:** ExpectiMax - метод пошуку в умовах протидії, що вважає, що відома оцінка імовірності вибору тих чи інших дій суперниками.

Метод ExpectiMax є варіацією алгоритму Minimax, основна ідея якого полягає в тому, щоб розглядати деякі стани гри як можливість випадкового вибору дій. У випадку, коли є невизначеність, ExpectiMax робить припущення щодо можливих результатів і використовує середнє значення (очікуване значення) цих результатів для визначення найкращого ходу.

**Реалізація методу:**

class ExpectimaxAgent(MultiAgentSearchAgent):

  def getAction(self, gameState: GameState) -> str:

    def expectimax(agent,depth, gameState):

      if gameState.isLose() or gameState.isWin() or depth == self.depth:

        return self.evaluationFunction(gameState)

      if agent == 0:

        return max(expectimax(1, depth, gameState.generateSuccessor(agent, newState)) for newState in gameState.getLegalActions(agent))

      else:

        nextAgent = agent + 1

        if gameState.getNumAgents() == nextAgent:

          nextAgent = 0

        if nextAgent == 0:

          depth +=1

        return sum(expectimax(nextAgent, depth, gameState.generateSuccessor(agent, newState)) for newState in gameState.getLegalActions(agent)) / float(len(gameState.getLegalActions(agent)))

    maximum = float("-inf")

    action = Directions.WEST

    for agentState in gameState.getLegalActions(0):

      utility = expectimax(1,0,gameState.generateSuccessor(0,agentState))

      if utility > maximum or maximum == float("-inf"):

        maximum = utility

        action = agentState

    return action

**Результати застосування розробленого методу:**

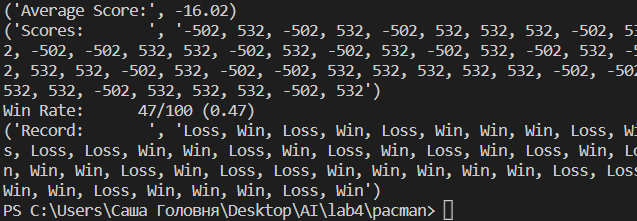
*python pacman.py -p ExpectimaxAgent -l trappedClassic -a depth=3 -q -n 100*

Запустимо цей сценарій 100 разів, щоб оцінити результати.

Пакмен буде вигравати в середньому в половині випадків і для даної команди

кінцевий рахунок буде -502 якщо Пакмен програє, та 532 або 531 якщо

виграє. Ці значення можна використовувати для перевірки реалізації



**Оцінка результатів:**

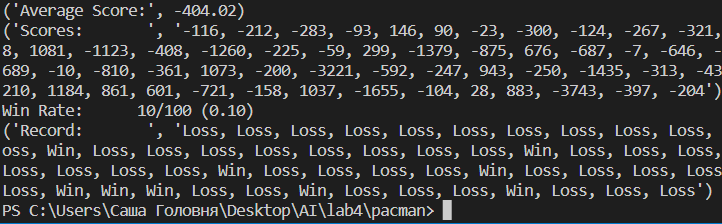
Отримані результати вказують на те, що алгоритм ExpectiMax, застосований до гри Pac-Man, демонструє прийнятну ефективність.

Очікується, що привиди будуть рухатися рандомно в кожному стані гри, оскільки їх рух визначається шляхом очікуваного кроку. Це відрізняється від реальної поведінки привидів у грі Pac-Man, де їх рух має певний рівень стратегії і непередбачуваності.

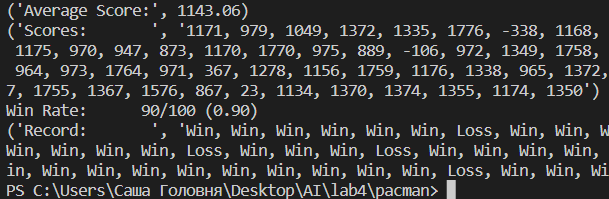
Щодо доцільності застосування методу, він може бути ефективним у вирішенні задач у грі Pac-Man. Однак, варто розглянути можливість оптимізації алгоритму, шляхом покращення оцінювальної функції.

**Власна функція оцінки станів:** В оціночній функції, враховуються різні аспекти стану гри Pac-Man, такі як: відстань до найближчої їжі та привидів, кількість доступних капсул і загальний бал гри. Функція обчислює ваговану суму цих метрик для оцінки поточного стану гри і допомагає агентові Pac-Man у вирішенні, які ходи робити для досягнення кращого результату.

python pacman.py -l smallClassic -p ExpectimaxAgent -q -n 100:



python pacman.py -l smallClassic -p ExpectimaxAgent -a evalFn=better -q -n 100:



**Реалізація власної функції оцінки станів:**

def betterEvaluationFunction(currentGameState: GameState) -> float:

  newPos = currentGameState.getPacmanPosition()

  newFood = currentGameState.getFood()

  newFoodList = newFood.asList()

  min\_food\_distance = -1

  for food in newFoodList:

      distance = util.manhattanDistance(newPos, food)

      if min\_food\_distance >= distance or min\_food\_distance == -1:

          min\_food\_distance = distance

  distances\_to\_ghosts = 1

  proximity\_to\_ghosts = 0

  for ghost\_state in currentGameState.getGhostPositions():

      distance = util.manhattanDistance(newPos, ghost\_state)

      distances\_to\_ghosts += distance

      if distance <= 1:

          proximity\_to\_ghosts += 1

  """Obtaining the number of capsules available"""

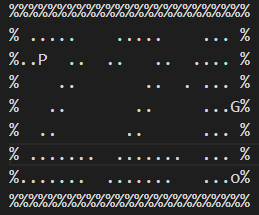
  newCapsule = currentGameState.getCapsules()

  numberOfCapsules = len(newCapsule)

  """Combination of the above calculated metrics."""

  return currentGameState.getScore() + (1 / float(min\_food\_distance)) - (1 / float(distances\_to\_ghosts)) - proximity\_to\_ghosts - numberOfCapsules

**Задача дослідження впливу параметра алгоритму чи фактору середовища:** результати серії експериментів з різними значеннями параметру, бажано у вигляді зведеної таблиці, графіку чи діаграми. Зробити висновки щодо впливу зміни параметрів.

Для дослідження впливу перешкод я використав карту openClassic вигляду:  


Далі, випадковим чином буду добавляти перешкоди на карту і підраховувати Average Score та WinRate, використовуючи один і той самий сценарій:  
*pacman.py -l openClassic -p ExpectimaxAgent -a evalFn=better -q -n 100*

|  |  |  |
| --- | --- | --- |
| Кількість перешкод | WinRate | Average Score |
| 0 | 99 | 1230.96 |
| 2 | 98 | 1230.26 |
| 4 | 100 | 1127.48 |
| 8 | 98 | 1123.73 |
| 16 | 98 | 932.01 |
| 25 | 95 | 974.0 |
| 36 | 93 | 955.53 |
| 49 | 90 | 872.46 |
| 64 | 55 | 407.42 |

Оскільки алгоритм Експектімакса моделює поведінку всіх привидів як випадковий вибір ходу, наявність перешкод може ускладнити йому знаходження оптимального рішення.

Таким чином:

Перешкоди можуть зробити обчислення більш складними, оскільки алгоритм повинен розглядати більшу кількість можливих ходів у зв'язку з обмеженими маршрутами через перешкоди.

Перешкоди можуть зменшити шанси на перемогу, оскільки алгоритму може бути складніше уникати привидів або досягати цілей через перешкоди.

У разі наявності перешкод алгоритм може змушений приймати більше випадкових рішень через обмежену доступність маршрутів.