מבוא לבינה מלאכותית
236501

דו"ח הגשה – תרגיל בית רטוב 2

מתן צחורי 208936989

אלון פנפיל 318598166

חלק 1

1. בעיית החיפוש במרחב מצבים עבור ריבוי סוכנים מוגדר ע"י $< S, A, f, c, s_0, R >$. עבור בעיית החיפוש שלנו:

S – מרחב המצבים, נגדיר מצב במרחב החיפוש ע"י מכלול המידע שצריך לדעת הסוכן:

$$S = \{s \mid s = (Agt1, Agt2, Pckg1, Pckg2, C, T\}$$

כאשר: $Agt\{i\}$ מכיל את המידע של סוכן i - $(Position, Fuel, Credits, StepsLeft, Package)$

$Pckg\{i\}$ מכיל את המידע על חבילה i - $(Position, Destination)$

C מכיל את מיקום תחנות הטעינה $(Position1, Position2)$

T מכיל את תור איזה סוכן

A – מרחב הפעולות שיכול לבצע סוכן. עבור בעיית החיפוש שלנו, לשני הסוכנים אותם פעולות והם:
$\{“move\_north”, ”mode\_south”, ”move\_west”, ”mode\_east”, ”park”, ”pickup”, ”dropoff”, ”charge”\}$

f – פונקציית מעבר בין מצבים, מעבר בין מצב נתון למצב אפשרי. עבורינו תיתן את המצבים הבאים של הלוח עבור כל מהלך חוקי.

c – פוקנציית מחיר עבור פעולה חוקית שביצענו.

s_0 – מיקום התחלתי, מוגדר רנדומלית על סמך פרמטרי כניסה (או דטרמניסטים לפי $seed$).

R - פונקציית reward לכל רובוט, במשחק שלנו לשני הסוכנים תהיה אותה פונקציה, נרצה לתגמל פעולות כמו איסוף חבילות והורדת החבילות ביעד שלהן.

2. נגדיר היוריסטיקה רחבה יותר שלוקחת בחשבון מספר מאפיינים מהמשחק עבור סוכן i:
נרצה להתחשב בניקוד שלנו, בניקוד הרובוט השני, במצב הדלק, במיקומי החבילות, מיקומי היעדים של החבילות.

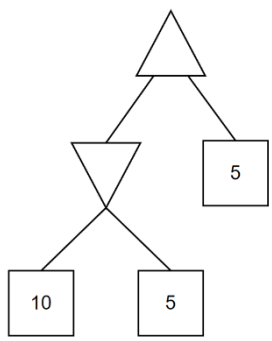
נסמן:
R_i – הרובוט שתורו, P – מיקום, H_i – החבילה הקרובה לרובוט ה-i, S_i – ניקוד הרובוט i, F – דלק, C_i – תחנת תדלוק הקרובה ביותר לרובוט ה-i, ונסמן ב-man את מרחק מנהטן.

$$Heuristic = \begin{cases} R_i.F - S_i - C_i & R_i.F \leq C_i \; and \; 0 < S_i < S_{(i+1)mod2} \\ 3 \cdot S_i + reward(R_i.H) - man(R_i.P, R_i.H.dest) & o.w \; R_i.H \neq None \\ 3 \cdot S_i - man(R_i.P, H_i) & o.w \end{cases}$$

חלק 2

1. בשימוש באלגוריתם $Minimax$ מוגבל משאבים, היתרונות לשימוש ביוריסטיקה שהיא קלה לחישוב (אם כי פחות מיודעת) הם שכך נאפשר לאלגוריתם לרוץ לעומק ארוך יותר בזמן העומד לרשותו, מה שיאפשר לאלגוריתם להתקרב יותר לעלים של מרחב החיפוש ובכך להתקרב לתוצאה "אמיתית" במקום הסתמכות על היוריסטיקה. החסרונות הם שבמקרים בהם לא נצליח להגיע לעלים, נסתמך על ערכי יוריסטיקה פחות מיודעים ובכך נחזיר בחירות פחות אופטימליות בכל פעולה.
לעומת זאת, שימוש ביוריסטיקה מיודעת יותר אך יותר כבדה לחישוב יאפשר לנו פחות להתקרב לפתרונות ה"אמיתיים" בעלים, אך במקרים בהם לא נגיע לעלים נחזיר פתרונות יותר מיודעים שייגרמו לנו לבחור בפעולה יותר אופטימלית בכל שלב.

2. דנה טועה. אלגוריתם $Minimax$ אמנם יחזיר פתרון אופטימלי בהנחת יריב אופטימלי, אך לא מובטח כי זה הפתרון בעל מספר הצעדים הקטן ביותר.
למשל, בהינתן עץ החיפוש הבא (בהינתן שאם מספר בנים בעלי אותו ערך, נבחר בשמאלי מביניהם):



צומת ה-min תבחר בערך 5, ואז צומת ה-max תבחר גם כן בערך 5, אך הפעולה שתחזיר עבור המצב הנוכחי יגרום לנו להמשיך לשחק עוד תור, למרות שכבר יכולנו לסיים בתור הקודם עם אותו ערך תועלת.

3. במקרה שבו המשאבים שמגבילים אותנו הוא זמן ריצת האלגוריתם, נריץ את האלגוריתם המוגבל עומק: $RB - Minimax$ באופן איטרטיבי שבו בכל איטרציה נגדיל את עומק ההגבלה על החיפוש. כל עוד יש לנו זמן לריצת האלגוריתם ננסה להריץ את $RB - Minimax$ בעומק עמוק יותר. ברגע שנגמר הזמן, נחזיר את הפתרון הטוב ביותר שמצאנו עד לעומק הנוכחי.
אלגוריתם כפי שתיארנו שמשפר את ביצועיו ככל שיש לו יותר זמן נקרא:
$Anytime\ Contract\ Algorithms$.
אלגוריתם דומה שלמדנו עליו בקורס הוא למשל $\alpha\beta - Anytime$.

5. בהנחה שיש K סוכנים ולא 2, נערוך את השינויים הבאים:
   a. נשים לב שבמימוש מינימקס רגיל, ההנחה היא שהסוכן היריב מבצע מהלכים בצורה שתניב לסוכן שלנו את התועלת הנמוכה ביותר (מניחים שהיריב משחק בצורה הכי פסימית עבורנו, ולא הכי אופטימית עבורו למשל). לכן, אם נשנה את ההנחה שסוכן יריב מבצע החלטות על פי מיקסום התועלת האישית שלו, נרצה לשנות את הקוד בצורה שעבור כל סוכן במשחק תתבצע בחירת מהלך שתפיק עבורו את מירב התועלת. לכן נשתמש בפסאודו קוד הבא:

```
function minimax(state, agent)
    if G(state)
        return U(state, agent)
    turn = turn(state)
    children = succ(state)
    currMax = -inf
    for c in children
        v = minimax(c, turn)
        currMax = max(v, currMax)
    return currMax
```

b. נשים לב שתחת ההנחה שסוכן יריב מעוניין רק לפגוע בתועלת שלנו, משמע לעשות לה מינימזציה, אנחנו מגיעים למימוש הרגיל של מינימקס – לבחור תועלת מקסימלית תחת ההנחה שסוכן יריב מבצע מהלכים באופן שמקטין את התועלת שלנו במידה הרבה ביותר. בנוסף, הפסאודו קוד הנתון בתרגול לא מניח קיום של רק שני שחקנים ולכן נוכל להשתמש בו כמו שהוא:

```
function minimax(state, agent)
    if G(state)
        return U(state, agent)
    turn = turn(state)
    children = succ(state)
    if turn = agent
        currMax = -inf
        for c in children
            v = minimax(c, agent)
            currMax = max(v, currMax)
        return currMax
    else:
        currMin = inf
        for c in children
            v = minimax(c, agent)
            currMin = min(v, currMin)
        return currMin
```

c. בהנחה שכל סוכן רוצה שהסוכן שאחריו יקבל תועלת מקסימלית, נרצה שעבור הסוכן שלנו ההתנהגות תהיה זהה, אבל עבור סוכנים יריבים הם יבצעו מקסימיזציה של התועלת של הסוכן שתורו הבא. לכן נשתמש בפסאודו קוד הבא:

```
function minimax(state, agent)
    if G(state)
        return U(state, agent)
    turn = turn(state)
    children = succ(state)
    if turn = agent
        currMax = -inf
        for c in children
            v = minimax(c, agent)
            currMax = max(v, currMax)
        return currMax
    else:
        nextAgent = (turn + 1) % K
        currMax = -inf
        for c in children
            v = minimax(c, nextAgent)
            currMax = max(v, currMax)
        return currMax
```

חלק 3

2. מבחינת זמן ריצה, לשני הסוכנים יש אותה הגבלת זמן לתור ולכן שניהם ינצלו את כל הזמן העומד לרשותם לבחירת המהלך הבא, אך כיוון והסוכן $AlphaBeta$ "גוזם" בנים שלא משפרים, הוא יוכל לנצל את הזמן שהוא חוסך על מנת לחפש יותר לעומק.
מבחינת בחירת המהלך הבא לבצע, ייתכן שהסוכן $AlphaBeta$ יבחר במהלך אחר מכיוון והיה לו יותר זמן להעמיק בעץ (זאת כיוון והוא חוסך בדיקות מיותרות ולכן נשאר לו זמן רב יותר להעמיק בעץ) ולכן יוכל אולי למצוא מהלך טוב יותר ש-$Minimax$ לא מצא.

חלק 4

1. בהנחה שאנו משתמשים ב-$Expectimax$ כנגד סוכן רנדומלי לחלוטין, נבחר בהסתברות $\frac{1}{\#operations}$ לכל אחד מהפעולות של היריב. זאת מכיוון שהיריב בוחר באחת מהן באופן אחיד.

2. בהינתן שיוריסטיקה h מקיימת $\forall s \ -1 \leq h(s) \leq 1$. נניח שאנחנו מסתכלים על צומת מקסימום עם בנים הסתברותיים. בדומה ל$minimax$ נחשב את הבן ההסתברותי הראשון שלנו בצורה מלאה. לבנים ההסתברותיים הבאים, לכל פעולה אפשרית, נוכל לחשב את ערך התועלת המשוכללת שלה, להניח ששאר הפעולות האפשריות שלו מגיעות עם ערך תועלת מקסימלי (1), לחשב באופן זה את התוחלת של המצב ולבדוק: אם ערך התוחלת קטן או שווה מערך תוחלת שכבר מובטח לנו, נוכל לגזום את כל הענף של הצומת ההסתברותי הזה כי גם אם היא תקבל ערך מקסימלי עבור שאר הפעולות שלה – עדיין נוכל למקסם את הרווח שלנו על ידי פעולה אחרת.

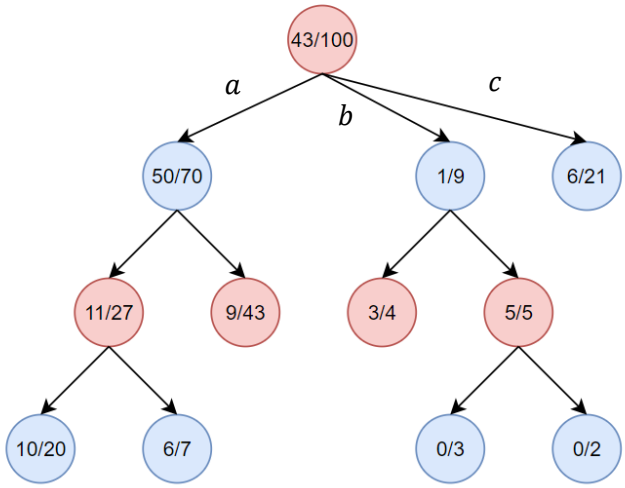
חלק 5

1. עבור השינויים האפשריים:
   a. הגדלת הלוח לגודל של $8x8$ לא תשנה את מקדם הסיעוף, מכיוון שבכל מצב מספר הפעולות שיכול לבצע הסוכן הן לכל היותר 7 – תנועה לכל אחד מהכיוונים, איסוף או הורדה של חבילה או תדלוק. הוספת מחסומים בלוח גם כן לא תשנה את מקדם הסיעוף, כיוון והוספתם רק תקטין את מספר הפעולות המקסימלי שניתן לבצע בכל מצב.
   b. הוספת הפעולה של הוספת בלוק במיקום כלשהו על הלוח תגדיל את מקדם הסיעוף בגודל הלוח, למשל עבור הלוח $5x5$ נקבל תוספת של לכל היותר 25 פחות מספר המשבצות התפוסות.
   זאת מכיוון ובכל תור מספר הפעולות האפשרי של סוכן עולה במספר המשבצות הפנויות בלוח.

2. בהנחה שמתבצע השינוי השני מהסעיף הקודם:
   a. כל האלגוריתמים שמימשנו, מחייבים באיזשהו שלב סריקה לרוחב של הפעולות האפשריות בכל צומת מסוים ולכן זמן הריצה שלהם תלוי במקדם הסיעוף (בהנחה שלא מגבילים את זמן החיפוש). לכן לכל האלגוריתמים שמימשנו ייקח זמן גדול מהותית להחזיר צעד, מאשר לריצה על המשחק לפני השינוי.
   b. על מנת להתמודד עם השינויים, שגרמו למקדם סיעוף גדול במיוחד נרצה להשתמש באלגוריתם $MCTS$. נבחר להשתמש בו מכיוון והוא מסוגל להתמודד טוב עם משחקים בעלי מקדם סיעוף גדול יחסית ולכן יוכל למצוא את הפעולה העדיפה לאור הסימלוצים שנבצע לתור זה.

חלק 6

1. העץ המלא הוא:



2. נחשב עבור הפעולה הראשונה לאיזה מהבנים של השורש נעדיף לפנות:

a. עבור a נקבל: $\frac{50}{70}+\sqrt{2}\cdot\sqrt{\frac{\ln(100)}{70}}=1.07702$

b. עבור b נקבל: $\frac{1}{9}+\sqrt{2}\cdot\sqrt{\frac{\ln(100)}{9}}=1.12272$

c. עבור c נקבל: $\frac{6}{21}+\sqrt{2}\cdot\sqrt{\frac{\ln(100)}{21}}=0.94797$

נעדיף לבחור בבן b, ולכן הצומת הבא שיפותח הוא צאצא של b.

3. בהנחה שהסימולציות הבאות נגמרות תמיד בניצחון של הכחול, נקבל שהערך המחושב עבור צאצאי b הולך וקטן, נחשב כמה תורות עד שזה יקרה:

תור ראשון:

עבור a נקבל: $\frac{50}{70}+\sqrt{2}\cdot\sqrt{\frac{\ln(101)}{70}}=1.07741$

עבור b נקבל: $\frac{1}{10}+\sqrt{2}\cdot\sqrt{\frac{\ln(101)}{10}}=1.06074$

עבור c נקבל: $\frac{6}{21}+\sqrt{2}\cdot\sqrt{\frac{\ln(101)}{21}}=0.94868$

מכאן ניתן להסיק שאחרי תור אחד שבו הכחול מנצח כבר נעדיף צאצא אחר של השורש.

חלק 7

1. הבעיה איתה מנסה להתמודד אלגוריתם מונטה קרלו היא המצב שבו לסוכן יש רק מידע חלקי על המצב של סוכנים אחרים במשחק. האלגוריתם מנסה להתמודד עם בעיה זו בכך שהוא דוגם מספר קבוע של "השלמות" של מצבי המשחק, ומחשב ממוצע של $alphabeta$ עבורם.

2. האלגוריתם משלים אינפורמציה שלא ידועה לו מתוך האפשרויות, ודוגם K השלמות כאלה, אותן הוא מנתח. עבור ערכי K גדולים, ייקח לאלגוריתם יותר זמן לרוץ, לכן מלכתחילה הוספנו את הפרמטר הזה, כי לפעמים ריצה על כל אפשרויות ההשלמה לא ריאלית. בסופו של דבר האלגוריתם מעריך את התוחלת על התועלת שצעד מסויים יביא לנו על פני כל המצבים האפשריים. מכיוון שכל השלמת מצב סבירה באותה מידה, ערכי K קטנים יקרבו בצורה פחות טובה את התוחלת של הצעד, ובכך האלגוריתם יקבל בחירה פחות מושכלת.

3. נציע מימוש לאלגוריתם $Expectimax$ שימדל מונטה קרלו מידע חלקי בכך שהצמתים עם המידע החלקי יהיו צמתי max, הבנים של צמתי max ייצגו פעולות אפשריות והם יהיו צמתים הסתברותיים. לכל צומת הסתברותי יהיו לכל היותר K בנים, כמספר ההשלמות האפשריות למצב מלא, שכל אחד מהם מייצג השלמת מצב מלא ואת התועלת של בחירת הצעד על ההשלמה הספציפית הזאת.
האלגוריתמים יהיו שקולים, מכיוון שעל פי ההנחה של דנה, מונטה קרלו יפתח את כל ההשלמות האפשריות לכל מצב חלקי. אופן הפעולה של האלגוריתם כאמור הוא לחשב תוחלת של צעד אפשרי על כל ההשלמות של המצבים החלקיים, וזה בדיוק מה שאנחנו ממדלים בעזרת האלגוריתם $expectimax$ המוצע.

4. נציע גרסת $Anytime$ לאלגוריתם שבה פרמטר השיפור הוא K מספר הדגימות שנשלים לכל פעולה בכל מצב. כל עוד יש לאלגוריתם זמן להמשיך לרוץ, נגדיל את K וננסה לבחור פעולה איכותית יותר:

```
function MSPI_anytime(partialState, Agent, D, time_limit):
    K = 3
    while not exceed time_limit:
        a = MoneteCarloPartialInformation(partialState, Agent, D, K)
        K++

    return a

function MoneteCarloPartialInformation(PartialState, Agent, D ,K):
    Actions = get all legal actions in ParialState
    S_Complete = all states consistent with PartialState
    Sample = Sample K random states from S_Complete
    Loop for a in Actions:
        Loop for s in Sample:
            v(a) = v(a) + RB-AlphaBeta(a(s), Agent, D, Alpha=-inf, Beta=inf)
        v(a) = v(a)/K
    select a with maximal v(a)
```