מבוא לבינה מלאכותית

236501

תרגיל בית 2

5/1/2020

מגישים:

רוני אנגלנדר 312168354

נדב אורזך 311549455

חלק ב'

2. השחקן GreedtAgent בודק עבור כל מהלך אפשרי שלו את כל המהלכים האפשריים של יריביו, ומבין כל השילובים האלו השחקן מכניס לרשימה את כל המהלכים בעלי הערך היוריסטי הגבוה ביותר. לאחר מכן בוחר רנדומלית מהלך מתוך רשימה זו.
הפונקציה היוריסטית בה השחקן משתמש מניחה כי בכל מהלך עתידי של השחקן יאכל פרי עד שנגמר מספר התורות במשחק או עד שייגמרו כל הפירות במשחק (סוכמת גם את אורכי הנחשים מכיוון שייתכן כי יהפכו לפירות אם ימותו). בנוסף, עבור כל צעד עתידי שיותר מתקדם מהצעד הנוכחי נקטין את המשקל של צעד זה בחישוב הפונקציה, מכיוון שהוא פחות וודאי. הפונקציה מחושבת באופן הבא – עבור מקדם הנחה נתון (a < 1) ועבור מספר צעדים אפשרי (k) הפונקציה מחשבת :

$$a + a^2 + a^3 + \cdots + a^k$$

הפונקציה מחזירה את תוצאת החישוב הנ"ל + אורך הנחש לאחר הצעד.
הפונקציה היוריסטית מאפשרת להימנע ממוות מכיוון שעבור צעד בו הנחש מת, הפונקציה מחזירה רק את אורך הנחש ללא התוספת העתידית. מכיוון שתוספת זו תמיד גדולה מ-0, במקרה של מוות תוצאת היוריסטיקה תהיה פחות עדיפה על מהלכים אחרים בהם הנחש לא מת, לכן לא תיבחר.

חלק ג'

1. נגדיר יוריסטיקה המתחשבת בערכים הבאים:

- snake_length = the length of the snake in the next state (considering eating fruits or dying)
- distance_from_closest_walls = calculates the distance from the head position of the snake to the closest horizonal wall and vertical wall. This value holds the sum distance from both walls.
- distance_from_closest_fruit = calculates the "Manhattan" distance from the head position of the snake to the closest fruit.
- closest_rival_snake = calculates the "Manhattan" distance from the head position of the snake to the position of each rival snake and chooses the minimal distance. Meaning, the distance from the closest point of the rival snake body to the head of the player snake. Then, if this distance is greater than the length from this closest point of the rival snake body to its tail, then by the time the player will potentially reach the rival snake, it won't be at this point anymore, so we don't take this rival snake into consideration.

הפונקציה היוריסטית מחזירה את המשוואה הבאה:

(w_of_len * snake_length) + (w_of_walls*distance_from_closest_walls) + (w_of_rival*closest_rival_snake) – (w_of_fruits*distance_from_closest_fruit)

כאשר הפרמטרים מהצורה *_W מציינים משקל מסוים אותו אנחנו מכפילים בערך המתאים עבור כל פרמטר על מנת לקבל את התוצאה הרצויה.

2. נתייחס עבור הערכים שתוארו לעיל ונסביר את המוטיבציה מאחוריהם:

- אורך הנחש (snake_length) – כאשר אנחנו מחשבים במהלך הבא של הנחש את אורכו אנחנו בפועל מייחסים חשיבות האם הנחש אכל פרי או לא. לכן אם במהלך הבא יש אפשרות לאכול פרי, הפונקציה היוריסטית נותנת למהלך זה משקל גדול יותר.
- מרחק מהקירות (distance_from_closest_walls) – המוטיבציה מאחורי ערך זה היא שככל שהנחש קרוב יותר לקיר יש פוטנציאל גדול יותר להיתקע ופחות מהלכים טובים אפשריים. לכן הפונקציה היוריסטית תתעדף מהלכים שמרחיקים את הנחש מהקיר.
- מרחק מהפרי הקרוב ביותר (distance_from_closest_fruit) - המוטיבציה מאחורי ערך זה היא שככל שהנחש קרוב יותר לפרי יש פוטנציאל גדול יותר לאכול אותו. לכן הפונקציה היוריסטית תתעדף מהלכים שמקרבים את הנחש לפירות.
- מרחק מהיריב הקרוב ביותר (closest_rival_snake) – המוטיבציה מאחורי ערך זה היא שכאשר הנחש נתקל ביריב הנחש מת, וככל שהוא קרוב יותר ליריב סיכויו להיתקל בו גבוהים יותר. הפונקציה היוריסטית לוקחת גם בחשבון את המרחק מהנקודה הקרובה ביותר של היריב אל ראש הנחש ביחס למרחק של אותה נקודה מהזנב של אותו נחש, מכיוון שכאשר הנחש יגיע לנקודה זו היריב כבר לא יהיה שם. לכן הפונקציה היוריסטית תתעדף מהלכים שמרחיקים את הנחש מהיריב.

כל הערכים (פרט לאורך הנחש) שציינו אינם נלקחים בחשבון ביורסטיקה של הנחש הפשוט, לכן לפי המוטיבציה שהסברנו אנחנו צופים כי הביצועים של השחקן החדש ישתפרו.

<u>חלק ד'</u>

2. אנחנו צופים כי אלגוריתם ה-MinMax שכתבנו יעבוד טוב בתרחישים בהם יש חשיבות להסתכלות מספר צעדים קדימה. לדוגמא, במצב בו ייתכן כי הסוכן ייכנס ללולאה סביב עצמו שבסופה ייחסם ויאלץ להרוג את עצמו, צפייה של מספר צעדים קדימה יכולה למנוע מהסוכן להכנס לבור זה. כמו כן ייתכן מצב בו אחד היריבים סוגר על הסוכן ויוכל להרוג אותו בעוד מספר מהלכים, ולכן גם כאן צפייה של מספר צעדים קדימה יכול לעזור לסוכן. לעומת זאת, הסתכלות קדימה גם יכולה לפגוע בפעולת הסוכן, למשל כאשר בצעד הבא קיימת אפשרות לסוכן לאכול פרי, אבל במספר צעדים קדימה תתכן אפשרות מנוונת לסוכן להיפגע, אפשרות שממנה הסוכן יוכל להימנע בכל מקרה, לכן במצב כזה פעולת הסוכן פחות טובה.
3. בהנחה שמספר היריבים הוא k לכל צומת מינימום יהיו 3^k בנים, זאת מכיוון שניתן להגדיר כל צומת של בן מינימום כווקטור מהצורה - $(a_1, a_2, \dots, a_k)$ כך ש - $a_i \in \{L, S, R\}$ כאשר k הוא מספר היריבים. כלומר עבור כל רמה בעץ ה-MinMax יפותחו 3^{dk} צמתים לכן סיבוכיות זמן הריצה תהיה $O(3^{dk})$ כאשר d הוא עומק העץ (העומק שניתן לאלגוריתם).עבור מספר רב (20+) של יריבים לא ניתן בפועל להשתמש באלגוריתם מכיוון שזמן החישוב של כל צעד גדול מאוד.
4. עבור המשחק שלנו כאשר יש יותר מיריב אחד, אנחנו מניחים כי פעולות כל היריבים יחד יזיקו הכי הרבה לסוכן שלנו. אך בפועל ייתכן כי כאשר כל יריב יבחר את המהלך הטוב ביותר בשבילו, סך הבחירות של כל היריבים של הסוכן שלנו יהיו יותר טובות מהמקרה הגרוע ביותר ואולי אף יעזרו לסוכן שלנו. לכן בבחירת המהלך הבא אנחנו מבצעים הנחות שאינן בהכרח נכונות.
5. נציע דרך אחרת לממש את האלגוריתם Minimax כך שיש שכבה נפרדת לכל יריב:

a. כל עומק בעץ מגדיר מהלך משחק של שחקן בודד. העומק הראשון יהיה פעולת הסוכן שלנו ובשונה מהמימוש בתרגיל שבו השכבה הבאה מייצגת את פעולות כל היריבים נגדיר כעת את השכבה הבאה לייצג פעולת יריב בודד. לשכבה הנ"ל יהיו שלוש אפשרויות שייצגו את שלושת המהלכים האפשריים של היריב כאשר לכל צומת יהיו שלושה בנים המייצגים את המהלכים האפשריים של היריב הבא וכך הלאה. כלומר,

לאחר המהלך של הסוכן קיימות k שכבות נוספות של צמתים, כאשר כל שכבה תייצג את אופן פעולות של יריב יחיד. כל שכבת יריב הינה צמתי MIN וכל שכבה מחושבת ביחס ללוח בתחילת התור (כלומר כל יריב לא לוקח בחשבון את מהלכי יריביו על מנת שהמהלכים יתבצעו בו"ז). נדגיש כי בכל תור השכבה הראשונה המייצגת את פעולת הסוכן היא MAX ואחריה k שכבות MIN רצופות המייצגות את פעולות היריבים, בניגוד לאלגוריתם המקורי בו כל שכבה אופי הצמתים מתחלף.
יתרונות: כמות הצמתים המפותחים במימוש זה זהה למימוש המקורי אך כעת לכל צומת יש שלושה בנים בניגוד למימוש המקורי שבו לצמתי MIN היו 3^k בנים ולכן נקבל למעשה עץ יותר עמוק ופחות רחב. לכן במימוש אלפא-ביתא הגיזום יהיה יותר אפקטיבי ונפתח במצטבר פחות צמתים.
חסרונות: סיבוכיות המקום של מימוש זה גדולה מהמימוש המקורי. לפי ההסבר לעיל אנו מפתחים עץ יותר עמוק, ולכן ריצת הרקורסיה שלנו תהיה יותר ארוכה ונצטרך לפתח יותר שכבות כל פעם.

b. ההנחה שלנו בנוגע לסדר קבלת ההחלטות של הסוכנים במשחק היא שכל הסוכנים מבצעים מהלך במקביל ללא תלות בסוכנים האחרים. במימוש המקורי הגדרנו מצב פנימי חדש על מנת לאפשר להנחה זו להתקיים. גם במימוש החדש אנחנו דואגים שהסוכנים יבצעו את המהלך על אותו לוח וללא תלות במהלך שמבצעים הסוכנים האחרים בפועל, אך למעשה כן קיימת תלות מאופן בניית העץ. כלומר, יריבים הנמצאים בשכבה גבוהה יותר בתוך אותו תור לוקחים בחשבון את המהלך הטוב ביותר של יריביהם בשכבות שמתחת.

חלק ה'

2. ההבדלים בין סוכן האלפא-בתא לסוכן Minimax:

א. סוכן האלפא-בטא מבחינת זמן הריצה צפוי לרוץ יותר מהר, מכיוון שלפי עקרון הגיזום שנלמד בשיעור האלגוריתם אינו מפתח תתי עצים שלא יילקחו בחישוב הכולל. לכן סך הפיתוחים בסוכן אלפא-בטא יותר קטן ולכן זמן הריצה הכולל קצר יותר ביחס ל-MiniMax הרגיל.

ב. סוכן האלפא-בטא מבחינת בחירת מהלכים צפוי להיות זהה לסוכן ה-MinMax הרגיל. זאת מכיוון שבריצת האלפא-בטא נעשה גיזום של מהלכים שלא ייבחרו בכל מקרה, כלומר התוצאה הסופית ששני הסוכנים השונים יחזירו למעשה תהיה זהה.

חלק ו'

1. במצב בו אנחנו צריכים לבחור מראש את כל N המהלכים של הסוכן שלנו, נצטרך כי מצב הלוח ההתחלתי ופעולות היריבים לכל אורך המשחק יהיו דטרמינסטים. זאת מכיוון שלצורך חישוב הפעולה ה-i אנחנו נצטרך לדעת את מצב הלוח והיריבים בתור זה. כלומר, על מנת שנוכל לבחור את סדרת N הפעולות הטובות ביותר נצטרך כי הסביבה של הסוכן תהיה ידועה. במידה וסביבה זו אכן הייתה משתנה היינו נתקלים בבעיה שבתכנון סדרת המהלכים היה חסר לנו מידע על מנת לבחור את המהלך הטוב ביותר לכל צעד.
2. המשחק אינו משחק סכום אפס. פונקציית התועלת של כל שחקן מחזירה את אורך הנחש + נקודה נוספת במידה והוא סיים את המשחק חי. כלומר הנקודה שנחש מקבל על היותו חי לא באה על חשבון נחש אחר, ובנוסף כל אכילת פרי שמגדילה את אורכו של הנחש לא באה על חשבונו של נחש אחר. כלומר סכימת הניקוד של כל השחקנים בסוף המשחק אינה אפס ולכן המשחק אינו סכום אפס.
3. כל מצב יהיה וקטור באורך N כאשר כל איבר בווקטור ייצג פנייה ימינה, שמאלה או המשך ישר: $(a_1, a_2, \dots, a_n)$ כך ש - $a_i \in \{L, S, R\}$. סך הכל הווקטור ייצג את בחירת הצעדים של הסוכן לאורך כל המשחק.

4. האופרטורים במרחב החיפוש יהיו בכל איטרציה החלפת איבר מסוים בווקטור ל-$\{L, S, R\}$. בכל איטרציה אנחנו נשתמש ב-3 אופרטורים עבור שלושת הצעדים האפשריים.
5. לפי האמור לעיל, כמות האיטרציות המבוצעות בחישוב זה הוא N. לכל איבר במערך נבצע איטרצית חישוב לבדוק מה הצעד העדיף מבחינת החיפוש המקומי, נבחר את הצעד המועדף ונעבור לאיטרציה הבאה. כלומר לאחר i איטרציות חושבו i הצעדים הראשונים.
6. נגדיר את הפונקציה היוריסטית להערכת טיב הפתרון כהפרש בין ניקוד הסופי של הסוכן לבין הניקוד הסופי של היריב בעל הניקוד הגבוה ביותר, כאשר נגדיר את הניקוד הסופי להיות אורך הנחש + נקודה אם הנחש חי בסוף המשחק. בעזרת פונקציה זו נוכל למדוד שיפור של מצב הסוכן בסוף המשחק ביחס ליריביו לאחר כל שינוי בכל איטרציה, ונשפר את רצף הפעולות בהתאם.
7. החלטנו לבחור מצב התחלתי בצורה דטרמיניסטית. תחילה הגדרנו מצב התחלתי "שמאל, ימין, שמאל, ימין...." מתוך הנחה שכך הנחש מגיע לאזורים נרחבים בלוח ויש לו פוטנציאל שיפור גדול בהרצת החיפוש הלוקלי. אך אחרי הרצת האלג', לא נראו שינויים משמעותיים מהוקטור ההתחלתי והחלטנו לנסות מצב התחלתי נוסף. לכן בחרנו וקטור שכולו מוגדר להיות "ישר" מתוך מחשבה כי גם בעזרת וקטור זה ניתן להגיע לאזורים נרחבים בלוח. בעזרת מצב התחלתי זה, הגענו לתוצאות טובות יותר ולכן הוחלט להשאירו.
8. באיטרציה ה-i בה אנחנו משנים את הפעולה ה-i של הסוכן אנחנו משפיעים למעשה על אותה פעולה ועל כל הפעולות שיגיעו אחריה, זאת מכיוון שכל פעולה משנה את כיוון התקדמותו של הסוכן ביחס למיקומו הנוכחי. לכן כל חישוב בהמשך התהליך יושפע ממקומו החדש. המצבים שנמצאים בתחום בין 0 ל- i-1 אינם מושפעים מהשינוי ה-i. לכן שינוי הפעולה ה-i משפיע על המצב הנוכחי ועוד 3^{N-i} מצבים שרק מהם ניתן לבחור, למשל עבור השינוי הראשון יושפעו לכל היותר 3^N מצבים.
10. תוצאת הריצה הייתה ניצחון לסוכן שהגיע לאורך של 13. וקטור הצעדים שהתקבל הוא:
[<GameAction.STRAIGHT: 1>, <GameAction.RIGHT: 2>, <GameAction.STRAIGHT: 1>, <GameAction.RIGHT: 2>, <GameAction.STRAIGHT: 1>, <GameAction.RIGHT: 2>, <GameAction.STRAIGHT: 1>, <GameAction.STRAIGHT: 1>, <GameAction.STRAIGHT: 1>, <GameAction.STRAIGHT: 1>, <GameAction.RIGHT: 2>, <GameAction.STRAIGHT: 1>, <GameAction.STRAIGHT: 1>, <GameAction.STRAIGHT: 1>, <GameAction.STRAIGHT: 1>, <GameAction.STRAIGHT: 1>, <GameAction.STRAIGHT: 1>, <GameAction.STRAIGHT: 1>, <GameAction.STRAIGHT: 1>, <GameAction.STRAIGHT: 1>, <GameAction.STRAIGHT: 1>, <GameAction.STRAIGHT: 1>, <GameAction.STRAIGHT: 1>, <GameAction.STRAIGHT: 1>, <GameAction.STRAIGHT: 1>, <GameAction.STRAIGHT: 1>, <GameAction.STRAIGHT: 1>, <GameAction.STRAIGHT: 1>, <GameAction.LEFT: 0>, <GameAction.STRAIGHT: 1>, <GameAction.STRAIGHT: 1>, <GameAction.STRAIGHT: 1>, <GameAction.STRAIGHT: 1>, <GameAction.STRAIGHT: 1>, <GameAction.STRAIGHT: 1>, <GameAction.LEFT: 0>, <GameAction.STRAIGHT: 1>, <GameAction.STRAIGHT: 1>, <GameAction.STRAIGHT: 1>, <GameAction.STRAIGHT: 1>, <GameAction.STRAIGHT: 1>, <GameAction.RIGHT: 2>, <GameAction.STRAIGHT: 1>, <GameAction.STRAIGHT: 1>, <GameAction.RIGHT: 2>, <GameAction.STRAIGHT: 1>, <GameAction.STRAIGHT: 1>, <GameAction.STRAIGHT: 1>]
11. האלגוריתם שבחרנו להשתמש בו הוא stochastic hill climbing שנלמד בהרצאה. כפי שצויין לעיל השינוי באיטרציה ה-i משפיע על מיקום הנחש לכל הפעולות שבאות לאחר מכן, וכתוצאה מכך משפיע על בחירת המצבים. לכן, בעזרת השימוש באלגוריתם stochastic hill climbing אנחנו מאפשרים באיטרציה ה-i לבצע בחירה שאינה השיפור המיטבי אך בשילוב

עם בחירות עתידות עשויה להוביל לתוצאה טובה יותר. לכן ניתן לצפות כי אלגוריתם זה ייצר תוצאה טובה יותר.

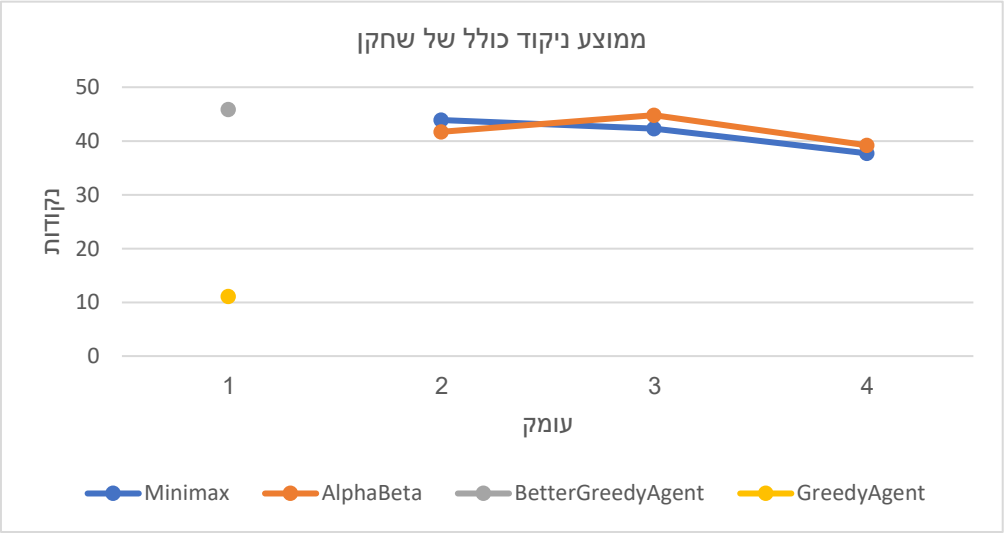
12. בדומה לאלגוריתם ה-SAHC, כל מצב יהיה וקטור באורך N כאשר כל איבר בווקטור ייצג פנייה ימינה, שמאלה או המשך ישר: $(a_1, a_2, \dots, a_n)$ כך ש - $a_i \in \{L, S, R\}$. סך הכל הווקטור ייצג את בחירת הצעדים של הסוכן לאורך כל המשחק. כמו כן, האופרטורים במרחב החיפוש יהיו בכל איטרציה החלפת איבר מסוים בווקטור ל-$\{L, S, R\}$.

14. תוצאת ההרצה הייתה ניצחון ואורך הסוכן היה 14. וקטור הפעולות שהתקבל הוא:

[<GameAction.STRAIGHT: 1>, <GameAction.RIGHT: 2>, <GameAction.STRAIGHT: 1>, <GameAction.RIGHT: 2>, <GameAction.STRAIGHT: 1>, <GameAction.RIGHT: 2>, <GameAction.STRAIGHT: 1>, <GameAction.STRAIGHT: 1>, <GameAction.STRAIGHT: 1>, <GameAction.STRAIGHT: 1>, <GameAction.RIGHT: 2>, <GameAction.STRAIGHT: 1>, <GameAction.STRAIGHT: 1>, <GameAction.STRAIGHT: 1>, <GameAction.STRAIGHT: 1>, <GameAction.STRAIGHT: 1>, <GameAction.STRAIGHT: 1>, <GameAction.STRAIGHT: 1>, <GameAction.STRAIGHT: 1>, <GameAction.STRAIGHT: 1>, <GameAction.STRAIGHT: 1>, <GameAction.STRAIGHT: 1>, <GameAction.STRAIGHT: 1>, <GameAction.STRAIGHT: 1>, <GameAction.STRAIGHT: 1>, <GameAction.STRAIGHT: 1>, <GameAction.LEFT: 0>, <GameAction.STRAIGHT: 1>, <GameAction.STRAIGHT: 1>, <GameAction.STRAIGHT: 1>, <GameAction.STRAIGHT: 1>, <GameAction.STRAIGHT: 1>, <GameAction.LEFT: 0>, <GameAction.STRAIGHT: 1>, <GameAction.STRAIGHT: 1>, <GameAction.STRAIGHT: 1>, <GameAction.STRAIGHT: 1>, <GameAction.STRAIGHT: 1>, <GameAction.RIGHT: 2>, <GameAction.STRAIGHT: 1>, <GameAction.STRAIGHT: 1>, <GameAction.RIGHT: 2>, <GameAction.STRAIGHT: 1>, <GameAction.STRAIGHT: 1>, <GameAction.STRAIGHT: 1>, <GameAction.RIGHT: 2>, <GameAction.STRAIGHT: 1>]

חלק ז'

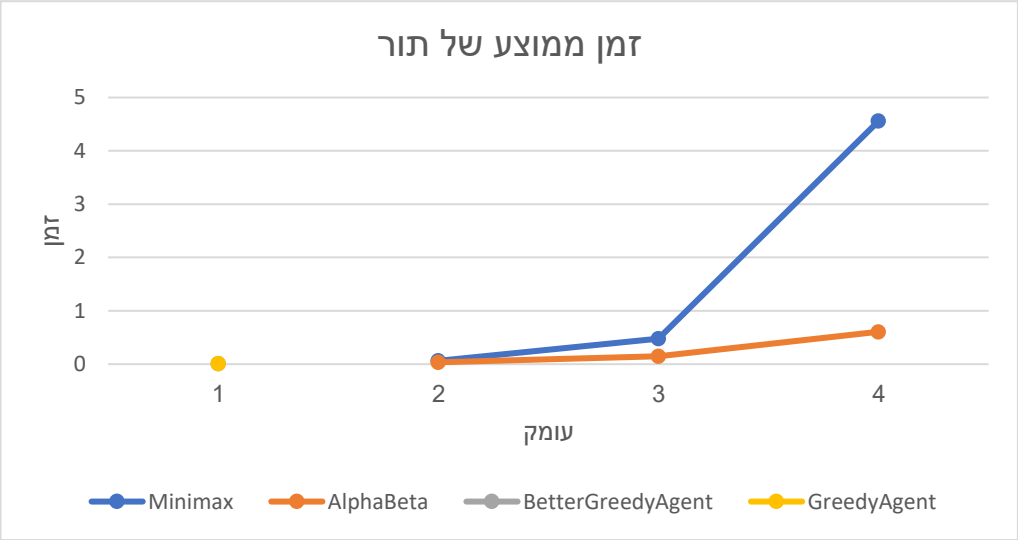
2. נציג גרף המייצג את ממוצע הניקוד הכולל עבור כל שחקן כתלות בעומק:



נציג את טבלת הנתונים המתאימה:

d = 1	d = 2	d = 3	d = 4	
11.1				**GreedyAgent**
45.9				**BetterGreedyAgent**
	43.9	42.3	37.7	**MinimaxAgent**
	41.7	44.8	39.2	**AlphBetaAgent**

3. ראשית ניתן לראות באופן מובהק כי ניקוד שלושת הסוכנים שמימשנו גבוה מאוד ביחס לסוכן החמדן הפשוט. כמו כן נשים לב כי גרף הניקוד של הסוכנים Minimax ו-AlphaBeta במגמת ירידה ככל שהעומק עולה, מה שבא בניגוד לציפיות שלנו. היינו מצפים כי אלגוריתם הבודק מספר גדול של צעדים קדימה וצופה מהלכים עתידיים של היריב יעבוד בצורה מוצלחת יותר לעומת אלגוריתם הבודק עומק קטן יותר. למרות האמור לעיל, נציין כי שלושת העומקים שנבדקו עבור הסוכנים הנ"ל נתנו תוצאות מסדר גודל דומה.
עוד גילוי שהגיע בניגוד לציפיות שלנו, הוא כי הסוכן בעל הניקוד הגבוה ביותר הוא הסוכן BetterGreedyAgent. אומנם שלושת הסוכנים משתמשים באותה יוריסטיקה שיצרנו אך שני הסוכנים Minimax ו-AlphaBeta בודקים מהלכים אפשריים עתידיים על מנת לקבל החלטה, לכן היינו מצפים כי ישיגו ניקוד גבוה יותר ביחס ל-BetterGreedyAgent, בניגוד למה שקורה בפועל.
4. נציג גרף המייצג את הזמן הממוצע שלקח כל תור עבור כל שחקן כתלות בעומק:



נציג את טבלת הנתונים המתאימה:

d = 1	d = 2	d = 3	d = 4	
0.0119				**GreedyAgent**
0.0104				**BetterGreedyAgent**
	0.0632	0.4779	4.5567	**MinimaxAgent**
	0.0351	0.1495	0.6071	**AlphBetaAgent**

5. ניתן לראות בהתאם לציפיותינו הזמן הממוצע לביצוע תור של הסוכן החמדן הפשוט והסוכן BetterGreedyAgent קצר משמעותית לעומת הסוכנים הרקורסיביים Minimax ו-AlphaBeta, זאת מכיוון שהאלגוריתם הרקורסיביים מבצעים מספר גדול יותר של חישובים, ביחס לעומק הרקורסיה בכל מהלך. כמו כן, נשים לב כי ככל שהעומק גדל כך גם זמן החישוב של הסוכנים הרקורסיביים עולה, זאת מכיוון שכפי שהסברנו כמות החישובים עולה כאשר העומק גדל.
דבר נוסף שניתן לראות בגרף הוא שריצת סוכן AlphaBeta מהירה יותר לעומת סוכן ה-Minimax, כאשר ההבדלים ביניהם גדלים ככול שהעומק גדל, בהתאם לציפיות שלנו. ניתן להסביר הבחנה זו על ידי כך שהסוכן AlphBeta מבצע גיזום לפי הנלמד בשיעור ולכן חוסך מספר רב של חישובים. גיזום זה נעשה משמעותי יותר ככל שעומק האלגוריתם גדל, בהתאם למה שרואים בגרף.
6. ראשית נציין כי היוריסטיקה שלנו גוברת על היוריסטיקה הפשוטה בכל המקרים בפער גדול, מכיוון שהיא יותר מיודעת ומתייחסת למספר רב יותר של פרמטרים במשחק.
בנוסף בשקלול התוצאות של משך תור ממוצע, ניקוד כולל ממוצע ומגבלת עומק ולפי האמור לעיל הסוכן BetterGreedyAgent הוא הסוכן הטוב ביותר מבין הארבעה. כפי שתיארנו, מעבר לכך שמשך תור ממוצע שלו נמוך בהרבה מהסוכנים הרקורסיביים וזהה לסוכן החמדן הפשוט, הוא גם השיג את הניקוד הגבוה ביותר בסיכום הריצות. בנוסף, כפי שכבר ציינו, כאשר שיחקנו נגד סוכן חמדן פשוט והגדרנו עומק הגדול מ-1 (למעשה אלו הסוכנים Minimax ו-AlphaBeta), לא שיפרנו את ביצועי הסוכן מבחינת ניקוד, ויתר על כך ככל שהגדלנו את העומק הניקוד ירד.
ננסה לנתח את התופעה המתוארת לפי מספר גורמים:
- ייתכן כי המשקלים שבחרנו לפרמטרים בחישוב היוריסטיקה ממקסמים את ביצועי הנחש החמדן כנגד הנחש הפשוט, אך בחישוב של מהלך עתידי חישוב זה מועיל פחות לסוכן החכם.
- בנוסף ייתכן כי הפרמטרים עצמם שבחרנו לתת עליהם דגש ביוריסטיקה שלנו כגון אורך הנחש והמרחק לפרי הקרוב ביותר, עוזרים לסוכן שאינו מסתכל מספר מהלכים קדימה.
- ייתכן כי הגדרת המשחק וצורת החישוב של ניקוד של שחקן ובחירתו כמנצח משרתת שחקן שאופן פעולתו הוא לאסוף כמה שיותר פירות וכמה שיותר מהר. לכן השחקן החמדן שמחפש את הדרך המהירה ביותר לאכול את הפרי הבא ושאינו מתחשב במהלכים עתידיים יפעל טוב יותר במסגרת המשחק הקיים.

למרות כל האמור לעיל, במידה ולא הייתה מגבלת עומק וזמן על ריצת הסוכנים הרקורסיביים אנו משערים כי במתן עומק גדול הסוכנים הרקורסיביים היו משיגים תוצאות טובות יותר ואולי אף מצליחים לעלות על סוכן ה-BetterGreedyAgent.

נציין כי כאשר הרצנו זוג סוכנים מיודעים אחד נגד השני, כאשר הראשון היה AlphaBeta והסוכן השני היה BetterGreedyAgent, ברוב המקרים היה זה סוכן ה-AlphaBeta שניצח. אנו מסבירים תופעה זו מכיוון שכאשר שני הסוכנים מיודעים יש חשיבות גדולה יותר לפעולות היריב ולכן יש יתרון לסוכן ה-AlphaBeta המסתכל קדימה מספר צעדים.

חלק ח'

שחקן התחרות שלנו יהיה סוכן ה-AlphaBeta. אנו מניחים כי יריבינו לסמסטר ישתמשו בסוכנים מיודעים ולכן לפי האבחנה מסעיף קודם נבחר בסוכן זה. תיאורו של הסוכן מוסבר בפירוט בחלקים ד' ו-ה'.