מבוא לבינה מלכותית – 236501

תרגיל בית 2

מאיה שטיין	315816686
נעם דובדבני	318765351

[CS - GR++] 236501 - Winter 2021-2022 - _____ 2 - 315816686-318765351

1. נסביר את יתרונות היוריסטיקה לעומת חסרונותיה:
   - חסרונות:
     - היורסטיקה מעדיפה יצירת טחנות חלקיות על פני יצירת טחנה מלאה. לדוגמה במצב בו ניתן ליצור בעזרת שחקן בודד טחנה חצי ריקה או טחנה מלאה, היורסטיקה תעדיף ליצור תחנה חלקית. דבר אשר יפגע בהתקדמות המשחק (השלמת תחנה לתחנה מלאה תאפשר הורדת שחקן של היריב, אשר מקדמת את המשחק ברוב המקרים לטובת הסוכן). בנוסף, מילוי של טחנה הינו מוריד את ערך מספר הטחנון החלקיות של הסוכן, דבר אשר נחשב פחות טוב מבחינת היורסטיקה.
   - יתרונות:
     - היוריסטקיה מנסה לחסום הצלחות של היריב ביצירת טחנה מלאה. לדוגמה במצב בו ישנו חייל בודד אשר יכול לחסום טחנה מלאה של היריב, היורסטיקה תעדיף זאת מכיוון שבכך היא מקטינה את מספר הטחנות החלקיות של היריב.
     - נשים לב כי היורסטיקה נותנת אינדיקציה טובה יחיסית לעומת מצב המשחק של הסוכן על היריב. כי מי שלישיה של שני חיילים ותא ריק ניתן להגיע בקלות לטחנה מלאה בפחות צעדים, דבר המקדם את המשחק לטובת הסוכן.

2. הגדרנו שתי יוריסטיקות שונות , אחת לכל שלב במשחק. נפרט תחילה על מרכיביי היורסטיקות ולאחר מכן נפרט על חלקותן לפי שלבים.
   - Closed_mil:
     - מקבלת לוח ואת הצעד האחרון שבוצע בו.
     - בודקת האם הסוכן יצר טחנה בצעד האחרון שבוצע. אם יצר, החזר 1, אם היריב יצר במהלך האחרון טחנה החזר 1-. במקרה בו לא נוצרה תחנה במהלך האחרון, מוחזר 0.
     - כאשר נסגרת טחנה, מצד אחד נחסמת האופצייה ליריב לנוע בשורת/עמודת התחנה, בנוסף סגירת הטחנה מאפשרת הורדת כלי של היריב. פעולה זו מקדמת את הסוכן לכיוון ניצחון.
   - _number_of_player_pieces_hur:
     - מקבלת את לוח המשחק הנוכחי.
     - מחזירה את ההפרש בין הכלים של הסוכן הנמצאים בלוח לבין הכלים של היריב הנמצאים על הלוח.
     - כאשר לסוכן יש יותר כלים על פני היריב, הוא יכול ליצור ולסגור יותר טחנות וגם יכול לחסום יותר כלים של היריב. בנוסף , על פי חוקי המשחק, כאשר לסוכן יש פחות מ3 כלים הוא מפסיד ולכן הגדלת הפרש זה תקדם את הסוכן לניצחון.
   - diff_mine_and_opponent_blocked_pieces_hur:
     - מקבלת את לוח המשחק הנוכחי.
     - מחזירה את ההפרש בין מספר הכלים החסומים של היריב לבין מספר הכלים החסומים של הסוכן.

[CS - GR++] 236501 - Winter 2021-2022 - _____ 2 - 315816686-318765351

- כאשר הסוכן חוסם כלים רבים יותר של היריב, הוא מונע מהיריב ליצור יותר טחנות. בנוסף, אם כל כלי היריב חסומים אז הסוכן מנצח. ולכן הגדלת הפרש זה תקדם את הסוכן לניצחון.

- number_of_incomplete_mils_hur:
  - מקבלת את לוח המשחק הנוכחי.
  - מחזירה את ההפרש בין מספר הטחנות החלקיות של הסוכן לבין מספר התחנות החלקיות של היריב.
  - כפי שהסברנו בשאלה 1 בחלק הייבש, יורסטיקה זו עוזרת לקדם את הסוכן לכוון ניצחון.
  - על מנת לפתור את הבעיה בה ניתנת העדפה ליצירת תחנות חלקיות על פני מלאות, נתנו משקל רב יותר לפונקצייה Closed_mil.
  - נשים לב כי פונקצייה זו כמעט ואינה רלוונטית בשלב 2 של המשחק, בו נעדיף לחסום את היריב מלנוע ולהשלים טחנות על פני ליצור טחנות חלקיות חדשות. לכן **לא נשתמש בפונקצייה זו ביורסטיקה של שלב 2**.

- difftence_between_number_of_mills_hur:
  - מקבלת את לוח המשחק הנוכחי.
  - מחזירה את ההפרש בין מספר הטחנות הנוכחי של הסוכן לבין מספר הטחנות הנוכחי של היריב.
  - ככל שלסוכן ישנם יותר טחנות, הוא מונע תזוזה של יותר חיילים של היריב. בנוסף , הוא מאפשר לעצמו את היכולת של פירוק והרכבה מחדש של הטחנה, דבר המאפשר להוריד חיילים נוספים של היריב. לכן ככל שהפרש זה גדול יותר, הסוכן יהיה קרוב יותר לניצחון.

- Goal:
  - מקבלת את לוח המשחק הנוכחי את התור של המשלק ואת השחקן אשר תורו לשחק.
  - מחזירה 1 האם הסוכן ניצח במשחק , 1- אם היריב ניצח במשחק ו0 אם אין הכרעה בלוח זה.
  - כאשר ישנו ניצחון מובטח במשחק , עלינו "לדחוף" את הסוכן לבצע את המהלכים אשר יובילו לניצחון. כאשר מובטח ניצחון של היריב, עלינו למנוע מהסוכן לצבע מהלכים אשר יובילו להפסדו. לכן ניתן **משקל מיריבי** לפונקצייה זו ביורסטיקה.
  - נשים לב כי לא ניתן לנצח בשלב ה-1 של המשחק ולכן **לא נשתמש בפונקצייה זו ביורסטיקה של שלב 1**.

- לסיכום זהו חישוב היורסטיקות לפי שלבי המשחק:
  - שלב ראשון של המשחק :

18 * closed_mill(state) + 1 * diff_mine_and_opponent_blocked_pieces(state)
+ 9 * number_of_player_pieces_hur(state) + 10 * number_of_incomplete_mils_hur(state)
+ 26*difftence_between_number_of_mills_hur(state)

  - שלב שני של המשחק:

14 * closed_mill(state) +10 * diff_mine_and_opponent_blocked_pieces(state)
+ 12* number_of_player_pieces_hur(state) + 1086* goal(state , turn, maximizing_player)
+ 46 difftence_between_number_of_mills_hur(state)

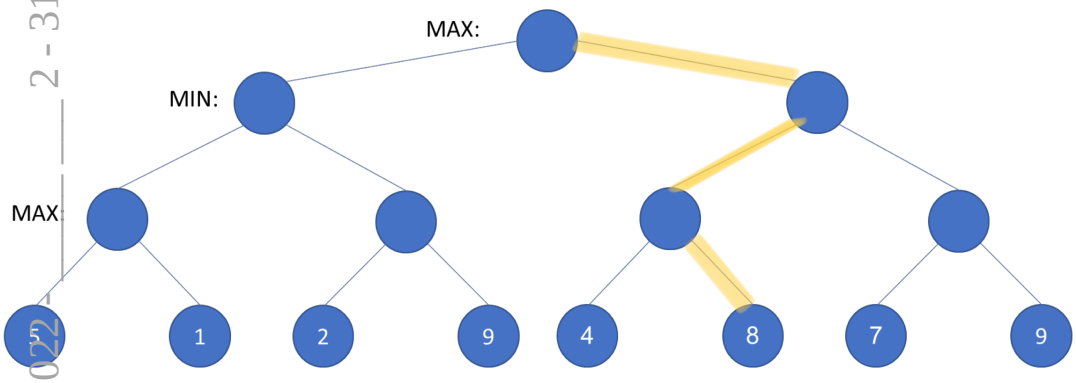
- הערה: המשקלים המדוייקים של היורסטיקה נקבעו על פי ניסוי הרצה ומחקר אינטרנטי.

3. .

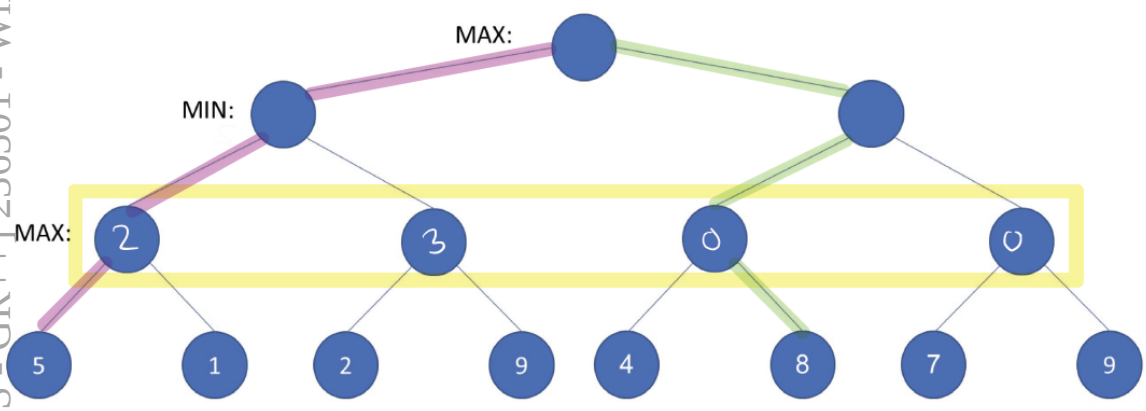
א. היתרון בהוספת גיזום אלפא בטא לאלגוריתם מיני מקס הוא : אפשור צמצום זמן החיפוש מכיוון שמאפשר לצמצם את כמות הצמתים אותם יש לפתח במהלך החיפוש. זה מתבצע על ידי קיצוץ ענפים שפיתוחם בהכרח לא ישנה את ערך המיני מקס , מסלול המיני מקס ואסטרטגית המיני מקס (נעביר כלפי מטה בקריאה הרקורסיבית שני חסמים ובהתאם להם נגזום). נציין כי במקרה ללא גיזום , ניתן להגיע לכמות הצמתים שיש לפתח הינם B^D ובמקרה הטוב של גיזום, ניתן להגיע לתוצאה של $B^{\frac{D}{2}}$ צמתים לפיתוח.

ב. על מנת לקבל גיזום מקסימאלי היינו ממינים את צמתי העץ באופן הבא: בנים של צמתי מקס (צמתי הסוכן) נסדר בסדר יורד וכך אם ערך המינימקס של אחד הבנים גדול מ b (המינמום מבין הערכים המובטחים לאבות קדמונים של צמתי min) נדע זאת לאחר פיתוח הבן הראשון ונגזום. בנים של צמתי min (היריב) נמיין בסדר עולה (מהסיבה ההפוכה).

ג. לא יתבצע גיזום כלל. המסלול האופטימלי להמשך המשחק בהינתן הידע שבעץ:



4. שקר. נראה דוגמה נגדית: נקבע את היורסטיקה הבאה: על העץ מסעיף ג. כאשר נסמן במרקר את היורסטיקה אשר מוגדרת לכל צומת ברמה השלישית.



על פי היורסטיקה שהגדרנו בדוגמה, בשלב הראשון יבחר השחקן בצומת השמאלית (מכיוון שעל פי היורסטיקה, בבחירת מסלול זה על פי אלגוריתם מינמקס הוא יקבל את התוצאה הגדולה ביותר שהיא 2). היריב במקרה הגרוע יבחר את הצומת השמאלית ברמה השנייה ואז יבחר הסוכן את העלה ה5

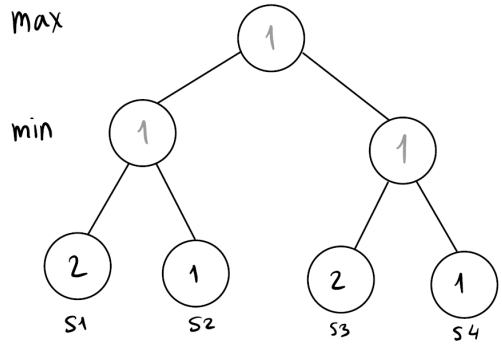
[CS - GR++] 236501 - Winter 2021-2022 - ____ 2 - 315816686-318765351

ויסיים את המשחק עם 5 נקודות (המסלול מסומן בסגול). כמו שראינו בסעיף הקודם, המסלול האופטימלי שאלגוריתם מינמקס יבחר בו , הוא המסלול המסומן בירוק וערך הנקודות שיוחזר ממנו הינו 8. לכן, האסטרטגיה אינה אופטימאלית.

.5

a. $U(s,k) = \begin{cases} 6\,, & P_G(s) \ \wedge k \ is \ winner \\ 6, & P_G(s) \wedge k \ is \ losser \\ Undefined, & Not \ P_G(s) \end{cases}$. נסביר: לכל מצב סופי s1 עד s4 נגדיר כי בהגעה למצב זה, כל שחקן יזכה ב6 נקודות (בין אם נגדיר כניצחון לשחקן 1 או בין אם נגדיר כהפסד, סכום הנקודות שייקבל כל שחקן הינו 6). זהו אינו משחק סכום אפס מכיוון שבכל מצב סופי הסכום אינו 0 (הוא 6 כפול מספר השחקנים). כל מצב סופי הינו מצב אופטימלי כיוון שבו כל שחקן מרוויח את סכום הנקודות המירבי. לכן האלגוריתם מינמקס אשר מחזיר מצב סופי מסויים הוא האופטימלי.

b. נגדיר $U(s,k) = \begin{cases} 2, & P_G(s) \ \wedge k \ is \ winner \\ 1, & P_G(s) \wedge k \ is \ losser \\ Undefined, & Not \ P_G(s) \end{cases}$ ונגדיר את המצבים s1,s3 להיות מצב ניצחון של הסוכן ו s2,s4 הפסד של הסוכן (ניצחון של היריב).



לפי עץ החיפוש, לא משנה מה יבחר הסוכן במצב ההתחלתי, תמיד יוחזר המצב הפחות אופטימלי (s2,s4) על פי אלגוריתם מינמקס. (מכיוון שלא משנה איזו בחירה תיבחר על ידי הסוכן בהתחלה, היריב יוכל לבחור את s2,s4 שהם המצב האופטימלי מבחינתו). נשים לב כי לא הוחזר מסלול אופטימלי באלגוריתם זה (s1,s3).

.6

a. נתבונן בעץ החיפוש הבא:
נתאר את המצב הבא: לאחר m צעדים, ניתן לנצח בצעד ה m+1 אם נפנה לעלה הימני, או לפנות לצומת השמאלית ולנצח במהלך ה m+2. במצב זה ,לאלגוריתם a,b אין תיעדוף איזה מסלול לבחור (מכיוון שאלגוריתם זה מחפש את התוצאה האופטימלית מבחינת הניקוד ולא מבחינת אורך המסלול) , לכן , ייתכן כי יבחר המסלול הארוך יותר. בקצרה: ישנו מצב בו מובטח ניצחון בצעד הבא, וגם בעוד x צעדים, ומבחינת האלגוריתם הם שקולים. לכן ייתכן כי האלגוריתם לא יבחר בניצחון בצעד הבא (אלא בהמשך).

b. השיפור אותו נבצע כדי להימנע ממצבים כאלה הוא שימוש ב - iterative deepening.
באמצעות שיפור זה , כאשר נגיע לפתרון אופטימלי שעומקו הוא מינימלי אז ניקח אותן, וגם אם יש פתרון אופטימלי (ניצחון). נוסף אך הוא עמוק יותר אז לא נגיע אליו כי נעצור לפני. (במצב הנוכחי אלגוריתם אלפא בטא מבצע את הסריקה שלו כמו dfs ולכן הוא יקח את הפתרון הראשון שהוא מוצא והוא לא בהכרח הקצר ביותר מה שהוביל למקרה המתואר).

7. נראה את פעולות ה Expectimax :

[CS - GR++] 236501 - Winter 2021-2022 - _____ 2 - 315816686-318765351

[CS - GR++] 236501 - Winter 2021-2022 - _____ 2 - 315816686-318765351

א.

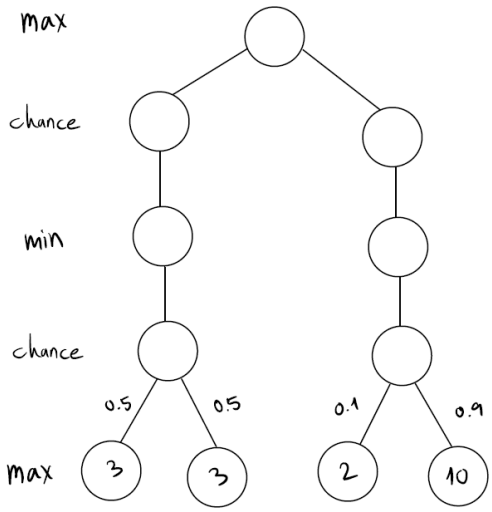
i. $V(B) = 0.3 * 5 + 0.7 * 1 = 2.2$

ii. $V(C) = 0.4 * 2 + 0.2 * 3 + 0.4 * 9 = 5$

iii. $V(D) = 0.1 * 4 + 0.9 * 7 = 6.7$

ב. פעולת ה max תבחר את הערך המקסימאלי שהוא a3.

ג. לא ניתן לגזום באותו אופן אשר ניתן לגזום את אלגוריתם אלפא בטא. ניתן דוגמה:



תבונן בתת העץ השמאלי, מכיוון שהתוכלת של העלים הינה 3 (0.5*3 +0.5*3) נחזיר כי ערך האלפא תרם הירידה לתת העץ הימני, ערך האלפא הינו 3.
נתבונן בתת העץ הימני, נראה כי העלה השמאלי הינו 2, לכן ערך העלה הימני קטן מאלפא ולכן נגזום את העלה השמאלי והערך שיוחזר מהאלגוריתם הינו 3.
אבל, במצב בו לא נגזום, נקבל כי ערך התוכלת של תת העץ הימני הינו 9.2 (0.9*10+ 0.1*2) ולכן על פי אלגוריתם expectimax הערך אשר יוחזר הינו 9.2. ולכן הגזימה פוגעת בנכונות האלגוריתם. זה נגרם כיוון שערך כל צומת (התוכלת שלו) תלוייה בערכי כל הבנים שלו וההסתברויות לקבל כל בן בניגוד לאלפא בטא, בו כל ערך של צומת תלוי ב min/max של הבנים שלו.

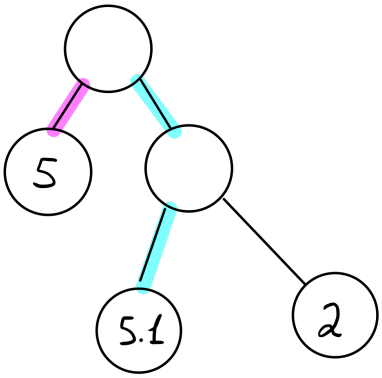
8.

א. נוסיף את השינויים הבאים לקוד:

a. בפונקציה maxValue יש להחליף את השורה if(value>= beta) בשורה if(value + epsilon>=beta).

b. בפונקציה minValue יש להחליף את השורה if(value<= alpha) בשורה if(value - epsilon<= alpha).

ב.



נבחר את אפסילון להיות שווה ל-1.

- באלגוריתם המקורי כאשר נגיע לעלה ה 5.1 , ערך האלפא יהיה 5. התנאי $value \leq alpha$ לא יתקיים ($5.1 \nless 5$) ולכן הגיזום של עלה 2 לא יתקיים. וייבדק גם העלה השני שהוא 2. התוצאה שתוחזר מאלגוריתם זה הינו 5.
- באגלוריתם החדש, כאשר נגיע לעלה ה 5.1 ערך האלפא יהיה 5. התנאי $value - \epsilon \leq alpha$ יתקיים $5.1 - 1 = 4.1 \leq 5$ ולכן יוחזר ערך ה value שהוא 5.1. (יגזם הענף הימני של הבן השני). ולכן, יוחזר הערך 5.1 מאלגוריתם זה.

9. .

א. נתבונן ב-2 מיקרים:

- במידה בו מקדם הסיעוף הינו קבוע, כל שימוש בפרוצדורה תחסוך מאתנו את פיתוח הרמות המייצגות את היריב בעץ החיפוש (זאת מכיוון שהפרוצדורה בהינתן מצב נתון מחזירה כיצד היריב אמור לשחק, ולכן אין צורך לפתח את כל אפשרויות המשחק שלו). לכן, בכל שימוש בפרוצדורה נפתח באותו זמן נתון פי -2 רמות. לדוגמה, על מנת לחזות 4 תורות קדימה, עלינו לפתח 4 רמות בעץ כאשר 4 תורות הן 2 תזוזות של היריב ו2 תזוזות של הסוכן. בעזרת הפרוצדורה, אין צורך לפתח את רמות היריב וניתן לפתח 4 רמות של תזוזות הסוכן ובכך להסתכל 8 מהלכים קדימה. לכן , נקבל כי העומק שחושב בזמן T בשימוש בפרוצדורה יהיה כפול מהעומק שחושב במקור. כלומר $d2 = 2 * d1$.
- אם מקדם הסיעוף אינו קבוע, אזי העומק החדש הוא קירוב ל $d2 = 2 * d1$ מכיוון ש כאשר מקדם הסיעוף גדל עם הזמן , העומק המקסיאמלי יצטמצם (יש יותר צמתים לפתח) לעומת מקדם סיעוף שקטן עם הזן , הגורם לעומק המקסימאלי לגדול (יש פחות צמתים לפתח אזי האלגוריתם יותר מהיר)

ב. כאשר העומק מוגבל, הערך המוחזר במינמקס ללא פרוצדורה קטן או שווה לערך המוחזר בעזרת הפרוצדורה. נסביר: כאשר מגיעים לצומת s , אלגוריתם מינמקס ללא הפרוצדורה מחזיר את הערך המקסימאלי **בהינתן והיריב בחר את הבחירות האופטימליות עבורו.** כלומר, אם היריב יבחר את הבחירות האופטימליות עבורו, סך הניקוד המקסימאלי הסופי שנוכל לקבל יקטן (משחק סכום 0). בשימוש בפרוצדורה, נקבל את המהלך אשר היריב יבצע . אם הוא יבצע רק מהלכים אופטימלים **עבורו** נקבל את אותה התוצאה כמו ללא שימוש בפרוצדורה. אם היריב יטעה, ויבחר מהלך שאינו אופטימלי **עבורו**, אזי נקבל כי התוצאה המקסימאלית אותה נוכל לקבל תגדל (כי הוא אינו בחר מהלך אופטימאלי ולכן התוצאה הסופית אותה יוכל לקבל קטנה).

10. .

א. הסטודנט לא בהכרח צודק. נתאר שני מצבים אשר הערך שיוחזר בהן אינו מקסימום גלובלי:

- ייתכן מצב בו ישנם 1000 מקסימומים לוקלים , כלומר נקודות אשר גבוהות משכניהם אבל אינם גבוהות מהמקסימום הגלובלי. לכן, ייתכן כי בכל 1000 הריצות נגיע לנקודת מקסימום

[CS - GR++] 236501 - Winter 2021-2022 - _____ _____ _____ 2 - 315816686-318765351

לוקלי ונחזיר את ערכה לפי האלגוריתם SAHC. אבל, כך לא בהכרח הגענו למקסימום הגלובלי ולכן התוצאה שתוחזר לא תהיה המקסימום בהכרח.

- ייתכן מצב בו רוב הנקודות הינם מישור, ולפי אלגוריתם SAHC כאשר נגיע למישור, נחזיר את התוצאה. לכן ייתכן שבכל 1000 הנקודות נפלנו בעיקר על מישורים דבר אשר גרם לאלגוריתם לא להחזיר בהכרח את המקסימום הגלובלי.

ב. היינו מעדיפים להשתמש ב SAHC with sideway steps בשילוב עם Simulated Annealing נסביר:

- שימוש ב Simulated Annealing ינסה למנוע מצב בו ניפול על מקסימומים לוקלים (בריחה מאדוות) על ידי כך שהוא מאפשר גם לקיחת צעדים שמורידים את התוצאה (בחירת שכן נמוך יותר) , בנוסף ככל שהערך אשר התקבל גדול יותר , כך הסיכוי לבחור את המקסימום הגלובלי עולה. ולבסוף, עצם העובדה שהוא מגריל נקודות ומריץ אותם בזה אחר זה תעלה את הסטטיסטיקה בליפול על נקודה שתוביל אותנו אל המקסימום המקומי.
- שימוש ב SAHC with sidewayמתיר צעדים השומרים על אותו ערך (ולכן זה טוב במקרים של בריחה ממישור למשל). שימוש באלגוריתם זה יכול לדייק את הביצועים מפני שבמקרה של קיום מישורים נוכל לקבל ערך גבוה יותר מזה שקיבל הסטודנט (ואם לא קיימים אז תתקבלנה אותן תוצאות כמו של הסטודנט). בשורה התחתונה, באמצעות שימוש באלגוריתם זה נוכל לוודא את תוצאות הסטודנט כי הוא יאפשר לנו למצוא מקסימומים (לוקליים\גלובליים) שהסטודנט היה עלול לפספס מחד, ומצד שני נקבל גם את כל מה שמצא הסטודנט.

ג. מרחב החיפוש שמתאים לדרישה:

ניתן לראות כי בסבירות גבוהה בכל אחת מהרצות האלגוריתם הסטודנט יתחיל מנקודה על המישור, ומאופן פעולת SAHC זה הערך שהוא יחזיר (כי אין שכן משפר, ועל פי אופן פעולת האלגוריתם הוא עוצר אם הגיע למצב מטרה או שאין שכן משפר כמו במקרה זה). לעומת זאת, SAHC with sideway step כן מתיר צעדים השומרים על אותו ערך ולכן בהסתברות גבוהה הוא "יתקדם" לאורך המישור ובסופו של דבר להגיע למקסימום כלומר לפתרון האופטימלי.

<u>חלק ה (רטוב)</u>

1. הגדרנו שתי יוריסטיקות שונות , אחת לכל שלב במשחק. נפרט תחילה על מרכיביי היורסטיקות ולאחר מכן נפרט על חלקותן לפי שלבים.

- Closed_mil:
  - מקבלת לוח ואת הצעד האחרון שבוצע בו.
  - בודקת האם הסוכן יצר טחנה בצעד האחרון שבוצע. אם יצר, החזר 1, אם היריב יצר במהלך האחרון טחנה החזר 1-. במקרה בו לא נוצרה תחנה במהלך האחרון, מוחזר 0.
  - כאשר נסגרת טחנה, מצד אחד נחסמת האופצייה ליריב לנוע בשורת/עמודת התחנה, בנוסף סגירת הטחנה מאפשרת הורדת כלי של היריב. פעולה זו מקדמת את הסוכן לכיוון ניצחון.

[CS - GR++] 236501 - Winter 2021-2022 - _____ 2 - 315816686-318765351

- _number_of_player_pieces_hur:
  - מקבלת את לוח המשחק הנוכחי.
  - מחזירה את ההפרש בין הכלים של הסוכן הנמצאים בלוח לבין הכלים של היריב הנמצאים על הלוח.
  - כאשר לסוכן יש יותר כלים על פני היריב, הוא יכול ליצור ולסגור יותר טחנות וגם יכול לחסום יותר כלים של היריב. בנוסף , על פי חוקי המשחק, כאשר לסוכן יש פחות מ3 כלים הוא מפסיד ולכן הגדלת הפרש זה תקדם את הסוכן לניצחון.

- diff_mine_and_opponent_blocked_pieces_hur:
  - מקבלת את לוח המשחק הנוכחי.
  - מחזירה את ההפרש בין מספר הכלים החסומים של היריב לבין מספר הכלים החסומים של הסוכן.
  - כאשר הסוכן חוסם כלים רבים יותר של היריב, הוא מונע מהיריב ליצור יותר טחנות. בנוסף, אם כל כלי היריב חסומים אז הסוכן מנצח. ולכן הגדלת הפרש זה תקדם את הסוכן לניצחון.
- number_of_incomplete_mils_hur:
  - מקבלת את לוח המשחק הנוכחי.
  - מחזירה את ההפרש בין מספר הטחנות החלקיות של הסוכן לבין מספר התחנות החלקיות של היריב.
  - כפי שהסברנו בשאלה 1 בחלק הייבש, יורסטיקה זו עוזרת לקדם את הסוכן לכוון ניצחון.
  - על מנת לפתור את הבעיה בה ניתנת העדפה ליצירת תחנות חלקיות על פני מלאות, נתנו משקל רב יותר לפונקצייה Closed_mil.
  - נשים לב כי פונקצייה זו כמעט ואינה רלוונטית בשלב 2 של המשחק, בו נעדיף לחסום את היריב מלנוע ולהשלים טחנות על פני ליצור טחנות חלקיות חדשות. לכן **לא נשתמש בפונקצייה זו ביורסטיקה של שלב 2**.

- difftence_between_number_of_mills_hur:
  - מקבלת את לוח המשחק הנוכחי.
  - מחזירה את ההפרש בין מספר הטחנות הנוכחי של הסוכן לבין מספר הטחנות הנוכחי של היריב.
  - ככל שלסוכן ישנם יותר טחנות, הוא מונע תזוזה של יותר חיילים של היריב. בנוסף , הוא מאפשר לעצמו את היכולת של פירוק והרכבה מחדש של הטחנה, דבר המאפשר להוריד חיילים נוספים של היריב. לכן ככל שהפרש זה גדול יותר, הסוכן יהיה קרוב יותר לניצחון.

- Goal:
  - מקבלת את לוח המשחק הנוכחי את התור של המשלק ואת השחקן אשר תורו לשחק.
  - מחזירה 1 האם הסוכן ניצח במשחק , 1- אם היריב ניצח במשחק ו0 אם אין הכרעה בלוח זה.
  - כאשר ישנו ניצחון מובטח במשחק , עלינו "לדחוף" את הסוכן לבצע את המהלכים אשר יובילו לניצחון. כאשר מובטח ניצחון של היריב, עלינו למנוע מהסוכן לצבע מהלכים אשר יובילו להפסדו. לכן ניתן **משקל מיריבי** לפונקצייה זו ביורסטיקה.

[CS - GR++] 236501 - Winter 2021-2022 - _____ ____ 2 - 315816686-318765351

- נשים לב כי לא ניתן לנצח בשלב ה-1 של המשחק ולכן **לא נשתמש בפונקצייה זו ביורסטיקה של שלב 1**.

- לסיכום זהו חישוב היורסטיקות לפי שלבי המשחק:
  - שלב ראשון של המשחק :

18 * closed_mill(state) + 1 * diff_mine_and_opponent_blocked_pieces(state)
+ 9 * number_of_player_pieces_hur(state) + 10 * number_of_incomplete_mils_hur(state)
+ 26*difftence_between_number_of_mills_hur(state)

  - שלב שני של המשחק:

14 * closed_mill(state) +10 * diff_mine_and_opponent_blocked_pieces(state)
+ 12* number_of_player_pieces_hur(state) + 1086* goal(state , turn, maximizing_player)
+ 46 difftence_between_number_of_mills_hur(state)

- הערה: המשקלים המדוייקים של היורסטיקה נקבעו על פי ניסוי הרצה ומחקר אינטרנטי.

2. בכדי לממש את שחקן התחרות, השתמשנו באותה היורסטיקה המצויינת בסעיף אחד. אך הוספנו לפונקצייה make_move את השינויים הבאים:

- הוספת צעדים ראשונים מובנים בתחילת כל משחק. ישנם מהלכי פתיחה "קלאסים" אשר יש מעניקים יתרון לשחקן (בהתאם אם הוא השחקן הראשון או השני). את מהלכי הפתיחה מימשנו בהתאם לתור. נתאר את מהלכי הריצה בחלוקה לשני מיקרים:
  - הסוכן הוא השחקן שתורו הוא ראשון: מטרתינו היא לשים שני חיילים בפינות מנוגדות של הלוח. כלומר המהלך הראשון יהיה לשים חייל בפינה ובמידה ובתור הבא של הסוכן הפינה הנגדית תהיה פנויה, נשים את החייל בה.
  - במידה והסוכן אינו השחקן הראשון, במידה והיריב שם חייל בפינה, על הסוכן לשים בתור הראשון שלו חייל בפינה הנגדית. אחרת ישים חייל בפינה פנוייה כלשהי.
- בשאר המקרים, נריץ את אלגוריתם אלפא בתא (או מינמאקס תלוי בסעיף) באופן איטרטיבי. כלומר, נריץ תחילה את האלגוריתם בחישוב עומק 1, לאחר מכן 2 והלאה עד סוף מגבלת הזמן. כאשר האלגוריתם מגיע אל הגבלת העומק ערך כל צומת מחושב לפי היורסטיקה. במידה ויוחזר מסלול מהאלגוריתם אשר ממנו נגיע למצב ניצחון , נפסיק את הריצה ונחזיר את המהלך הראשון במסלול זה.

3.

- **עבור מגבלת זמן מוגבלת לכל תור**: ניהלנו את make move באופן הבא: ביצענו ריצה איטרטיבית של אלגוריתם מינמקס בסדר עולה לפי הגבלת עומק (עומק 1 , 2 וכך הלאה). בכל פעם, נשמור את ערך המצב אליו כדאי לעבור בסוף האיטרציה ונעלה את העומק אליו ניתן לרוץ ב1. כאשר נגמרת מגבלת הזמן, נזרק execpt אשר עוצר את ריצת האלגוריתם ומחזיר את המצב ,האחרון שחושב, אליו כדאי לעבור(מהאיטרציה הקודמת). המצב שמוחזר הוא ה move שיש לבצע.
- **עבור מגבלת זמן גלובלית:**
  - לאחר מחקר אינטרנטי, הבנו כי המהלכים החשובים ביותר במשחק הינם המהלכים הראשונים לכן החלטנו לתת זמן יחסי גבוהה יותר למהלכים הראשונים על פני המהלכים המתקדמים. לכל מהלך נתנו זמן ריצה מקסימאלי של ¼ מזמן הריצה שנותר לנו.

[CS - GR++] 236501 - Winter 2021-2022 - ____ 2 - 315816686-318765351

- בחירת מהלכי הפתיחה הקבועים נועדה לחסוך את הרצת אלגוריתם החישוב (אלפא בטא) במהלכים הראשונים ובכך לחסוך זמן ריצה.
- מהזמן המוגבל לכל מהלך, ביצענו הרצה של אלגוריתם אלפא בטא באופן איטרטיבי (כפי שהוסבר בנקודה הקודמת) אך הפעם, מדדנו את זמן הלוקח לכל איטרציה לחשב את המהלך הקודם (נסמנו t). מכיוון שזמן החישוב של האיטרצייה הבאה (מכיוון שדורשת ירידת רמה נוספת) יהיה גדול מ t נבדוק את זמן הריצה הנותר לתור זה. במידה והזמן הנותר קטן מ t, נדע כי בכל מקרה לא נספיק להריץ את האיטרצייה הבאה ולכן נחזיר את תוצאת האיטרצייה הזו על מנת לחסוך זמן משחק גלובלי (בכך שנחסוך זמן חישוב מיותר).

<u>חלק ו (רטוב)</u>

4. הרצנו את השחקן AlphaBeta ו Minimax במגבלות זמן שונות לכל תור. כאשר מגבלת הזמן הייתה נמוכה (קטן מ 20 שניות), לא הייתה הכרעה חד משמעית בין שני השחקנים, אך כאשר הגדלנו את מגבלת הזמן (מעל 20 שניות) AlphaBeta ניצח. תוצאה זו מתאימה לציפיותינו מכיוון ש:

- כאשר מגבלת הזמן של כל תור גבוהה: אלגוריתם AlphaBeta מבצע קיצוץ של ענפים בעץ החיפוש. כלומר, הוא יודע לוותר על פיתוח ענפים אשר לא ישנו את תוצאות האלגוריתם minmax. דבר זה ייקצר את זמן החיפוש של כל איטרציה של שחקן AlphaBeta לעומת שחקן Minimax. בכך, יוכל השחקן AlphaBeta לחשב יותר רמות בעץ החיפוש לעומת השחקן Minimax במגבלת הזמן הנתון. דבר זה יאפשר לו "לחשוב" יותר מהלכים קדימה ולבחור את המהלך אשר ייקדם אותו בצורה טובה יותר במהלך המשחק מהשחקן Minimax. בזכות זאת, ציפינו כי השחקן AlphaBeta ינצח.
- כאשר מגבלת הזמן נמוכה, אזי הזמן שחוסך אלגוריתם ה AlphaBeta בגיזום הענפים אינו משמעותי מספיק. כלומר, חסיכת הזמן בגיזום לא מאפשרת גילוי רמה נוספת בעץ החיפוש של שחקן ה AlphaBeta לעומת שחקן ה Minmax. דבר שהופך את אלגוריתמי החיפוש שלהם לשווים ולכן תוצאת המשחק אינן ידועות.

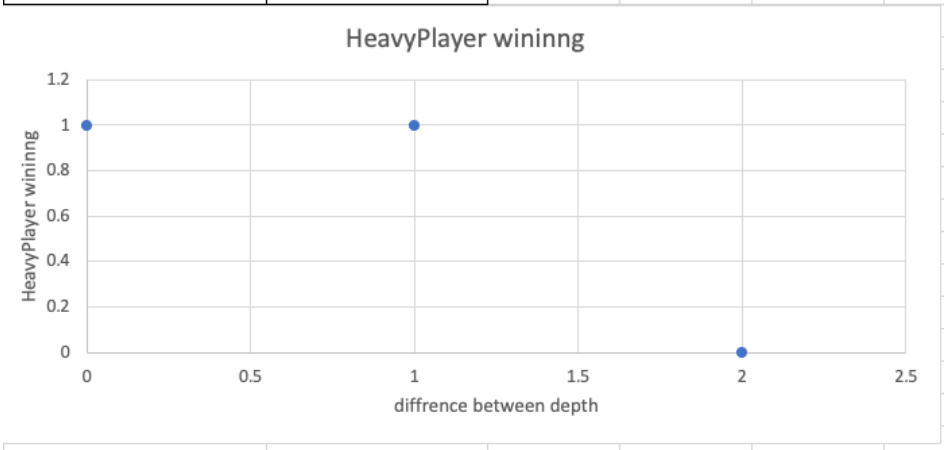
5. .

- הערה: בהרצות מסוימות בסעיף זה, השחקנים נכנסו למשחק אינסופי.
  - כיצד זה יכול לקרות? שחקן 1 רואה כי הפעולה הכי טובה לביצוע היא מעבר ממצב a למצב b , היריב מתגונן ועובד ממצב c למצב d ואז שחקן 1 רואה כי המהלך הטוב ביותר מבחינתו הוא לעבור חזרה למצב a ושחקן 2 עובר חזרה למצב c. מצב זה מתרחש רק במקרים בהם התור הינו מוגבל בזמן ולא המשחק.
  - לכן במצב זה, סימנו את מספר הניצחונות של HeavyPlayer ב- 0 , ניצחון כ -1 והפסד כ 1-.
- לגבי יוריסטיקה, השתמשנו עבר HeavyPlayer ביוריסטיקה בה השתמשנו עבור היוריסטיקה של השחקן אשר משתתף בתחרות (מוסברת בחלק ה') ועבור LightPlayer השתמשנו ביוריסטיקה אשר מחשבת את הפרש בין מספר חיילים אשר נותרו לסוכן במשחק לבין מספר החיילים אשר נותרו ליריב.

- כאשר heavy יורד לעומק 2 קבוע ו light לעומקים 2,3,4:

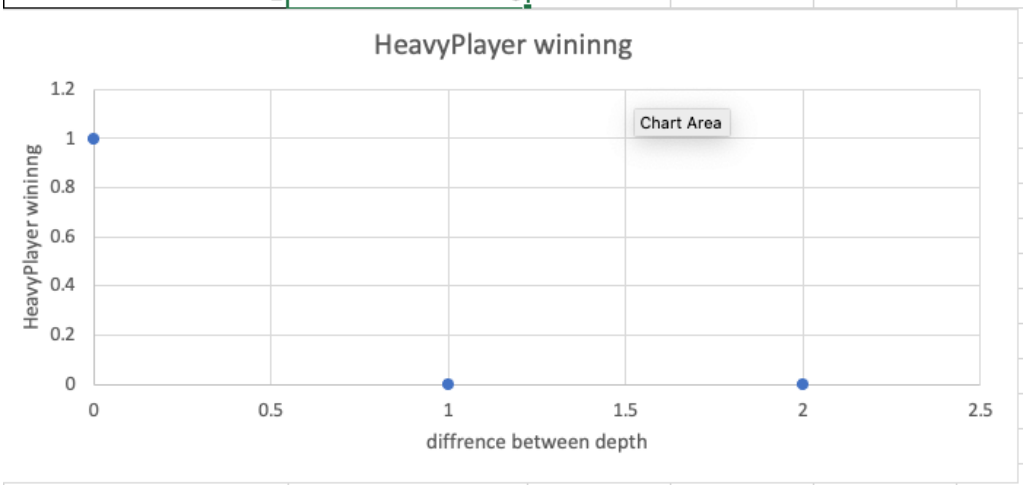
[CS - GR++] 236501 - Winter 2021-2022 - ___ 2 - 315816686-318765351

diffrence between depth	HeavyPlayer wininng
0	1
1	1
2	0



- כאשר heavy יורד לעומק 3 קבוע ו light לעומקים 3,4,5:

diffrence between depth	HeavyPlayer wininng
0	1
1	0
2	0



- תוצאות הניסוי:
  - באף ניסוי השחקן light אינו מנצח.
  - בניסוי בו heavy יורד לעומק 2 , השחקן heavy מנצח בשני השלבים הראשונים ובשלב האחרון נכנס ללולאה אינסופית של משחק עם ligth כפי שהוסבר בתחילת הסעיף.
  - בניסוי בו heavy יורד לעומק 3 , השחקן heavy מנצח רק בשלב ה -1. בשלב השני והאחרון המשחק נכנס ללולאה אינסופית כמו שהוסבר בתחילת הסעיף.
  - הסיבה ש heavy לא מפסיד בכל ניסוי היא בשל היורסטיקה (היורסטיקה של heavy משמעותית טובה יותר מהיורסטיקה של light דבר אשר "מכפר" על עומק הרמות שמוצאים בעץ החיפוש). אך, במקרה של לולאה אינסופית, נניח כי בעומק מסויים ligth

[CS - GR++] 236501 - Winter 2021-2022 - ____ ____ ____ 2 - 315816686-318765351

בוחר מהלך אשר גורר את שחקן heavy לבחור במהלך אחר שיוצר את הלולאה האינסופית. כאשר משתנה גודל העומק, יתכן מצב בו השחקן light יבחר מהלך אשר אומנם יותר טוב מבחינתו (כאשר עומק החיפוש גדול יותר, המהלך אשר יבחר באלגוריתם יקרב את השחקן לניצחון מהר יותר), אך במקרה זה , heavy יבחר מהלך אשר יוביל אותו לניצחון ולא מאפשר לlight להגיב בצורה שתכניס את המשחק ללולאה אינסופית (כמובן שמסיבה זו , שינוי העומק יכול לגרום לכך שניכנס למשחק אינסופי כאשר במצב הקודם המשחק נגמר). ולכן , התוצאות שונות בין הניסויים.

[CS - GR++] 236501 - Winter 2021-2022 - _____ ____ ____ 2 - 315816686-318765351