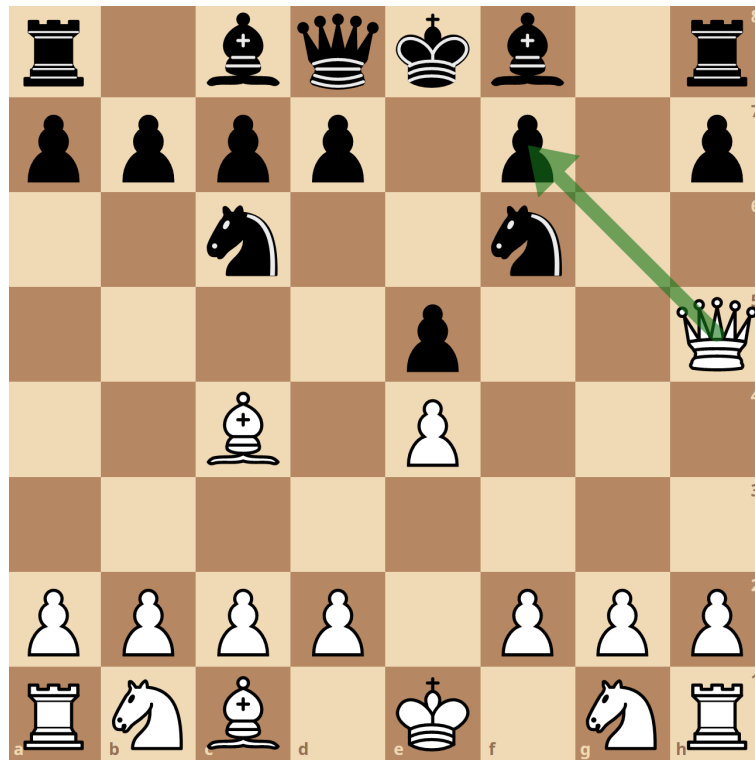


## Minimax Selective Deepening



## פרק 1 - מבוא: רקע כללי והצגת האלגוריתם:

### רקע על אלגוריתם המינימקס:

אלגוריתם המינימקס הוא אלגוריתם שבהינתן שחקן ומצב משחק, צריך להחזיר לשחקן את המהלך הטוב ביותר בו יכול השחקן לבחור. האלגוריתם הוא אלגוריתם רקורסיבי, שמפעיל את עצמו על כל המהלכים האפשריים ממצב מסויים. האלגוריתם הוא אלגוריתם כללי המתאים לכל משחק 2 שחקנים עם אינפורמציה מלאה (דמקה, איקס עיגול וכדומה...) וכמו כן, לאלגוריתם זה יש וריאציות למשחקים המערבים גורמים הסתברותיים כמו שש-בש וכו'.

האלגוריתם פועל באופן הבא:

האלגוריתם מקבל כקלט מצב המשחק  $S$  והשחקן שתורו לשחק  $P$ .

1. בדוק האם  $S$  הינו מצב סיום משחק.

1.1 החזר אינסוף אם השחקן מנצח.

1.2 החזר מינוס אינסוף אם היריב מנצח.

1.3 החזר 0 אם המשחק מסתיים בתיקו

2. אם תור השחקן:

2.1 לכל  $s$  באוסף המהלכים החוקיים אליהם אפשר להגיע מ  $S$  חשב מינימקס על  $s$  עם  $Next\ P$

והחזר את המקסימלי.

3. אם תור היריב:

3.1 לכל  $s$  באוסף המהלכים החוקיים אליהם אפשר להגיע מ  $S$  חשב מינימקס על  $s$  עם  $Next\ P$

והחזר את המינימלי.

אלגוריתם זה עובד נהדר על משחק כמו איקס עיגול, משום שכמות המהלכים האפשריים הוא קטן יחסית, אך במשחק כמו שחמט, מספר המצבים האפשריים הוא עצום ואינו מאפשר סיום האלגוריתם. כדי לפתור את הבעיה הזאת המציאו גרסה מוגבלת משאבים של מינימקס שמשתמשת בפונקציה היוריסטית כדי להעריך כמה המצב בזירת המשחק טוב עבור שחקן מסוים.

האלגוריתם מוגבל המשאבים עובד באופן דומה לאלגוריתם הקלאסי, רק שהפעם בנוסף מקבל כקלט עומק מקסימלי  $D$  ועושה שימוש בפונקציה היוריסטית  $H$ , והוא פועל באופן הבא:

1. בדוק האם  $S$  הינו מצב סיום משחק או אם  $D=0$ .

1.1 החזר הפעלת  $H$  על  $S, P$ .

2. אם תור השחקן:

2.1 לכל  $s$  באוסף המהלכים החוקיים אליהם אפשר להגיע מ  $S$  חשב מינימקס על  $s$  עם  $Next\ P$  ועם

$D-1$  והחזר את המקסימלי.

3. אם תור היריב:

3.1 לכל  $s$  באוסף המהלכים החוקיים אליהם אפשר להגיע מ  $S$  חשב מינימקס על  $s$  עם  $Next\ P$  ועם

$D-1$  והחזר את המינימלי.

את האלגוריתם הזה ניתן לייעל עוד בעזרת גיזום  $alpha\ beta$ . גיזום  $alpha\ beta$  בא לעזור לנו להימנע ממצב בו אנחנו משערכים מצבים שבטוח שלא יבחרו, בהינתן תתי עצים אחרים שכבר שוערכו על ידי המינימקס.

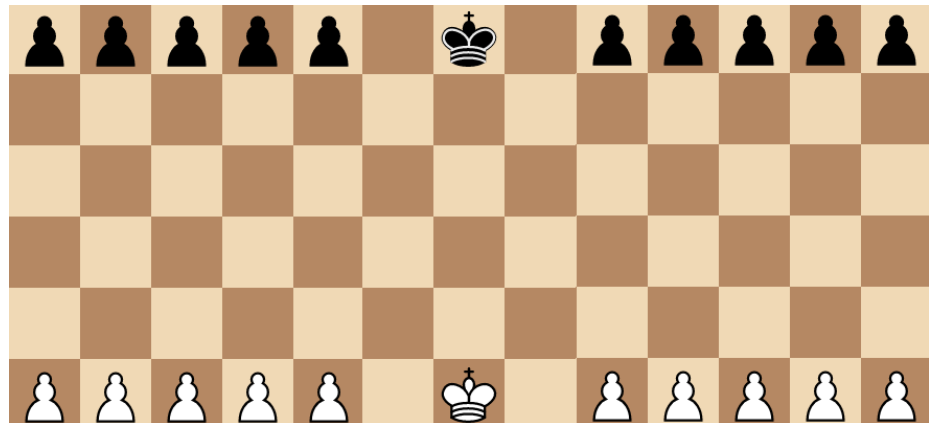
המינימקס הקלאסי הוא אלגוריתם מושלם מבחינת איכות הפתרון, משום שאם קיימת אפשרות לשחקן לנצח, אז אלגוריתם המינימקס ימצא את הדרך לניצחון. אלגוריתם המינימקס המוגבל עומק עם היוריסטיקה שחוזרת את ערך התועלת של המצב בצורה טובה ובדק את כל המצבים האפשריים לפיתוח עד עומק  $D$ , מביאה לאיכות פיתרון טובה ככל שהעומק  $D$  גדל.

עם זאת, לא כך עובד המוח האנושי - אמנם האדם השקול ירצה לשאוף לכך שצורת חשיבתו כשהוא מגיע לזירת המשחקים תהיה זהה לצורת החשיבה של המינימקס, אך בפועל מה שבסוף משפיע על השחקן האנושי לבחור מהלך הוא אינטואיציה המבוססת על ניסיון עבר ועל שקילות המהלכים הרלוונטיים ביותר לפי הבנתו.

אנחנו רואים את הפער בין צורת חשיבת המוח האנושי לבין אופן פעולת המינימקס מוגבל המשאבים, במשחקים המוגבלים בזמן לכל שחקן. הגבלת הזמן לא רק מגבילה את השחקן במספר הצעדים קדימה שאותם הוא שוקל, אלא גם את כמות המצבים שאותם הוא רוצה לקחת בחשבון בכל שלב בהעמקה. המינימקס מוגבל המשאבים מגביל את עצמו **בעומק הפיתוח**, ובכך שומר על "שלמות" הטקטיקה עד לעומק הרלוונטי. אבל מה יקרה אם נוותר על השלמות? האם כל מהלך שנבדק הוא רלוונטי? כמה זמן המינימקס מבזבז על בדיקת מהלכים סתמיים? לצורך המחשת חשיבות השאלות, נציג כעת משחק סתמי מבוסס שחמט.

### משחק סתמי מבוסס שחמט:

חוקי המשחק: לכל שחקן במשחק יש מלך שמטרתו להגיע לצד השני, 10 חיילים סתמיים וזמן מוגבל לכל תור.



מטרת המלך של הלבן להגיע למשבצת שמימין או משמאל למלך השחור, ומטרת המלך השחור להגיע למשבצת שמימין או משמאל למלך הלבן. המלך יכול לזוז צעד אחד בכל כיוון (כולל אלכסונים). במשחק הסתמי אין מצב של סילוק כלים. החיילים הפשוטים יכולים לזוז קדימה או אחורה מספר לא מוגבל של משבצות בכל תור, הם גם יכולים "לקפץ" מעל חייל סתמי של היריב. נשים לב שלפי חוקי המשחק הסתמי מי שמתחיל יכול לנצח בקלות (הוא הראשון לעשות צעד לעבר המטרה המרחק מהמטרה הינו 5 תורות לשני הצדדים). נסתכל על גודל עץ המינימקס של הצעד הראשון של הלבן (שכן הסוף ידוע מראש בגלל חוקי המשחק הסתמי). גודל עומק 1 של עץ המינימקס הוא כמספר המהלכים האפשריים.

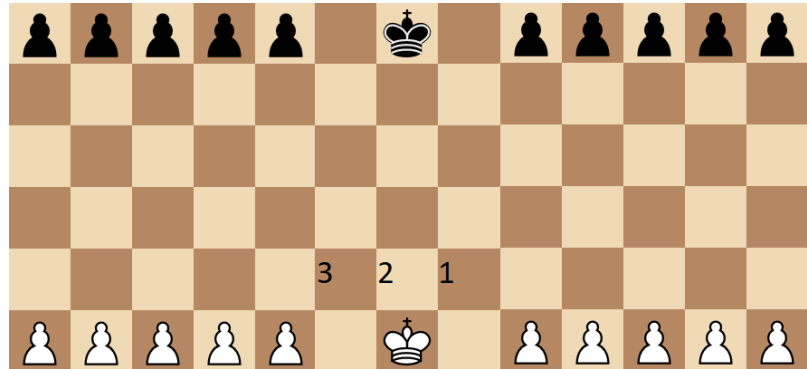
להלן תמונה הסופרת את מספר המהלכים האפשריים עבור תור ראשון של לבן. כל מספר זה מצב בפיתוח של עומק 1. (בכל עמודה אפשר להתקדם לכל אורכה עם חייל לבן, והמלך יכול להתקדם לכל הכיוונים צעד אחד).

1	5	9	13	17				26	30	34	38	42
2	6	10	14	18				27	31	35	39	43
3	7	11	15	19				28	32	36	40	44
4	8	12	16	20	22	23	24	29	33	37	41	45

קיבלנו שלעץ המינימקס (שסך הכל אומר שמי שמנצח זה מי שמתחיל) יש מקדם סיעוף של כ-45! (לאחר תנועת המלך נפתחת האפשרות ללכת אחורה והסיעוף יעלה ל-48) היכול להיות שעל משחק כה סתמי המוגבל בזמן, לא נוכל להפעיל מינימקס מוגבל לעומק הגדול מ-4, מבלי שנאבד את כל משאב הזמן שלנו?

הבעיה של המינימקס מוגבל המשאבים שאנחנו מעלים בפרויקט הזה היא שהאלגוריתם הקיים עובד עם היוריסטיקה אחת המעריכה את מצב המשחק בעוד D מהלכים, כשהיוריסטיקה הזו בעצם "מבטיחה" משהו על עומק החיפוש - היא מבטיחה שבהינתן עומק קבוע, אם נבדוק את כל הכלים נגיע ל-Reward ההיוריסטי המדובר. אבל מה אם נעשה היוריסטיקה נוספת שמנסה לחסום רוחב של חיפוש? למשל, במצב ההתחלתי של המשחק הסתמי ממקודם, ההיוריסטיקה החדשה תתן למהלכי מלך קדימה ערך גבוה, ולשאר המהלכים 0, ובכך נבחר להעמיק רק את המהלכים המקדמים את המלך למטרה - כלומר, נעמיק את ח המהלכים בעלי הערך הגבוהה ביותר לפי היוריסטיקה החדשה, המופעלת לכל מצב שאותו נשקול אם לפתח.

להלן תמונה הסופרת את מספר המהלכים האפשריים עבור תור ראשון של לבן, עבור  $n=3$ .  
(ה-3 צעדים הטובים ביותר להעמקה - בעלי ערך היוריסטי גבוהה ביותר מבין ה-49).



במצב כזה ניתן להעמיק הרבה יותר, שמקדם הסיעוף של העץ ירד מ-45 ל-3.

נשים לב כי לאלגוריתם עם אופטימיזציית הגיזום - rb-alphabeta, קיימות שיטות שונות להפחתת כמות הילדים שנפתחו והן מבוססות על סידור ילדים היוריסטי, שעם היוריסטיקה מספיק טובה לסידור, יכולים לגרום לגיזום ממש טוב של הענפים ובכך אף ניתן להעמיק (תחת אותה מגבלת זמן) לעומק גדול יותר. אך סידור ילדים לא מסנן מהלכים באופן מיידי, שכן הוא ימשיך להעמיק את כל המהלכים עד שיימצא אפשרות לגיזום - כלומר במקרה הגרוע ביותר (היוריסטיקה גרועה) הוא שקול לחלוטין בזמן ל-rb-alphabeta ללא סידור ילדים. אנחנו מציעים וריאנט שעובד גם לאלגוריתם ה-rb-minimax וגם ל-rb-alphabeta, שתמיד יבטיח שיפור בזמן משמעותי (כתלות במספר המצבים שנשאיר) והוא וריאנט ה-Minimax Selective Deepening.

### **Minimax Selective Deepening**

כעת נציג את השינוי שלנו באלגוריתמים של המינימקס, וריאנט ה-MSD (Minimax Selective Deepening) הסבר כללי על אופן פעולת הוריאנט:

עבור סוכן המשתמש באלגוריתם ה-rb-minimax או rb-alphabeta עם וריאנט ה-MSD, בכל עומק של האלגוריתם הרקורסיבי, לאחר שמפתחים את הילדים של המצב הנוכחי על ידי פונקציית ה-Succ, האלגוריתם יבחר לפתח מתוכם רק את המהלכים בעלי הפוטנציאל הגדול ביותר (לפי היוריסטיקה מתאימה) - כלומר יבצע סינון (filter) של children בשיעור מסוים שנקבע מראש.

משמעות "העמקה סלקטיבית", היא שגודל ההעמקה נשאר אותו הדבר, כלומר, עד לעומק מוגבל D (או עד למצב סופי) אך מספר הילדים בכל שלב בעץ קטן על ידי הסינון. בנוסף, הסינון מתבצע גם ל-children של צמתים שלנו וגם ל-children של צמתי היריב, זאת משום שאנחנו מניחים שהצעדים אותם היריב יבחר בסופו של דבר לרעתנו, יהיו הצעדים אותם בחרנו שלא לסנן (שכן אנחנו תמיד נחפש את המקרה הגרוע בו היריב יבחר).

השאיפה הכללית היא לגרום לכך שאם יש לנו למשל, יריב מסוג rb-alphabeta שמבצע העמקה לעומק D בזמן T, אז הסוכן rb-alphabeta עם וריאנט ה-MSD, יעמיק לעומק D+1 בזמן קרוב מאוד ל-T ובכך ירוויח ראייה עמוקה יותר תחת אותה מגבלת זמן וכל זאת ללא פגיעה באיכות האסטרטגיה של הסוכן שיכולה להיפגע בשל סינון המצבים - כלומר, שתחת אותה מגבלת זמן עם עומק גדול יותר ועם סינון מסוים של מצבים, יגרום לסוכן טוב יותר מיריבו בעל עומק D.

### rb-minimax עם וריאנט ה-Minimax Selective Deepening:

```
RB-Minimax(State, DecidingAgent, D, KeepRate)
  If G(State) then return U(State, DecidingAgent)
  If D=0 then return h(State, Agent)
  AgentToMove ← Turn(State)
  Children ← Succ(State, AgentToMove)
  Children ← Filter(State, Children, KeepRate)
  If AgentToMove = DecidingAgent then
    CurMax ←  $-\infty$ 
    Loop for c in Children
      v ← RB-Minimax(c, DecidingAgent, D-1)
      CurMax ← Max(v, CurMax)
    Return(CurMax)
  else ; AgentToMove ≠ DecidingAgent
    CurMin ←  $\infty$ 
    Loop for c in Children
      v ← RB-Minimax(c, DecidingAgent, D-1)
      CurMin ← Min(v, CurMin)
    Return(CurMin)
```

### rb-alphabeta עם וריאנט ה-Minimax Selective Deepening:

```
RB-AlphaBeta(State, DecidingAgent, D, Alpha, Beta, KeepRate)
  If G(State) then return U(State, DecidingAgent)
  If D=0 then return h(State, Agent)
  AgentToMove ← Turn(State)
  Children ← Succ(State, AgentToMove)
  Children ← Filter(State, Children, KeepRate)
  If AgentToMove = DecidingAgent then
    CurMax ←  $-\infty$ 
    Loop for c in Children
      v ← RB-AlphaBeta(c, DecidingAgent, D-1, Alpha, Beta)
      CurMax ← Max(v, CurMax)
      Alpha ← Max(CurMax, Alpha)
      If CurMax ≥ Beta then return  $\infty$ 
    Return CurMax
  else ; AgentToMove ≠ DecidingAgent
    CurMin ←  $\infty$ 
    Loop for c in Children
      v ← RB-AlphaBeta(c, DecidingAgent, D-1, Alpha, Beta)
      CurMin ← Min(v, CurMin)
      Beta ← Min(CurMin, Beta)
      If CurMin ≤ Alpha then return  $-\infty$ 
    Return CurMin
```

התוספת ל-2 האלגוריתמים מוגבלי המשאבים היא זהה, כאשר השוני היחידי מ-2 האלגוריתמים המקוריים, הוא שלאחר קבלת ה-children מפונקציית ה-Succ, נבצע פונקציית Filter שתבצע את MSD עבור הילדים (המצבים הבאים לפיתוח) של המצב הנוכחי. הפרמטר **KeepRate** קובע איזה שיעור מהילדים (המצבים הבאים לפיתוח) נרצה להשאיר ולהעמיק בהם. למשל, KeepRate = 65%, משמעו **להשאיר** 65% מהילדים הטובים ביותר ולנפות 35% מהם.

### פונקציית ה-Filter:

```
Filter(State, Children, KeepRate)
  ChildrenSorted ← SortChildren(State, Children)
  Children ← CutChildren(Children, ChildrenSorted, KeepRate)
  Return Children
```

פונקציית ה-Filter תסנן את הילדים ב-2 שלבים:

- SortChildren תסדר את children לפי סידור היוריסטי בסדר יורד, כלומר המהלך החשוב ביותר יהיה הראשון ברשימת הילדים החדשה והממויינת שתוחזר.
- CutChildren תחתוך את הרשימה הממויינת ChildrenSorted לפי ה-KeepRate, על ידי חיתוך של n הילדים הראשונים ברשימה, כש-n הוא האינדקס ממנו יש לחתוך (כלומר, עבור ה-KeepRate = 50% ורשימה בגודל 8 יישארו 4 הילדים הראשונים ו-n=4). לאחר מכן, תעבור על n האיברים שנותרו ברשימה זו ותסיר אותם מהעתק של children ותחזיר ה-children החתוך (ובכך תשמור על הסדר של האיברים המקורי של children). כלומר, לאחר ביצוע פונקציה זו, הרשימה children לא תהיה רשימה ממויינת של מצבים, אלא תשמור על הסדר המקורי, זאת משום שאיננו רוצים לערבב את שיטת סידור ילדים לגיזום יעיל של rb-alphabeta כחלק מהאלגוריתם (ראה דוגמא בהמשך).

פונקציית הסידור SortChildren:

```
SortChildren(State, Children)
  NewChildren ← {}
  Loop for c in Children
    NewChildren[c] ← H_Deepening(c)
  Children ← SortByValue(Children, NewChildren)
  Return Children
```

למעשה, אנחנו נקרא לפונקציה היוריסטית החדשה לסידור, כדי לבדוק את ערכו של כל ילד ברשימת הילדים ולאחר מכן נבצע SortByValue, שתקבל Children ומיפוי של כל ילד לערכו ותמיינן את רשימת הילדים. כלומר, עבור ילדים: [a,b,c,d] והפעלה של היוריסטיקה: [H\_Deepening(a),H\_Deepening(b),H\_Deepening(c),H\_Deepening(d)] נקבל ערך לכל ילד ונוכל לבצע מיון מערך רגיל לפי מיפוי זה. במקרה של תיקו בערך היוריסטי בין 2 מצבים, נבחר רנדומלית את המיקום שלהם ברשימה הממויינת.

לדוגמא, הפעלת filter:

בקריאה ל-rb-alphabeta בעומק כלשהו ורשימת children שקיבלנו מ-Succ בגודל 6: [s1,s2,s3,s4,s5,s6], עם פרמטר KeepRate = 0.5 (השאר 3 מצבים), כשהמצבים s2 ו-s4 הם המצבים שנרצה להעמיק, משום ש:  $H\_D(s4) > H\_D(s6) > H\_D(s2) > H\_D(s1) > H\_D(s3) > H\_D(s5)$

תא:

$children = [s1,s2,s3,s4,s5,s6] \rightarrow filter \rightarrow children = [s2,s4,s6]$

(נשים לב, שהרשימה החדשה שומרת על הסדר המקורי שלה מה-Succ ואין סידור ילדים)

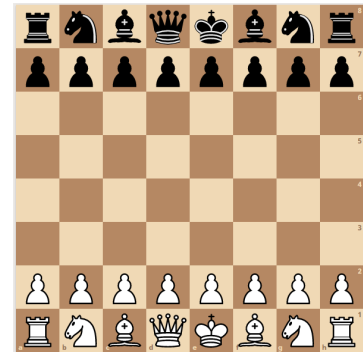
**משחק השחמט:**

משחק השחמט נחשב לאחד המשחקים האסטרטגיים המורכבים ביותר בתרבות האנושית. במשחק 2 שחקנים המשחקים האחד כנגד השני, כל אחד בתורו, כשלכל אחד מגבלת זמן כוללת ידועה מראש (כלומר, בכל תור של שחקן, הזמן הכולל שלו יורד).

### חוקי המשחק:

משחק השחמט משוחק על לוח משבצות 8 על 8. כל שחקן מתחיל עם שמונה חיילים פשוטים (pawn) שני צריחים (rook) שני פרשים (knight) שני רצים (bishop) מלכה אחת ומלך אחד.

סידור המשחק ההתחלתי:



### תנועת הכלים:

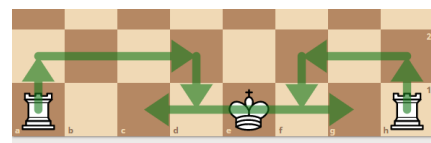
החייל יכול לזוז צעד אחד קדימה. הצריח יכול לזוז לכל האורך ולכל הרוחב (מהמשבצת עליה הוא עומד). הפרש זז שני צעדים לאורך וצעד אחד לרוחב או שני צעדים לרוחב וצעד אחד לאורך. הרץ זז באלכסון (לכל אורך האלכסונים). המלכה יכולה לזוז לאורך ולרוחב (כמו הצריח) וגם יכולה לזוז לאורך האלכסונים (כמו הרצים). המלך זז צעד אחד לכל כיוון שיבחר.

### הכאה:

כאשר כלי של שחקן זז לעבר משבצת עליה עומד כלי של היריב, הכלי של היריב העומד על המשבצת יוצא מן המשחק. מצב זה מכונה "הכאה". החייל (רגלי) מכה באלכסון למרות שזז צעד אחד קדימה (אם שני חיילים עומדים זה מול זה הם לא יכולים להכות האחד בשני, ההכאה חייבת להיות באלכסון). שאר הכלים מכים בתנועתם הרגילה.

### מהלכים מיוחדים:

מסע כפול - במהלכו הראשון של החייל הרגלי יכול לזוז 2 משבצות קדימה במקום משבצת אחת. הצרחה - במצב בו אין כלים בין המלך ואחד הצריחים (של אותו השחקן), המלך אינו מאויים, המלך והצריחים לא זזו ממקומם הטבעי (מקומם בתחילת משחק), ואין משבצת בין המלך לצריח המאויימת על ידי שחקן יריב, אפשר לעשות פעולת הצרחה - בפעולה זו המלך זז 2 משבצות לכיוון הצריח והצריח זז למשבצת שמצידו השני של המלך.



ההצרחה בצד המשחק של המלך (ימין בתמונה למעלה) מכונה הצרחה קצרה, והצרחה בצד המלכה (צד שמאל) מכונה הצרחה ארוכה.

הכאה דרך הילוכו (En Passant) - כאשר רגלי מבצע מסע כפול, יכול רגלי אחר להכות אותו כאילו עבר רק משבצת אחת. המהלך תקף רק לתור הבא (זהו למעשה המהלך היחיד בשחמט שניתן לעשות רק בתור הקרוב ולא לאחר מכן).



### סיום המשחק:

המשחק יכול להסתיים בניצחון או בתיקו. השחקן מגיע לניצחון כאשר הוא מכה את מלך היריב או אם ליריב נגמר הזמן לבחירת מהלך. כל עוד ליריב יש זמן ואפשרות להגן על המלך שלו המשחק נמשך ואלה מהלכיו החוקיים של היריב (במצב בו המלך מאויים מההלכים האפשריים הם מהלכים המגנים על המלך בלבד) כאשר לא ניתן להגן על המלך מפני איום המשחק נגמר בתיקו כאשר לשחקן אין מהלכים חוקיים אותם הוא יכול לבצע, או שמהלך מסוים שוחק מספר פעמים ברצף או שאין מספיק כלים לביצוע מט אפשרי לכל אחד מהשחקנים.

### מאפייני המשחק:

המשחק הוא משחק לוח סופי מסוג אינפורמציה מלאה - כלומר לשני השחקנים אינפורמציה מלאה על המצב הנוכחי של המשחק. בנוסף, המשחק הוא משחק בו מספר הסוכנים הוא 2, כאשר המשחק הוא מסוג "תור מתחלף לסירוגין".

הערכות למספר המצבים האפשריים במשחק שחמט, נעות בין 10 בחזקת 43 לבין 10 בחזקת 47 (לשם השוואה, ביקום הנצפה מוערך שמספר כל הכוכבים בכל הגלקסיות הוא בסדר גודל של 10 בחזקת 24). בניית עץ עבור כל המצבים האפשריים בשחמט אינו דבר ריאלי, הן מבחינת גודל הזכרון המוגבל והן מבחינת גודל הזמן המוגבל. אין אדם או מכונה היכולים לחשב את כל הצעדים האפשריים בכל תור וכל השלכותיהם האפשריות ולהסיק מכך איזה מהלך הוא הטוב ביותר באופן מוחלט. עם זאת, בהינתן בזמן סביר, שחקן אנושי מסוגל לבצע הערכה של כמה צעדים אפשריים קדימה ולהסיק את השלכותיו של כל צעד. כאשר ככל שעומד לרשותו זמן רב יותר, הוא יכול להעריך טוב יותר ולראות את ההשלכות האפשריות של כל צעדיו לעומק יותר. למשל, במשחק המוגבל ל-10 דקות לשחקן, השחקן יכול להתעמק יותר בצעדים האפשריים שלו, לעומת משחק המוגבל לדקה אחת, בו התעמקות דומה תוביל את השחקן לאיבוד הזמן שלו ולהפסד מחוסר זמן.

### בניית סוכן שחמט "קלאסי":

ראשית, הגדרת המשחק:

- קבוצת הסוכנים: 2 סוכנים, סוכן A1 לבן וסוכן A2 שחור.
- קבוצת המצבים S: קבוצת המצבים החוקיים בשחמט.
- מצב התחלתי s0: לוח התחלתי סטנדרטי של שחמט.
- פונקציית תור Turn: בכל מצב אנו מחזיקים את אינדקס השחקן האחרון, כאשר הפעלת הפונקציה על המצב הנוכחי, תחזיר את האינדקס של השחקן השני (הפעלת Turn על מצב בו A1 הוא השחקן האחרון ששיחק, תחזיר A2)
- פונקציית עוקב Succ: תחזיר את כל המצבים האפשריים שיכולים להתרחש כתוצאה מכל מהלך אפשרי של המצב הנוכחי ובהתאם לאינדקס של הסוכן במצב הנוכחי.
- פרויקט הבדוק מצבים סופיים G: הפעלת הפונקציה על מצב s, תחזיר True אם המצב הוא מט או תיקו, אחרת תחזיר False.
- תועלת U: עבור ניצחון, 1- עבור הפסד ו-0 עבור תיקו.

### שנית, עץ המשחק ומציאת אסטרטגיה:

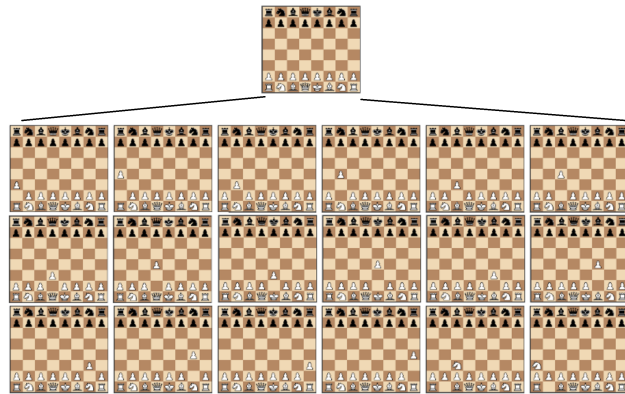
אנחנו מעוניינים לבנות עץ משחק של שחמט בתורו של כל סוכן וממנו לגלות את האסטרטגיה שתקבע מה תהיה סדרת המהלכים של הסוכן לכל סדרה אפשרית של תגובת הסוכן היריב, זאת על מנת לבחור את הצעד הטוב ביותר מבין כל הצעדים האפשריים - הצעד שתועלתו תהיה הגדולה ביותר.

כפי שנאמר קודם לכן, בניית עץ מלא במשחק כמו שחמט איננה אפשרית, הן במגבלת המקום והן במגבלת הזמן. לכן, נבנה עץ עד עומק מסוים, וכאשר נגיע לעומק זה, נפעיל פונקציה היוריסטית שתעריך את תועלתו של המצב. לא ניתן להשתמש באלגוריתם מינימקס בגלל דרישת המשאבים הגדולה של משחק השחמט, ולכן נשתמש בווריאציה מוגבלת משאבים של המינימקס - Resource-bounded Minimax (או בשמו הקצר: rb-minimax). בפועל, עבור הסוכן נשתמש בגרסת האופטימיזציה של Minimax מוגבל המשאבים והיא rb-alpha-beta.

### דוגמא:



פיתוח תת עץ של סוכן rb-minimax בעומק 1 במצב ההתחלתי של המשחק (תור הלבן):



(מקור התמונה)

כל לוח מבין 18 הלוחות מתאר את המצב הבא של הלוח, כאשר הלבן ביצע את המהלך וכעת תור השחור לפעול. ב-rb-minimax המוגבל לעומק 1, ניתן לשערך כעת על ידי הפונקציה היוריסטית את כל אחד מהלוחות ולהחזיר את הערך המקסימלי כמצב אליו הסוכן הנוכחי ירצה להגיע. ב-rb-minimax המוגבל לעומק 2 לעומת זאת, אנו נפתח כל אחד מ-18 המצבים לתת עץ של מצבים משלו (על ידי הפעלת Succ על כל אחד מ-18 המצבים הללו) ואותם נשערך על ידי הפונקציה היוריסטית, כאשר הערך המינימלי לכל אחד מתתי העץ של המצבים הללו יבחר (התועלת של היריב היא הפוכה משלנו) ומהם נבחר את הערך המקסימלי עבור הסוכן שלנו.

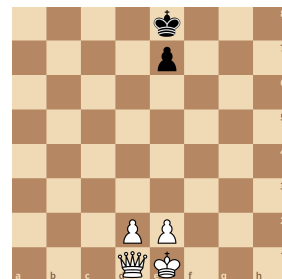
### הפונקציה ההיורסטית:

הגדרת הפונקציה ההיוריסטית H של סוכן השחמט, שתשערך לנו את התועלת המצב הנוכחי של הלוח, היא בעיה מורכבת, בשל מורכבתו הרבה של משחק השחמט. בשחמט יש חשיבות למיקומם של הכלים, לערכי הכלים - כלומר מהו החשיבות כל כלי (למשל, מלכה שווה יותר מרץ) ולסיטואציות רבות שיכולות להשפיע על השיערוך של התועלת של המצב הנוכחי כמו: מצב שיש בו שח, מצב בו היו חזרות של כמה צעדים לפני המצב הנוכחי (מהלך שיכול להוביל לתיקו) ועוד. כמו כן, השיערוך של המצב הנוכחי s על ידי הפעלת הפונקציה H, יהיה תלוי בזהותו של הסוכן המפעיל את H. היוריסטיקה שתשערך את המצבים בעץ בה אנו נשתמש בכל מהלך הפרויקט, מבוססת על המדדים הבאים: **ערכם של הכלים, מיקומם של הכלים, סיטואציות של שח ומט, מצב של תיקו אפשרי.**

- שיערוך של מצב הלוח לפי ערכם של הכלים:  
לכל כלי בשחמט נגדיר ערך משלו:







0	90	50	30	30	10

נגדיר פונקציה eval\_board המקבלת את המצב הנוכחי ומחזירה את ערך הלוח בהתאמה ובאופן הבא: נעריך את ערך הלוח לפי הכלים הלבנים בלבד, לאחר מכן לפי השחורים בלבד ולבסוף ערך הלוח יהיה חיסור הערכים בערך המוחלט. ערך המלך הוא 0 משום שלא קיים מצב בו נצטרך לשערך לוח בלי מלך.  
לדוגמא:



בהינתן המצב באיור הנ"ל, ערך הלבנים הוא 110:  $2*10 + 1*90 + 0$  וערך השחורים הוא 10:  $1*10 + 0$ , כלומר הפעלת eval\_board על המצב הנוכחי תניב לנו ערך 100. בהמשך, ערך זה יחשב כ-100 או 100, בהתאם לזהותו של הסוכן במצב הנוכחי (כך שלמעשה אם הסוכן שלנו הוא לבן, ניתן להגדיר באופן דומה את הכלים של הסוכן השחור כ-10- עבור החייל, כ-30- עבור הרץ וכדומה).

- שיערוך של מצב הלוח לפי מיקומם של הכלים:  
בנוסף לשיערוך לפי ערך הכלי, נוסיף לכך את מיקומו של הכלי. לכל כלי יש מיקומים בהם יש לו יתרון (למשל, יש לו יותר אפשרויות תזוזה ואפשרויות לאיים), לעומת מיקומים בהם הוא פחות יכול להוות יתרון לשחקן המשחק.  
לדוגמא: מלכה לבנה או שחורה, גרועה יותר בפניות של הלוח, מאשר סביב אזור מרכז הלוח.  
לכן, נוסיף לחישוב של eval\_board, את מיקומם של הכלים באופן הבא:  
לכל כלי נגדיר את כל המיקומים שעל הלוח בהם יכול להוות יתרון ואת כל המיקומים שעל הלוח בהם יכול להוות חיסרון. עבור הכלים הלבנים ובאופן סימטרי עבור הכלים השחורים, נגדיר:

	
<pre>[ -3.0, -4.0, -4.0, -5.0, -5.0, -4.0, -4.0, -3.0], [ -3.0, -4.0, -4.0, -5.0, -5.0, -4.0, -4.0, -3.0], [ -3.0, -4.0, -4.0, -5.0, -5.0, -4.0, -4.0, -3.0], [ -3.0, -4.0, -4.0, -5.0, -5.0, -4.0, -4.0, -3.0], [ -2.0, -3.0, -3.0, -4.0, -4.0, -3.0, -3.0, -2.0], [ -1.0, -2.0, -2.0, -2.0, -2.0, -2.0, -2.0, -1.0], [  2.0,  2.0,  0.0,  0.0,  0.0,  0.0,  2.0,  2.0 ], [  2.0,  3.0,  1.0,  0.0,  0.0,  1.0,  3.0,  2.0 ]</pre>	<pre>[ -2.0, -1.0, -1.0, -0.5, -0.5, -1.0, -1.0, -2.0], [ -1.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0, -1.0], [ -1.0,  0.0,  0.5,  0.5,  0.5,  0.5,  0.0, -1.0], [ -0.5,  0.0,  0.5,  0.5,  0.5,  0.5,  0.0, -0.5], [  0.0,  0.0,  0.5,  0.5,  0.5,  0.5,  0.0, -0.5], [ -1.0,  0.5,  0.5,  0.5,  0.5,  0.5,  0.0, -1.0], [ -1.0,  0.0,  0.5,  0.0,  0.0,  0.0,  0.0, -1.0], [ -2.0, -1.0, -1.0, -0.5, -0.5, -1.0, -1.0, -2.0]</pre>
	
<pre>[  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0], [  0.5,  1.0,  1.0,  1.0,  1.0,  1.0,  1.0,  0.5], [ -0.5,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0, -0.5], [ -0.5,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0, -0.5], [ -0.5,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0, -0.5], [ -0.5,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0, -0.5], [ -0.5,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0, -0.5], [  0.0,  0.0,  0.0,  0.5,  0.5,  0.0,  0.0,  0.0]</pre>	<pre>[ -2.0, -1.0, -1.0, -1.0, -1.0, -1.0, -1.0, -2.0], [ -1.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0, -1.0], [ -1.0,  0.0,  0.5,  1.0,  1.0,  0.5,  0.0, -1.0], [ -1.0,  0.5,  0.5,  1.0,  1.0,  0.5,  0.5, -1.0], [ -1.0,  0.0,  1.0,  1.0,  1.0,  1.0,  1.0, -1.0], [ -1.0,  1.0,  1.0,  1.0,  1.0,  1.0,  1.0, -1.0], [ -1.0,  0.5,  0.0,  0.0,  0.0,  0.0,  0.5, -1.0], [ -2.0, -1.0, -1.0, -1.0, -1.0, -1.0, -1.0, -2.0]</pre>
	
<pre>[ -5.0, -4.0, -3.0, -3.0, -3.0, -3.0, -4.0, -5.0], [ -4.0, -2.0,  0.0,  0.0,  0.0,  0.0, -2.0, -4.0], [ -3.0,  0.0,  1.0,  1.5,  1.5,  1.0,  0.0, -3.0], [ -3.0,  0.5,  1.5,  2.0,  2.0,  1.5,  0.5, -3.0], [ -3.0,  0.0,  1.5,  2.0,  2.0,  1.5,  0.0, -3.0], [ -3.0,  0.5,  1.0,  1.5,  1.5,  1.0,  0.5, -3.0], [ -4.0, -2.0,  0.0,  0.5,  0.5,  0.0, -2.0, -4.0], [ -5.0, -4.0, -3.0, -3.0, -3.0, -3.0, -4.0, -5.0]</pre>	<pre>[ 0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0], [ 5.0,  5.0,  5.0,  5.0,  5.0,  5.0,  5.0,  5.0], [ 1.0,  1.0,  2.0,  3.0,  3.0,  2.0,  1.0,  1.0], [ 0.5,  0.5,  1.0,  2.5,  2.5,  1.0,  0.5,  0.5], [ 0.0,  0.0,  0.0,  2.0,  2.0,  0.0,  0.0,  0.0], [ 0.5, -0.5, -1.0,  0.0,  0.0, -1.0, -0.5,  0.5], [ 0.5,  1.0,  1.0, -2.0, -2.0,  1.0,  1.0,  0.5], [ 0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0]</pre>

(מקור התמונה)

כך שבשיערוך של ערך הכלי כפי שהוגדר קודם כחלק מ-eval\_board, נחבר לערך שקיבלנו את הערך של המיקום של הכלי לפי הצבע שלו, ובסה"כ, הוספנו מידע היוריסטי נוסף של מיקום הכלי על הלוח במצב אותו רוצים לשערך, שיתווסף לחישוב בפונקציה eval\_board על ידי חיבור ערך זה.

- סיטואציות של שח ומט ותיקו:  
נגדיר פונקציות is\_in\_chess, is\_in\_draw, is\_in\_check ו-is\_in\_checkmate המקבלת כקלט את המצב הנוכחי ויחזירו האם הלוח נמצא במצב של שח, מט או תיקו (כשמצב של תיקו ייתכן במקרה של פט (stalemate), אין כלים שיכולים להוביל למט תיאורטי (insufficient material), או 3 חזרות שבוצעו עד כה).

בהגדרת הפונקציה היוריסטית, נשתמש בפונקציות הללו, כך שבמקרה של תיקו, נחזיר ערך 0 תמיד (כי אף שחקן לא מרוויח מתיקו), במקרה של מט נחזיר  $\infty$  תמיד ובמקרה של שח, נוסיף לערך הכולל של היוריסטיקה 1000 כפי שמוגדר למטה.

בסה"כ, הפונקציה היוריסטית  $H'(s)$  תוגדר באופן הבא:

$$H'(s) = \begin{cases} \infty & \text{is\_in\_checkmate}(s) \text{ is True} \\ 0 & \text{is\_in\_draw}(s) \text{ is True} \\ \text{eval\_board}(s) + 1000 * \text{is\_in\_check}(s) & \text{else} \end{cases}$$

וכפי שנאמר קודם, הערך המתקבל מהפונקציה בהפעלה על המצב, יהיה תלוי בזהותו של הסוכן הנוכחי ובהיותו של הסוכן שכעת תורו, כך שנגדיר את H שתקבל 2 פרמטרים, כפי שמוגדר באלגוריתמי ה-minimax:

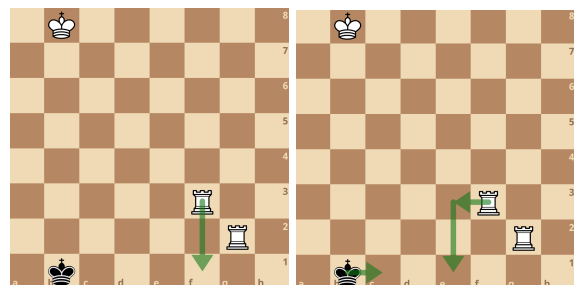
$$H(s, \text{agent}) = \begin{cases} H'(s) & \text{Turn}(s) == \text{agent} \\ -1 * H'(s) & \text{else} \end{cases}$$

### העדפת צעדים מידיים:

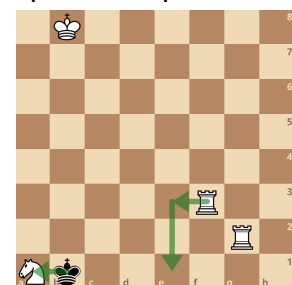
בשחמט ובאופן כללי במשחקים המחושבים על ידי אלגוריתמי rb-minimax, ייתכנו כמה מצבים בעלי ערך יוריסטי זהה אך מספר הצעדים שעלינו לעשות כדי להשיגם שונה. במקרה של היוריסטיקה שלנו, אם במצב הנוכחי יש מט לטובתינו, יוחזר  $\infty$  וייתכן שגם אם בעוד 2 צעדים יהיה מט לטובתינו ויוחזר גם כן  $\infty$ . במקרה זה, אלגוריתם ה-rb-minimax יבחר באופן רנדומלי את הצעד הבא ולכן ייתכן שיידחה את המט הקרוב, כי הוא לא מבדיל בין  $\infty$ .

### לדוגמא:

כעת התור של הלבן:



2 המצבים יניבו ערך  $\infty$ . האסטרטגיה של עץ המשחק שתוחזר ב-rb-minimax המוגבל לעומק 3 למשל, יכולה להיות כל אחד מהמצבים הללו - במצב באיור השמאלי, העומק של האלגוריתם יהיה בעומק 2 שכן המשחק נגמר במהלך יחיד ובמצב באיור הימני העומק יהיה 3. כלומר, הצעד הבא שנבחר יבחר שרירותית מבין ה-2 וייתכן שנעדיף צעדים דחיינים כמו המצב הימני. יתרה מכך, משום שרוב הצעדים האפשריים הם דחייניים בדוגמא, ההסתברות גבוהה יותר שיבחרו שרירותית מאשר המצב היחיד שמוביל למט מידיי. לעיתים, הדבר עלול לגרום אף לאיבוד הכלים, משום שיש מיסוך של מצבים גרועים ללבן על ידי המט המובטח לפי השיערוך של העומק. למשל:



שכן הלבן יוכל לבחור את הצעד f3e3 באופן שרירותי, כי יראה שבעתיד ייתכן לו מט והפסד הפרש "ימוסך" על ידי ה- $\infty$  שיוחזר מהיוריסטיקה. יתרה מכך, דחייה עלולה להוביל לבזבז זמן של הסוכן, כל תור עולה לסוכן משאב זמן יקר. לכן, באלגוריתם ה-rb-minimax נוסיף לערך הפונקציה היוריסטית במקום בו קוראים לה באלגוריתם, ערך epsilon העומק, כך שכל שנעמיק, יהיה ייתרון בערך היוריסטי למצבים פחות עמוקים. כמו כן, בפונקציה היוריסטית שהגדרנו, נגדיר את  $\infty$  להיות מספר גדול מאוד (10 בחזקת 10) כדי שיהיה משמעות ל-epsilon (שנוכל להבדיל בין מצבים שונים בעלי ערך  $\infty$  אך בעומקים שונים).

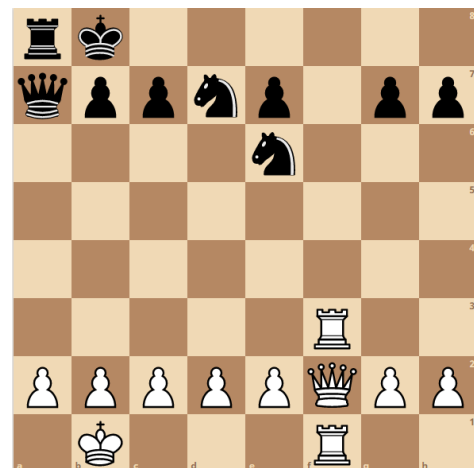
## פרק 2 - בניית סוכן Minimax Selective deepening למשחק

### השחמט:

מדוע כדאי להשתמש ב-Selective deepening כשמדובר במשחק השחמט? לפי ויקיפדיה מקדם הסיעוף של עץ מינימקס במשחק השחמט עומד על בין 31-35 (שזה בעצם מספר המהלכים אפשריים לתור במוצק). במצב בו מקדם הסיעוף גדול (כמו שהוצג בפתיחה) יש סיכוי גדול להרבה מהלכים סתמיים שיבדקו.

### אינטואיציה אנושית:

בפרק הקודם, נתנו אינטואיציה אנושית למשחק דמוי שחמט. כעת, ניתן חידה קלה להמחשת האינטואיציה בשימוש בוריאנט ה-MSD במשחק שחמט. לדוגמא: תור הלבן.



### הפתרון:

ניתן לראות שלבן מנצח את המשחק אם יקריב את הצריח בתורו הראשון והמלכה בתור השני (בעזרת מהלכי שח) ולבסוף שחמט עם הצריח השני.

אנחנו בטוחים שקורא אנושי של החידה בכלל לא שקל להזיז את החיילים פשוטים או את המלך שלו כשניגש לפתור את החידה, שכן ניתנה **העדפה מיידיית** לחקור במהלכים שעושים שח או מהלכים עם כלים חזקים. כמו כן, גם בשיקול צעדי היריב, בתגובה לצעדים שלו, הוא העדיף לחקור 2 עד 3 צעדים אפשריים ולא חקר כל צעד של כל חייל של היריב. ייתכן שפתרון החידה היה **מהיר** בזמן, על אף שיש ללבן מספר רב מאוד של צעדים שיכל לבחור להתעמק בהם.

סוכן המשתמש באלגוריתם המינימקס (או  $\alpha\beta$ ) המוגבל לעומק 2 למשל, לא יבצע מהלכים אלה כלל ובכך אולי יוותר על הניצחון (לפי ההיוריסטיקה שהצגנו ללבן ולשחור אותו value של כלים). אם היינו משתמשים בסוכן המינימקס עם MSD, הייתה לנו מלכתחילה את האפשרות לחקור **לעומק גדול יותר, תחת אותה מגבלת זמן אך עם tradeoff של ויתור של העמקת מהלכים בכל העמקה** (בחירת מהלכים להעמקה חכמה), הוא היה פותר את החידה ומנצח.

בנוסף, סוכן המשתמש ב-MSD יוכל לשחק את כל המשחק בעומק גדול יותר ובכך (אולי, בתנאי שלא סינן יותר מדי צעדים וסינן באופן חכם את הצעדים להעמקה) להרוויח יתרון על יריבו. סינון חכם של מצבים יתבצע על ידי הפונקציה היוריסטית.

### הפונקציה ההיוריסטית לסידור מהלכים אפשריים ותעדוף העמקה:

בבניית הפונקציה ההיוריסטית, הבנויה על האינטואיציה שדיברנו קודם לכן, עומד אתגר כפול. מלבד העובדה שהפונקציה צריכה להיות טובה בזיהוי מידי של מהלכים טובים, היא צריכה להיות יעילה מאוד בזמן. איננו מעוניינים בפונקציה בזבזנית בזמן, משום שאנו זקוקים לפונקציה שנקראת מספר רב (מאוד) של פעמים ותהיה יעילה הן מבחינת סיבוכיות הזמן והן מבחינת זיהוי המהלכים הטובים.

בנוסף, הפונקציה ההיוריסטית H\_Deepening שונה מהפונקציה אותה הגדרנו להערכת התועלת (שבה נשתמש באלגוריתם להערכת המצב) ומתבססת על האינטואיציות עליהן דיברנו קודם לכן:

- בחירת מהלכים אפשריים שפותחים המון מהלכים חדשים והפחתת חשיבות של כאלו שחוסמים.
- תזוזה של כלים חזקים עדיפה על תזוזה של כלים חלשים.
- מהלכים שנותנים מט מידי.
- מהלך שנותן תיקו מידי - למשל, תיקו שנובע מ-3 מהלכים רצופים חוזרים שהמהלך גורם.
- מהלך שמציל את הכלי מתקיפה.
- מהלך שמוביל את הכלי להיות תחת מתקפה.
- מהלך שמוביל את הכלי לאכילת (הכאת) כלי של היריב.

מבין כל המדדים הללו, 2 מדדים אינם בינאריים:

גודל תוספת המהלכים החדשים שנוספו בעקבות הצעד עתידי וערך הכלי אותו מזיזים בצעד העתידי.

בנוסף לכך, משום שאיננו יודעים מה עדיף, בחירת מהלכים אפשריים שפותחים המון מהלכים חדשים או תזוזה של כלים חזקים, ניתן משקל שונה לכל אחד מהם:  
 $\gamma$  - משקל לערך הכלי.  $\theta$  - משקל לתוספת הצעדים.

נגדיר 2 פונקציות עזר:

- $NewMoves$  שתחשב את תוספת הצעדים שהמצב מוסיף באופן הבא:  
 $NewMoves(s) = |Succ(s)| - |children|$   
כאשר  $s$  הוא מצב הלוח לאחר ביצוע המהלך אותו אנו רוצים לשערך ו- $|children|$  הוא כמות המהלכים האפשריים לפני ביצוע המהלך.
- $EvalPiece$  שתחשב את ערך הכלי שאותו אנו מעוניינים להזיז, בהתאם לאותם ערכי כלים אותם הגדרנו בפונקציה ההיוריסטית הרגילה קודם לכן, כאשר הערך המוחזר מחולק ב-10 כדי שסדר גודל הערכים יהיה זהה לשל  $NewMoves$ . כמו כן, נניח כי  $s.piece$  מחזיר את סוג הכלי של המהלך המשוערך.

נגדיר את הפונקציה ההיוריסטית H\_Deepening (בקיצור H\_D) לסידור מהלכים אפשריים באופן הבא:

$$H\_D(s) = \begin{cases} \infty & \text{is\_checkmate}(s) \text{ is True} \\ 1000 & \text{is\_escape\_from\_attack}(s) \text{ is True} \\ & \text{or} \\ & \text{will\_capture\_enemy}(s) \text{ is True} \\ -\infty & \text{is\_3\_repetitions}(s) \text{ is True} \\ -1000 & \text{will\_become\_attack\_by}(s) \text{ is True} \\ \theta * NewMoves(s) + \gamma * EvalPiece(s.piece) & \text{else} \end{cases}$$

כאשר:

- הפונקציה `is_checkmate` מחזירה `True` אם המצב יגרום למט.
- הפונקציה `is_3_repetitions` מחזירה `True` אם המצב יגרום ל-3 חזרות ולתיקו.
- הפונקציה `is_escape_from_attack` מחזירה `True` אם המצב יציל את הכלי מהתקפה.
- הפונקציה `will_become_attack_by` מחזירה `True` אם המצב יגרום לכלי להיות תחת התקפה.
- הפונקציה `will_capture_enemy` מחזירה `True` אם המצב הוא אכילת הכלי של היריב.

בנוסף, כחלק מהגדרת הפונקציה היוריסטטית, ניתנת עדיפות לפונקציה `will_capture_enemy` על פני הפונקציה `will_become_attack_by`. כלומר, משום שיתכן שהמצב הוא אכילת היריב אך בו בזמן הכלי שלי מאיים על ידי היריב (בעקבות האכילה) ו-2 הפונקציות האלו יחזירו `True`, הפונקציה של `will_capture_enemy` היא תחשב מבין ה-2, כלומר יוחזר 1000 במקרה זה. הדבר נובע מכך שלעיתים אכילת הכלי של היריב ואיומו של הכלי שלי על ידי היריב (באותו המצב) יכולה לעיתים לגרום למהלך מבריק (הקרבה בתמורה למהלך עתידי טוב). מפני שאנחנו לא יודעים אם באמת המהלך יכול להיות מבריק ואת זה נדע רק על ידי העמקה, נעדיף שלא לסנן מהלך זה ולכן ניתן לו ערך גבוהה מאוד (1000), כלומר, ייתכן שמהלך כזה לא יסונן. נשים לב שמתן ערך של  $\infty$  ייתקבל רק במקרה של מהלך של מט מידי ודאי.

#### אתגרים בסוכן שחמט מבוסס וריאנט MSD:

2 האתגרים המרכזיים בבניית סוכן שחמט המבוסס MSD, הם **הזמן והאיכות**. **מבחינת זמן**, השאיפה שלנו היא לבנות סוכן שיוכל להעמיק בעומק גדול יותר מסוכן מקביל לו ללא וריאנט ה-MSD, תחת אותה מגבלת זמן. כלומר, אם הסוכן היריב מסוג `rb-alphabeta` מבצע העמקה לעומק `D` בזמן `T`, אז הסוכן `rb-alphabeta` שלנו יעמיק לעומק `D+1` בזמן קרוב מאוד ל-`T`. כל זאת כמובן כתלות ב-`KeepRate`: במקרי הקיצון, עבור `KeepRate` קרוב ל-100%, ברור כי לא נוכל להעמיק יותר מהסוכן היריב תחת אותה מגבלת זמן `T`, משום שאנחנו זהים לו מבחינת בדיקת המצבים ואף מבצעים מספר רב של פעמים את הפונקציה `filter` ללא כל שימוש ועבור `KeepRate` קרוב ל-0%, סביר להניח שנוכל לבצע העמקה גדולה יותר מהסוכן היריב (עץ המינימקס יהיה שרוך).

**מבחינת איכות**, השאיפה שלנו היא לבנות סוכן חכם - כלומר, סוכן שיודע לנצח ולנצח טוב יותר מסוכן מקביל לו ללא וריאנט ה-MSD. כלומר, משום שלסוכן שלנו יש ייתרון של עומק גדול יותר מהסוכן המקביל לו ללא הוריאנט (בשל מגבלת הזמן), אך בו זמנית הסוכן שלנו מוותר על צעדים רבים לבדיקה, נרצה שהסוכן שלנו יהיה מספיק טוב.

כל זאת גם כן כתלות ב-`KeepRate`: במקרי הקיצון, עבור `KeepRate` קרוב ל-100%, מבחינת האיכות (ובהתעלמות ממגבלת הזמן) נהיה שחקנים טובים יותר משום שאנחנו מעמיקים יותר מיריבנו וגם בודקים את כל המצבים האפשריים באופן דומה ליריבנו ועבור `KeepRate` קרוב ל-0%, סביר להניח שנהיה שחקנים גרועים יותר משום שלמרות שהעמקה גדולה יותר, הראייה שלנו תהיה מאוד צרה מכיוון שנותר על רוב המהלכים שלנו ונשאר תמיד עם מהלך יחיד בכל העמקה. כמובן ששחקן איכותי שמנצח תלוי גם בזמן, משום שבשחמט שחקן שמאבד את כל זמנו הוא מפסיד.

על מנת לענות על האתגרים הללו ולמצוא את הקשר בין ה-`KeepRate` לזמן ולאיכות, נבצע סדרת ניסויים שבסופם נוכל לבנות שחקן אופטימלי ככל שניתן, כך שבהינתן משחק מוקצב בזמן `T`, נדע לתת לבנות סוכן עם `KeepRate` ועומק `D` בצורה הטובה ביותר, שתוביל לשחקן טוב ממקבילו בעומק `D-1` וללא וריאנט ה-MSD.

הערה: אנחנו נשתמש מעתה והלך רק `rb-alphabeta`, גם עבור הסוכן שלנו (עם וריאנט ה-MSD) וגם עבור הסוכנים שנבדוק מולו, משום שמעשית, אין כל סיבה להשתמש בסוכן מסוג `rb-minimax` - שני האלגוריתמים מחזירים את אותה האסטגרטיה ו-`rb-alphabeta` מהיר בהרבה מ-`rb-minimax`.

## **פרק 3 - מתודולוגיה ניסויית:**

המטרה שלנו היא לבדוק את הקשר בין ערך KeepRate לזמן ולאיכות וכן גם לעומק. לאחר הבנת כל הקשרים הללו, נוכל לבנות סוכן עם הפרמטרים הטובים ביותר של KeepRate ועומק D, שיתאימו באופן הטוב ביותר לכל סגנון של משחק שחמט המוגבל זמן: סוכן למשחק 1דקות-1דקות (כלומר, דקה אחת לכל שחקן בסה"כ) ולמשחק 5-5, 10-10 וכו'.

### **ניסויים בזמנים:**

האתגר הראשון שעומד לפנינו הוא אתגר הזמן. אנו מעוניינים לבדוק אם הוספת הוריאנט אכן מאפשרת לנו מבחינת זמנים, לחפש לעומק גדול יותר מאותו העומק שהיינו מחפשים באלגוריתם הרגיל, תחת אותה מגבלת הזמן. כדי לבדוק זאת, נבצע סדרה של ניסויים לבדוק האם בכלל כדאי לנו להשתמש באלגוריתם עם הוריאנט החדש מבחינת זמן ואם כן, מאיזה שיעור של סינון? כפי שנאמר קודם לכן, צפוי ששיעור קיצוני של 99.9% של השארת ילדים (KeepRate) יוביל אותנו לאלגוריתם rb-alphabeta רגיל אך בתוספת של קריאות רבות של פונקציות היוריסטיות ומיון ילדים, כך שבמקרה זה לא רק שלא ישתלם לנו מבחינת זמן להעמיק לעומק גדול יותר, אלא הדבר עלול להיות גרוע יותר בזמן מהאלגוריתם המקורי בעומק קטן יותר. כלומר, צפוי שעומק D של האלגוריתם המקורי יהיה מהיר בהרבה בזמן הממוצע לתור, מהזמן הממוצע לתור של עומק D באלגוריתם עם MSD. מהצד השני צפוי ששיעור קיצוני של 0.01% של השארת ילדים, אכן יראה שיפור משמעותי בזמן, בהנחה והפונקציה היוריסטית לא איטית באופן קיצוני. זאת משום שבכל פעם נישאר עם ילד אחד ונישאר למעשה עם עץ שהוא שרוך. במקרה זה, צפוי ששימוש באלגוריתם שלנו עם הוריאנט יהיה מהיר בהרבה בזמן הממוצע לתור מהאלגוריתם הרגיל ללא הוריאנט (כלומר, צפוי שעומק D של האלגוריתם המקורי יהיה איטי בהרבה בזמן הממוצע לתור, מהזמן הממוצע לתור של עומק D+1 באלגוריתם עם MSD). אנו מצפים שאכן החל משיעור מסוים של KeepRate, יהיה כדאי לנו להשתמש באלגוריתם עם הוריאנט החדש בעומק גדול יותר מעומק זהה של האלגוריתם המקורי.

הניסויים שנעשה כדי לבדוק זאת, הוא ביצוע משחקים רבים של האלגוריתם rb-alphabeta עם הוריאנט של MSD, **בעומקים משתנים ובשיעורים שונים של KeepRate וחישוב הזמן הממוצע לתור במשחק ללא הגבלת הזמן.** בנוסף, בדיקה של אלגוריתם rb-alphabeta הרגיל ללא הוריאנט, **בעומקים משתנים.** השחקן היריב יהיה זהה בכל המשחקים והוא יהיה מול סוכן יריב שבוחר צעדים באופן רנדומלי.

כל הניסויים שנבצע יתבצעו באותה הסביבה, כלומר, על אותו המחשב, כדי שכוח חישובי שונה לא ישפיע על הזמן. עם זאת, ייתכן שיהיו אירועים זניחים שיכולים להשפיע מעט על הזמן, כמו עומס המערכת ותוכניות שפועלות ברקע ששונות מניסוי לניסוי ונדאג ככל שניתן שלא ישפיעו באופן משמעותי על הזמנים.

### **ניסויים באיכות האסטרטגיה:**

האתגר השני שעומד לפנינו הוא אתגר האיכות. אנו מעוניינים לבדוק אם הוספת וריאנט ה-MSD לאלגוריתם לא תשפיע לרעה על איכות השחקן שלנו ואם כן, החל מאיזה שיעור? איכות השחקן שלנו **תימדד ביכולתו לנצח.** נגדיר את **ציונו של הסוכן** אותו נרצה לבדוק, לפי מספר הניצחונות שהשיג מתוך טורניר של 10 משחקים מלאים כנגד **סוכן הבקרה**. כלומר, אם ניצח ב-3 משחקים (30% מהמשחקים בטורניר), ציונו יהיה 3 מתוך 10. תיקו יחשב כחצי נקודה. הסוכן היריב שמולו נבדוק את התוצאות, "**סוכן הבקרה**", הוא סוכן rb-alphabeta רגיל (ללא הוריאנט) המוגבל לעומק 2 - שחקן לא טוב מדיי אך גם לא גרוע מדיי, בעל היוריסטיקה הרגילה להערכת המצב אותה הגדרנו בפרק המבוא. נבצע 10 משחקים (טורניר) של סוכן rb-alphabeta עם העמקה סלקטיבית, **לכל אחד מהעומקים הנבדקים,** כאשר **לכל עומק נבדוק שיעור סינון שונה.** למשל, נבדוק 10 משחקים לסוכן המסנן 50% מהילדים ובעומק 4 וציונו יהיה מספר הניצחונות שניצח את שחקן ה-rb-alphabeta המוגבל לעומק 2, בתוספת חצי נקודה לכל תיקו.

על מנת שכל המשחקים יהיו **שווים זה מזה**, אם יש לכמה מהלכים שונים את אותם הערכים היוריסטיים והם הערכים היוריסטיים הטובים ביותר, נבחר מהלך רנדומלי מבין אותם הערכים. כך, נוכל להשוות את איכות האסטרטגיה של הסוכנים rb-alphabeta עם העמקה סלקטיבית ואת הקשר של KeepRate לאיכות.

המשחקים בניסויים אלו לא יהיו מוגבלים בזמן, משום שכאן הבדיקה היא בדיקת הקשר של איכות הסוכן לעומק ול-KeepRate, ללא קשר לזמן המוגבל לו. עם זאת, איכות השחקן תלויה מאוד גם ביכולת שלו לעמוד בזמן שלו ולכן בנוסף לכל, נמדוד את הזמן הממוצע לתור של הסוכן, גם במשחקים אלו. (בהמשך נשקלל את התוצאות של הניסויים באיכות האסטרטגיה עם הניסויים של הזמנים לכדי תובנה סופית שתקשר בין KeepRate, עומק, זמן ואיכות).

משום שאיכות הסוכן תלויה ב**היוריסטיקת H\_Deepening** שבנינו, נרצה שהסוכן שלנו ישתמש בהיוריסטיקה באופן האופטימלי ביותר, משום שכזכור, כעת פרמטרי המשקלים  $\theta$  ו- $\gamma$  לא מוגדרים באופן מפורש (אין להם ערך קבוע). תפקידם בהיוריסטיקה כפי שהגדרנו קודם לכן הוא:  
 $\gamma$  - משקל לערך הכלי,  $\theta$  - משקל לתוספת הצעדים. היוריסטיקה שלנו היא קומבינציה לינארית של 2 אלו, יחד עם עוד סיטואציות בינאריות במצב הנוכחי כפי שהוגדר לפני כן. עלינו לכוון פרמטרים אלו עוד לפני שנבצע את הניסויים באיכות האסטרטגיה.

### ניסויים בכוון הפרמטרים $\theta$ ו- $\gamma$ של היוריסטיקת הסידור:

משום שאיננו יודעים בוודאות שההיוריסטיקה שלנו אכן תשאיר את הילדים הטובים ביותר, שכן איננו יודעים בוודאות האם המשקל לערך הכלי חשוב יותר מהמשקל לתוספת הצעדים, נרצה לכוון 2 פרמטרים אלו. לולא כוון, אם היינו מנחשים משקלים כלשהם עבור היוריסטיקה והיינו מקבלים בניסויים באיכות השחקן, שחקן פחות טוב, לא יכולנו לדעת האם הדבר קשור בשיעור הסיון או דווקא בפונקציה היוריסטית או בשניהם.

הדרך בה נכוון את הפרמטרים היא על ידי ביצוע טורנירים ללא הגבלת זמן של סוכן rb-alphabeta בעומק קבוע ועם העמקה סלקטיבית בשיעור של 40% מהילדים (KeepRate=40%), עם ערכים שונים של  $\theta$  ו- $\gamma$  בכל טורניר (קומבינציה לינארית שונה של  $\theta$  ו- $\gamma$ ). מול הסוכן יתמודד סוכן ה-rb-alphabeta ללא הוריאנט בעומק קטן ממנו ב-1 (עומק 3).

למשל, 10 משחקים עבור הקומבינציה של  $\theta=0.3$  ו- $\gamma=0.7$ , כאשר אם הסוכן ניצח ב-4 משחקים ועשה תיקו 1, ציוני הסופי עבור קומבינציה זו תהיה 4.5.

הסיבה לשימוש ב-KeepRate=40% היא כדי שלא נגרום לניפוי גם מדוי של ילדים, אך בו בזמן שלסיון תהיה משמעות גדולה. כלומר, אם למשל היינו בוחרים ב-KeepRate=95%, אז היינו מתנהגים באופן דומה לאלגוריתם ללא הוריאנט מבחינת איכות והמשמעות של סידור הילדים הייתה נמוכה ומהצד השני, אם היינו בוחרים ב-KeepRate=5%, היינו חותכים יותר מדוי ילדים והדבר יקשה עלינו בכיוון של הפרמטרים בהיוריסטיקת הסידור, שכן תמיד היינו נשארים עם מהלך בודד או 2 לכל היותר. עם זאת, נבצע סדרת ניסויים נוספת וזהה גם עבור סוכן עם KeepRate=60% כדי שנוכל לאשש את כל התובנות הללו.

לבסוף, נוכל לשערך את המשקלים של  $\theta$  ו- $\gamma$  ולהפוך אותם לקבועים, על מנת לעבור לניסויי איכות האסטרטגיה עם היוריסטיקה סידור האופטימלית ביותר, יחסית לשיטת הכוון הזו.



## פרק 4 - תיאור הניסויים, ניתוח תוצאות ומסקנות:

### ניסויים בזמנים:

לכל אחד מערכי KeepRate הבאים: 20% 35% 50% 75% 90%, נבדוק את כל אחד מהעומקים: 2 3 4 5 6, כך שבכל פעם נבצע משחק מול שחקן שבוחר מהלכים רנדומלית ונפעיל שעון כולל לסוכן שלנו, ומונה סה"כ צעדים שביצע השחקן כך שבסוף המשחק, נחלק את סה"כ הזמן לסה"כ הצעדים שביצע השחקן, לקבלת **הזמן הממוצע לתור** שביצע הסוכן שלנו.

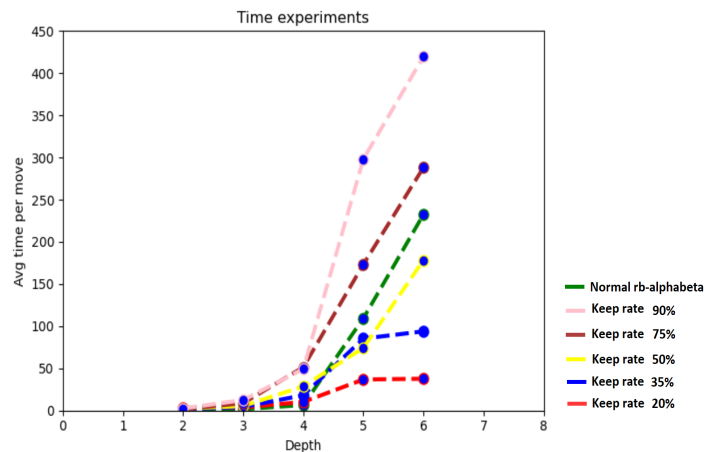
ערכי ה-KeepRate שבחרנו לניסוי משאירים 35%, 50%, ו-75% מהמהלכים שכן, אנו מעריכים שבטווח הזה נמצא ערך שהחל ממנו תהיה לנו תועלת אל מול האלגוריתם הרגיל ללא הסינון. בחרנו גם את השיעורים הקיצוניים 90% כדי לבדוק את הקצוות, כשאנו מעריכים ש-90% ומעלה עלול להזיק לנו מבחינת זמן ממוצע לתור אל מול סוכן בעומק דומה ללא הסינון ו-20% לעומת זאת, עלול להועיל לנו מאוד מבחינת זמן ממוצע לתור אל מול סוכן דומה בעומק דומה ללא הסינון.

בנוסף, יבוצע ניסוי זה עם סוכן רגיל של אלגוריתם rb-alphabeta (ללא וריאנט ה-MSD), בכל אחד מהעומקים 2 3 4 5 6, כך שגם כאן, נבצע משחק מול שחקן שבוחר מהלכים רנדומלית ולבסוף נחשב את הממוצע הזמן לתור בכל עומק.

### תוצאות הניסוי:

הגרף שהתקבל ותוצאות הניסוי:

name	2	3	4	5	6
normal	0.2	1.6	6.6	109.3	232.5
keep_rate_0_2	3.2	3.7	10.2	36.8	37.7
keep_rate_0_35	1.7	4.3	18.1	85.3	93.9
keep_rate_0_5	2.3	5.4	28.6	74.4	178.3
keep_rate_0_75	1.6	8.8	51.3	173.3	288.2
keep_rate_0_9	2.5	12.4	49.3	298.2	419.6



### ניתוח התוצאות:

כצפוי, כל הגרפים אקספוננציאליים, שכן העמקה גדולה יותר בעץ ה-minimax של שחמט, מגדילה את גודל העץ באופן אקספוננציאלי ובכך גם את זמן החיפוש (ציר ה-y). נשים לב שהגרף של rb-alphabeta (בירוק), נמצא בין הגרף של השארת 50% מהילדים (בצהוב) ובין הגרף של 75%. כמו כן נשים לב שהגרף של 90% (בורוד) הוא הגרף בעל העלייה האקספוננציאלית הגדולה ביותר. ננתח את כל התוצאות הללו, יחד עם טבלת הערכים שקיבלנו.

נראה כי השארת 20% מהילדים (באדום בגרף), בעומק 6, שוות ערך בממוצע **לכשליש** מהזמן לתור בממוצע של אלגוריתם rb-alphabeta בעומק 5. כלומר, לא רק שביצענו את האלגוריתם בעומק גדול יותר, אלא הרווחנו שיפור משמעותי בזמן, על אף העמקה גדולה יותר מהעמקה של האלגוריתם המקורי.

עם זאת, נראה כי השארת 35% מהילדים (בכחול בגרף) בעומק 6, בקירוב שוות ערך לזמן של אלגוריתם rb-alphabeta בעומק 5. כלומר, היא די משיגה את מה שאליה התכוונו, כך שיש לנו משאב זמן T דומה אך עומק גדול יותר, כאשר הסינון הוא לא גס מדי (השארת 35% מהילדים) וכן הזמן הממוצע לתור שהשגנו הוא בקירוב  $T \sim$  (בקירוב של שניות בודדות ל-T, דבר יחסית זניח למשחקים של 10 דקות למשל).

בסינון 50% מהילדים (בצהוב בגרף), העמקה לעומק 6 פחות טובה לערך הזמן של אלגוריתם rb-alphabeta בעומק 5. כלומר, לא השגנו את מטרת הוריאנט, משום שאם rb-alphabeta בעומק 5 לוקח T זמן, להעמקה של עומק 6 עם סינון 50%, ייקח T+70 וזה כלל לא קירוב של שניות בודדות אלא דקה ו-10 שניות יותר זמן.

החל מ-75%, כלומר בגרף של סינון 75%-90% (בחום ובורוד), נראה כי ביצוע סינון כזה לא רק שלא יועיל מבחינת זמנים, אלא אף יזיק לסוכן מבחינת זמן. הסיבה היא שביצוע הפונקציה היוריסטית לכל ילד וסידור הילדים (כחלק מהפונקציה Filter) נהפכת ליקרה בזמן כאשר היא מבוצעת מספר רב מאוד של פעמים.

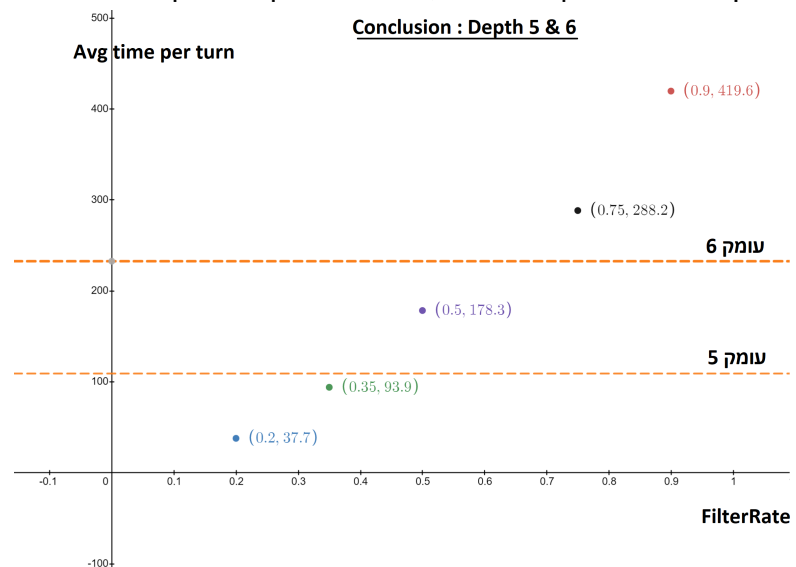
כפי שהוסבר קודם לכן, במקרה קיצון של 100% של KeepRate, האלגוריתם עם הוריאנט של הסינון יהיה זהה לחלוטין לאלגוריתם ה-rb-alphabeta, אך יבצע פונקציה היוריסטית + מיון לכל ילד, ללא כל השפעה לפלט של האלגוריתם. בשימוש ב-KeepRate של 100% הדבר שקול לחלוטין להוספת פונקציית sleep של כמה מילי-שניות במקום פונקציית ה-Filter באלגוריתם עם הוריאנט.

### מסקנות בניסויים בזמנים:

בסה"כ, מסקנות הניסוי הן:

- אם נבחר להתמקד בקבוצה קטנה של המהלכים החוקיים, נוכל להעמיק יותר עבור אותה מגבלה של זמן. (KeepRate של 25% עד 35% מקבוצת המהלכים החוקיים)
- אם נבחר להתמקד ב"חצי המבטיח" יותר של המהלכים החוקיים, נקבל שאנחנו עומדים על פחות או יותר על אותו סדר גודל של זמנים אם היינו בודקים את כל המהלכים החוקיים בעומק דומה באלגוריתם המקורי ללא הוריאנט (KeepRate סביב ה-50%).
- אם נבחר להתמקד ברוב הכלים (75% ומעלה) אנחנו מבזבזים זמן רב מדי על בחירת כלים מושכלת, כך שלא רק שלא נשיג ייתרון בהעמקה גדולה יותר אלא אף בזבזנו זמן רב יותר מאשר היינו משתמשים באלגוריתם המקורי ללא סינון ילדים ובאותו העומק.

ניתן לתאר את המסקנות של הניסוי באופן איכותי על ידי הגרף הבא, המתייחס להעמקה בעומק 6 בלבד של הסוכן שלנו ב-KeepRate שונים, אל מול העמקה בעומק 5 ו-6 של האלגוריתם המקורי ללא הוריאנט:



הקווים המקווקים בכתום הם הזמן של האלגוריתם rb-alphabeta ללא MSD, בעומק 6 ובעומק 5. כלומר, הזמן הממוצע לתור לסוכן הרגיל בעומק 6 לקח כ-230 שניות והזמן הממוצע לתור בעומק 5 לקח כ-109 שניות.

שאר הנקודות, הן התוצאות לפי עומק 6 של האלגוריתם עם העמקה סלקטיבית, לפי ה-KeepRate. נשים לב כי במקרה זה, הטווח של KeepRate העונה על מסקנה (1) - כל הערכים שמתחת וסביב לקו של עומק 5. הטווח של KeepRate העונה על מסקנה (2) - כל הערכים שממש מעל הקו של עומק 5 וממש

מתחת לקו של עומק 6. הטווח של KeepRate העונה על מסקנה (3) - כל הערכים שמעל וסביב הקו של עומק 6.

ניתן להעריך שטווח של KeepRate בין 0.36 ל-0.6 יוביל למסקנה (2), כלומר ביצוע עומק 6 לא בהכרח כדאי עבור הסוכן שלנו, משום שהוא לא שונה מהותית מהזמן של תור ממוצע של  $rb\text{-alphabet}$  אך הסוכן יפסיד ילדים וכל KeepRate מתחת ל-0.36 יוביל למסקנה (1), וכל KeepRate הגדול מ-0.6 יוביל למסקנה (3).

נשים לב שתוצאות אלה הן בדיוק מה שהאלגוריתם Selective Deepening מכון אליו - הורדת מקדם הסיעוף ממספר מאוד גדול (31~35 במקרה של שחמט או 45 במקרה של משחק סתמי) למספר קטן מאוד (3 במקרה של המשחק הסתמי או 10 במקרה של 30% מ-35) שמאפשר העמקה גדולה הרבה יותר.

### מסקנה עבור בניית הסוכן:

נראה כי סוכן המבצע  $rb\text{-alphabet}$  עם העמקה סלקטיבית בסביבות ה-36% (כלומר ניפוי של 64% מהילדים בכל פעם) ומטה, יניב לנו את המטרה עבורה יצרנו את הוריאנט - ביצוע עומק גדול יותר מהעומק המקורי, בהינתן אותו משאב של זמן. הדבר נכון גם עבור העומקים 2,3,4,5,6. לא נבדקו עומקים גדולים יותר, משום שבשחמט, גם באלגוריתמים מאוד מאוד יעילים, עומק 7 הופך ללא שמיש במגבלת זמן סבירה של משחק, בשל הגידול האקספוננציאלי במספר הילדים, גם עם KeepRate נמוך מאוד.

כמו כן, הפעלת הסוכן בעומק 6 המשאיר 35% מהילדים, לוקח 93 שניות בממוצע לתור, כלומר דקה וחצי לתור יחיד, זה לא דבר שרלוונטי במשחקים סטנדרטים של שחמט, שברובם שחקן מוגבל לזמן של פחות מ-20 דקות לכל היותר בכל המשחק ובשחמט מספר הצעדים ששחקן מבצע ממוצע מגיע ל-30.

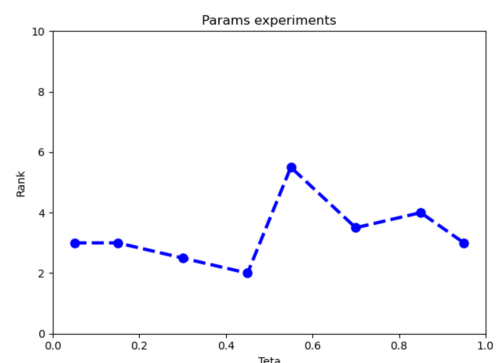
### ניסויים בכוונון הפרמטרים $\theta$ ו- $\gamma$ של היוריסטיקת הסידור:

ביצוע הניסויים בכוונון הפרמטרים בוצעו באופן הבא: סוכן  $alphabet$  המשאיר 40% מהילדים ובעומק 4 ישחק בכל פעם 10 משחקים כנגד סוכן ה- $alphabet$  ללא העמקה סלקטיבית בעומק 3 (קטן ב-1 מהסוכן שלנו), כאשר בכל פעם תהינה קומבינציה לינארית שונה של המשקלים  $\theta$  ו- $\gamma$  בפונקציה היוריסטית. למשל, 10 משחקים עבור הקומבינציה של  $\theta=0.3$  ו- $\gamma=0.7$ , כאשר אם הסוכן ניצח ב-2 מהשחקים והשיג תיקו אחד, 2.5 יהיה ציונו הסופי (rank) עבור קומבינציה זו. כמו כן, נבדוק ניסוי זהה נוסף גם עבור סוכן נוסף המשאיר 60% מהילדים. תזכורת:  $\theta$  הוא המשקל של תוספת הצעדים החדשים בהיוריסטיקה ו- $\gamma$  הוא המשקל של ערך הכלי.

### תוצאות הניסוי:

סוכן  $KeepRate = 40\%$  - סוכן המשאיר 40% מהילדים בלבד בכל עומק: הגרף שהתקבל ותוצאות הניסוי:

teta	gama	wins	draws	rank
0.05	0.95	1	4	3
0.15	0.85	0	6	3
0.3	0.7	1	3	2.5
0.45	0.55	1	2	2
0.55	0.45	3	5	5.5
0.7	0.3	2	3	3.5
0.85	0.15	1	6	4
0.95	0.05	0	6	3



הגרף הוא של  $\theta$  אל מול Rank, כך שערכי המשקלים של הסוכן נסכמים ל-1 כלומר עבור  $\theta$  בעל ערך של 0.3 אז  $\gamma$  נבדק בערך של 0.7. כל נקודה מציינת 10 משחקים (טורניר) ששחקן והדירוג אותו קיבל השחקן בסוף הטורניר.

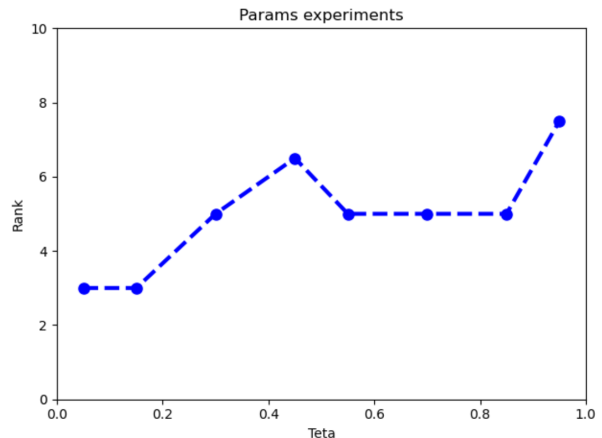
מגרף זה ניתן לראות כי  $\theta$  נמוך מדי (ובהתאמה  $\gamma$  גבוהה) אינו גורם לשחקן להיות טוב יותר. כמו כן ניתן לראות כי  $\theta$  גבוהה יותר, אך לא גבוהה מדי נותן את השחקן הטוב ביותר - כלומר, סביב הנקודה  $\theta=0.6$  ו- $\gamma=0.4$ , נראה שהשחקן שלנו משחק הכי טוב. נראה אם כך ששני המדדים חשובים, אך יש עדיפות ל- $\theta$  במעט, כלומר, ערך הרווח של מהלכים חדשים חשוב יותר במהלך כל המשחק, מערכו של הכלי.

נבדוק כעת את הסוכן הנוסף המשאיר 60% מהילדים.

סוכן  $\text{KeepRate} = 60\%$ :

הגרף שהתקבל ותוצאות הניסוי:

teta	gama	wins	draws	rank
0.05	0.95	1	4	3
0.15	0.85	2	2	3
0.3	0.7	2	6	5
0.45	0.55	4	5	6.5
0.55	0.45	1	8	5
0.7	0.3	2	6	5
0.85	0.15	3	4	5
0.95	0.05	6	3	7.5



כעת נראה שלא ניתן ללמוד הרבה מהגרף על הפרמטרים, כמו מהגרף הקודם, בשל העובדה שהשחקן כאן מקבל ערכים דומים של rank בערכים רבים של  $\theta$ . הדבר נובע מכך שאנחנו מסננים מעט מאוד מהילדים (קרובים יותר לפלט אלגוריתם ה-rb-alphabeta הרגיל).

עם זאת, מהגרף ניתן ללמוד כמה אבחנות מעניינות:

- הגרף מחזק את הטענה שקיבלנו מהגרף הקודם, שהחשיבות של  $\theta$  גדול יותר באופן כללי מהחשיבות של  $\gamma$ , שכן ניתן לראות זאת גם לפי הגרף וגם לפי הטבלה, שכאשר יש חשיבות גדולה ל- $\gamma$  על חשבון  $\theta$ , השחקן משחק פחות טוב (הדבר מורגש גם כשלא מסננים הרבה ילדים).
- כפי שציפנו, שחקן שמשאיר 60% מהילדים יהיה טוב יותר בציונו הכללי משחקן שמשאיר רק 40% מהילדים, כששניהם משחקים באותו העומק של החיפוש וכנגד אותו יריב, וכאשר אין משאב של זמן שמגביל אותם ועלול לפסול אותם (שחקן 60% איטי יותר בהרבה משחקן 40%, כפי שראינו בניסויי הזמן).
- $\text{KeepRate} = 60\%$  מוביל לתוצאה טובה בטורנירים מול הסוכן היריב הרגיל (ללא MSD) ובעומק הקטן ממנו ב-1. כלומר, נראה שהעמקה סלקטיבית שמסננת קרוב ל-50% מהילדים (בכל עומק), יוצרת שחקן שידוע לשחק טוב יותר מסוכן יריב רגיל בעומק הקטן ממנו מ-1, ועל כן ה-tradeoff של סינון ילדים אל מול העמקה גדולה יותר השתלם במקרה זה (עם התעלמות מאלמנט הזמן).

מ-2 הגרפים ומתוצאות הטבלה, ניתן להסיק שהקומבינציה הטובה ביותר, כלומר, הקומבינציה שהניבה לנו את הדירוג הגבוהה ביותר עם הפרמטרים  $\theta$  ו- $\gamma$  של היוריסטיקת הסידור, היא סביב הערכים:  $\theta = 0.6$  ו- $\gamma = 0.4$ . נראה כי סביב הנקודות הללו בגרף, השחקן שלנו הוא בעל הדירוג הגבוהה ביותר מבין שאר הנקודות ועל כן נבחר בהם להיות הערכים הקבועים של הפונקציה היוריסטית.

### מסקנות הניסוי:

היוריסטיקה שבחרנו להערכת הילדים, גורמת לסינון חכם של הילדים, שכן אחרת, היינו צפויים לקבל המון הפסדים בשל מהלכים חשובים שהיו מסוננים ומובילים להפסדנו כנגד הסוכן היריב הרגיל.

כמו כן, המסקנה העיקרית היא שהמשקל למהלכים חדשים שנפתחים חשוב באופן דומה למשקל של מהלכים עם ערך כלי גבוהה, עם יתרון קל למשקל של מהלכים חדשים שנפתחים. כעת, נוכל להגדיר את הפונקציה היוריסטית H\_Deepening באופן קבוע:

$$H\_D(s) = \begin{cases} \infty & \text{is\_checkmate}(s) \text{ is True} \\ 1000 & \text{is\_escape\_from\_attack}(s) \text{ is True} \\ & \text{or} \\ & \text{will\_capture\_enemy}(s) \text{ is True} \\ -\infty & \text{is\_3\_repetitions}(s) \text{ is True} \\ -1000 & \text{will\_become\_attack\_by}(s) \text{ is True} \\ 0.6 * \text{NewMoves}(s) + 0.4 * \text{EvalPiece}(s.\text{piece}) & \text{else} \end{cases}$$

ובפונקציה זו נשתמש בסדרת הניסויים באיכות האסטרטגיה.

## ניסויים באיכות האסטרטגיה:

ביצענו טורנירים (10 משחקים לטורניר) של סוכן rb-alphabeta עם העמקה סלקטיבית כנגד יריב קבוע (סוכן הבקרה), **לכל** אחד מהעומקים 2 3 4 5 6, כאשר **לכל** עומק בדקנו שיעור KeepRate שונה: 10% 20% 35% 40% 50% 60% 70%, ללא הגבלת זמן, כשבנוסף, על מנת שכל המשחקים יהיו שונים זה מזה, אם יש לכמה מהלכים שונים את אותם הערכים היוריסטים והם הערכים היוריסטים הטובים ביותר מבין שאר, נבחר מהלך רנדומלי מבין אותם הערכים. דירוג השחקן הוא מספר הניצחונות שלו בטורניר מתוך 10, כאשר תיקו שווה חצי נקודה.

בנוסף, מלבד הקשר שנבדק בין עומק השחקן לערך KeepRate, נמדוד גם את הזמן הממוצע לתור בכל המשחקים, על מנת שנוכל בהמשך להעריך את השחקן גם לפי הזמן.

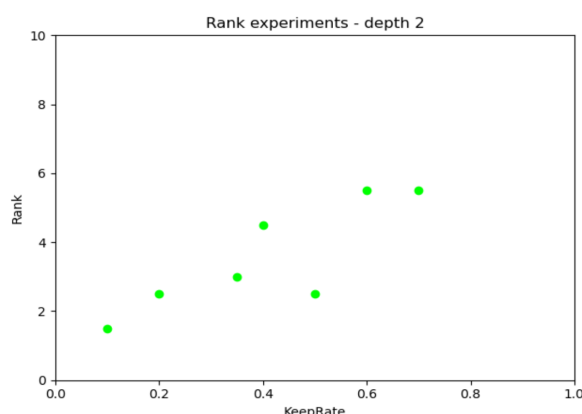
### תוצאות הניסוי:

הגרפים הבאים מתייחסים לאיכות השחקן (rank) אל מול ה-KeepRate, בכל עומק. כל נקודה בגרף, מתארת דירוג בטורניר (כלומר, נקודה = 10 משחקים).

כמו כן, מידע נוסף הוא הזמן הממוצע לתור, כאשר כאן הזמן ממוצע לתור, הוא הזמן הממוצע לתור בכל המשחקים באותו הטורניר בו שיחק. צבע נקודה בגרף מראה את הזמן הממוצע כך ש: ירוק בהיר: פחות משנייה לתור בממוצע, ירוק כהה: פחות מ-2.5 שניות (ומעל שנייה), ירוק זית: פחות מ-7.5 שניות (ומעל 2.5 שניות), צהוב: פחות מ-12.5 שניות, כתום: פחות מ-30 שניות, אדום כהה: פחות מ-45 שניות, אדום: יותר מ-45 שניות לתור.

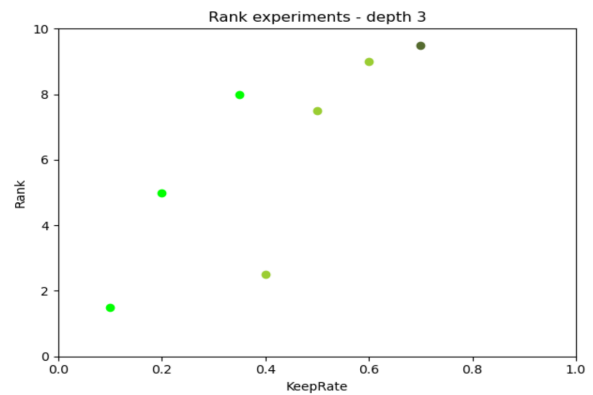
### **עומק 2:**

KeepRate	avg_time_per_turn(sec)	wins	draws	rank
0.1	0.05	0	3	1.5
0.2	0.12	0	5	2.5
0.35	0.21	1	4	3
0.4	0.36	0	9	4.5
0.5	0.44	1	3	2.5
0.6	0.8	2	7	5.5
0.7	0.73	2	7	5.5



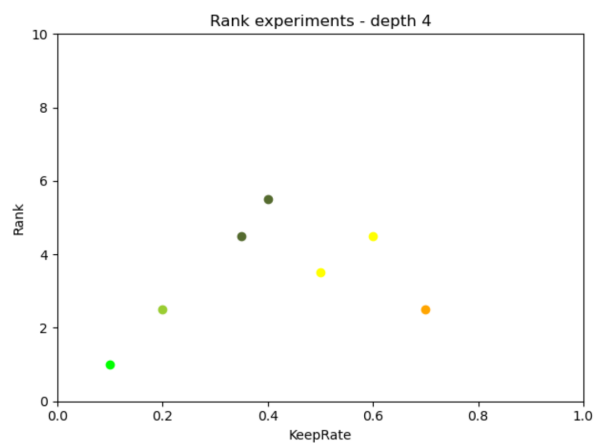
### עומק 3:

KeepRate	avg_time_per_turn(sec)	wins	draws	rank
0.1	0.31	1	1	1.5
0.2	0.41	3	4	5
0.35	0.72	7	2	8
0.4	1.31	5	2	2.5
0.5	1.09	7	1	7.5
0.6	2.1	9	0	9
0.7	2.86	9	1	9.5



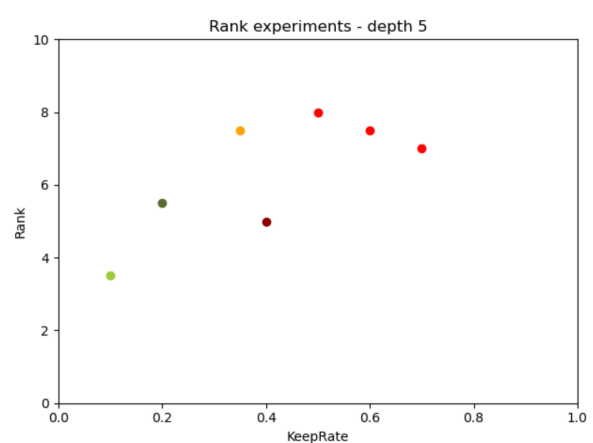
### עומק 4:

KeepRate	avg_time_per_turn(sec)	wins	draws	rank
0.1	0.52	0	2	1
0.2	1.51	1	3	2.5
0.35	3.72	2	5	4.5
0.4	4.9	3	5	5.5
0.5	8.1	2	3	3.5
0.6	9.61	2	5	4.5
0.7	14.83	0	5	2.5



### עומק 5:

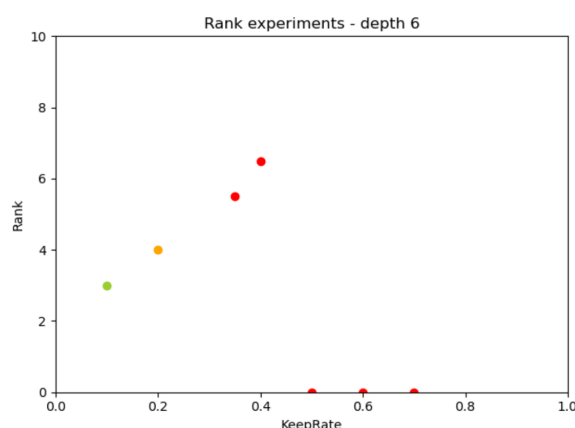
KeepRate	avg_time_per_turn(sec)	wins	draws	rank
0.1	1.22	1	5	3.5
0.2	5.32	3	5	5.5
0.35	25.53	7	1	7.5
0.4	32.61	3	4	5
0.5	54.58	7	2	8
0.6	68.4	5	5	7.5
0.7	129.4	6	2	7



## עומק 6:

KeepRate	avg_time_per_turn(sec)	wins	draws	rank
0.1	2.43	1	2	3
0.2	22.91	2	4	4
0.35	122.33	3	5	5.5
0.4	135.12	4	5	6.5
0.5	NaN	0	0	0
0.6	NaN	0	0	0
0.7	NaN	0	0	0

הערה: בהרצות בעומק 6, החל מ-50% KeepRate, המשחקים שנמדדו ללא הגבלת זמן, נמשכו למעלה מיום שלם ולא הסתיימו (ריצה על חומרה חזקה). משום שמשחק שחמט לא נמשך כמה ימים רצוף, נחשיב את כל המשחקים הללו כהפסד - סוכן שתמיד מפסיד מחוסר זמן.



### ניתוח תוצאות הניסויים ומסקנות הניסויים באיכות האסטרטגיה:

באופן כללי, הגרפים תואמים את הציפיות שלנו: KeepRate גדול יותר < שחקן עם rank גבוהה יותר, משום שמצבים רבים נשארים, שכן ככל שה-KeepRate גדול יותר אנחנו "שואפים" לאלגוריתם הרגיל של alphabeta ללא סינון המצבים. מרבית הגרפים מראים מגמה של עלייה באיכות האסטרטגיה של השחקן ככל שה-KeepRate גדל, מלבד חריגות קטנות שיכולות לנבוע משגיאות מדידה שעלולות לנבוע מכמות המשחקים בטורניר שלא ייצגו בהכרח את היכולת האמיתית של השחקן (למשל בגרף של עומק 4 ואנומליה בודדת בגרף של עומק 3).

אנחנו יכולים לראות שהעומק של שחקן אכן משפיע על ביצועיו יחסית לשאר העומקים, אם כי לא באופן גורף, כלומר, יש מצבים בהם העומק גדול יותר והשחקן נחשב לפחות טוב באותו ה-KeepRate, אך באופן כללי שחקן עמוק יותר שיחק טוב יותר משחקן פחות עמוק עם KeepRate זהה. דרך טובה לראות זאת היא לנתח את האיכות של השחקן היא בקצוות של הניסוי: גרף עומק 2 אל מול גרף עומק 6:

בגרף של עומק 2 ניתן לראות כי ה-rank של השחקן קטן יותר מה-rank **בכל אחד** מה-KeepRate הזחים בגרף של עומק 6. כלומר, העומק אכן משפיע על איכות השחקן. בהתבוננות בזמנים הממוצעים של תור הסוכן בכל טורניר, ניתן לראות כי המסקנות מהניסויים בזמנים תקפות בכל אחד מהניסויים שעשינו באיכות האסטרטגיה, כלומר עומק גדול יותר ו-KeepRate גדול יותר, לוקחים זמן רב יותר לזמן הממוצע לתור ככל שגדלים ערכיהם, כאשר עומק מגדיל את הזמן בצורה מעריכית ו-KeepRate משפיע על מגמת העלייה של הגרף המעריכי. בניסויים אלו ניתן לראות זאת לפי צבעי הנקודות, שבכל גרף של עומק מסוים, ככל ש-KeepRate גדל יותר, הנקודות נהפכות ליותר כהות, ובאופן כללי, ככל שעומק הגרף גדול יותר, כך סה"כ הנקודות כהות יותר ואדומות יותר - כלומר הזמן הממוצע לתור לוקח זמן רב.

עם תום ניסוי זה, ניתן לשערך את סוכן ה-rb-alphabeta עם וריאנט ה-MSD, עם הפרמטרים הטובים ביותר של Depth ו-KeepRate ל-3 סוגי המשחק הנפוצים בשחמט, המבוססים זמן מוגבל: 1-1, 5-5, 10-10.

## פרק 5 - פרק מסכם: דיון בתוצאות, מה חסר, כיוונים להמשך

### המחקר וסיכום:

#### דיון בתוצאות:

משקלול התוצאות ומסקנותיהם, הבנו לעומק יותר ובאופן כמותי, את הקשרים בין KeepRate, העומק לבין איכות השחקן וזמן השחקן.

מהניסוי של הזמנים הסקנו שהאפשרות להשיג ייתרון של עומק על ידי העקמה סלקטיבית חכמה ולא מסננת מדיי היא אכן אפשרית, שכן ראינו שיש בכל עומק טווח ערכים מסויים של KeepRate שמקנה לנו ייתרון על rb-alphabeta הרגיל באותו העומק, מבחינת הזמנים והוא באזור מ-30%. אך כדי שנשיג ייתרון של איכות, הדבר תלוי קודם כל בכך שסינוי הילדים יהיה חכם.

מהניסוי של בדיקת הפרמטרים  $\theta$  ו- $\gamma$  בהיוריסטיקת סידור הילדים לשם הסינון, הסקנו שמתן ערך היוריסטי חכם להעדפת מהלכים להעמקה הוא כזה שהמשקל למהלכים חדשים שנפתחים חשוב באופן דומה למשקל של מהלכים עם ערך כלי גבוהה, עם ייתרון קל למשקל של מהלכים חדשים שנפתחים ובכך הגדרנו את היוריסטיקה לסידור באופן קבוע.

מהניסוי של בדיקת איכות השחקן, ראינו כי KeepRate גדול יותר < שחקן עם rank גבוהה יותר, אך גם גילינו שבעומק גדול מדיי, KeepRate משפיע מאוד על הזמן של השחקן ובפועל גם על איכותו, משום שבמשחקים אמיתיים המוגבלים בזמן הוא עלול להפסיד מחוסר זמן ובכך לפגוע באיכותו.

מ-3 תוצאות הניסויים, ניתן ליצור סוכן המתאים לכל סוג משחק. לפני שנעשה זאת, עלינו להבחין ב-2 דברים חשובים:

1. בידנו הזמן הממוצע לסוכן שנבדק במשחקים רבים בהם שיחק, עם קומבינציה של פרמטרים שונים של KeepRate ו-Depth. על מנת לשערך פרמטרים אלו בהתאם לסוג המשחק המוגבל בזמן, עלינו לשערך את מספר הצעדים הממוצע במשחק שחמט סטדנרטי ומכאן לשערך את 2 הפרמטרים הללו לסוכן אותו נרצה לשלוח למשחק. לבניית הסוכן שחמט, אנו מניחים כי מספר הצעדים במשחק ממוצע בשחמט הוא 60 צעדים בסה"כ, כלומר 30 צעדים לשחקן, לפי הממוצע הכללי של משחקי שחמט.
2. כל הערכת הזמן של הניסויים שביצענו, נכונה אך ורק תחת ארכיטקטורה מסוימת של חומרה. כלומר, ייתכן שעל החומרה עלינו הרצנו, חומרה אחרת תראה תוצאות גרועות יותר או ההפך. לכן, בניית הסוכן נכונה רק לארכיטקטורה בה בדקנו.

#### בניית הסוכן:

כעת, ניתן לשערך את הפרמטרים של הסוכן עבור משחקים: 1-1, 5-5, 10-10, כאשר אנו מניחים שמשחק ממוצע בשחמט: 30 צעדים לשחקן. כל התוצאות מבוססות על תוצאות הניסויים.

#### סוכן 1-1:

ממוצע לצעד נדרש:  $60\text{sec} / 30\text{moves} = 2\text{sec}$ .

ניתן כמובן לשלוח סוכן של KeepRate: 70% עם Depth: 3, שיעשה את העבודה מצויין כפי שראינו בתוצאות ניסויי האסטרטגיה, אך במקרה של 70% יכולנו לשלוח פשוט את rb-alphabeta באותו העומק ולהרוויח יותר.

לכן, נבחר סוכן שבהתאם לסוכן יריב רגיל של rb-alphabeta בעל העומק הטוב ביותר לסוג המשחק 1-1, יכול לבצע עומק גדול יותר באותו מגבלת זמן עם עומק גדול יותר.

עומק אופטימלי ל-rb-alphabeta רגיל: 3, שכן בתוצאות הניסויים של הזמנים ראינו כי rb-alphabeta עושה בממוצע בעומק 3 1.6sec לתור ובעומק 4 עושה 6.6sec לתור ונדרש צעד ממוצע של 2sec, אז נבחר בעומק 3.

עומק רצוי של הסוכן שלנו:  $4 = 1 + 3$



**נעדיף לבחור בעומק הגדול ב-1**, על מנת שה-KeepRate לא יהיה קטן מדי ויסונו המון ילדים. כלומר, ניתן הרי גם לבחור בעומק 5 עם KeepRate של 10%, אך נעדיף KeepRate גבוהה כמה שניתן כדי לא לסכן יותר מדי ולכן נבחר בעומק הגדול ב-1 (הבנה זו היא למעשה מהות הרעיון של השימוש בוריאנט ה-MSD). ה-KeepRate הגדול ביותר במגבלת הזמן: **20%**, ניתן לראות זאת בטבלת הניסויים באיכות אסטרטגיית השחקן בעומק 4, שבעומק 4, הרצוי, ב-20% אנחנו בממוצע של 1.52sec והנדרש למשחק 1-1 הוא 2sec. **בסה"כ:**

**KeepRate: 20% | Depth: 4**

#### **סוכן 5-5:**

ממוצע לצעד נדרש:  $300\text{sec} / 30\text{moves} = 10\text{sec}$ .  
עומק אופטימלי ל-rb-alphabeta רגיל: 4, שכן בתוצאות הניסויים של הזמנים ראינו כי rb-alphabeta עושה בממוצע בעומק 4 6.6sec לתור ובעומק 5 עושה 109sec לתור, לכן נבחר בעומק 4.  
עומק רצוי של הסוכן שלנו:  $5 = 1 + 4$   
ה-KeepRate הגדול ביותר במגבלת הזמן: **20%**, ניתן לראות זאת בטבלת הניסויים באיכות אסטרטגיית השחקן בעומק 5 (באופן דומה להסבר של סוכן 1-1 ממקודם). **בסה"כ:**

**KeepRate: 20% | Depth: 5**

#### **סוכן 10-10:**

ממוצע לצעד נדרש:  $600\text{sec} / 30\text{moves} = 20\text{sec}$ .  
עומק אופטימלי ל-rb-alphabeta רגיל: 4, שכן בתוצאות הניסויים של הזמנים ראינו כי rb-alphabeta עושה בממוצע בעומק 4 6.6sec לתור ובעומק 5 עושה 109sec לתור, אז נבחר בעומק 4.  
עומק רצוי של הסוכן שלנו:  $5 = 1 + 4$   
ה-KeepRate הגדול ביותר במגבלת הזמן: **25%**, ניתן לראות זאת בטבלת הניסויים באיכות אסטרטגיית השחקן בעומק 5. **בסה"כ:**

**KeepRate: 25% | Depth: 5**

בחלק של הסיכום, נראה תוצאות של **טורניר הסיים**: טורנירים שערכנו של סוכן rb-alphabeta עם וריאנט MSD, לכל אחד מ-3 סוגי המשחקים, עם הפרמטרים האופטימליים אותם הסקנו מתוצאות כל הניסויים, כנגד סוכן יריב rb-alphabeta רגיל עם עומק הקטן מאיתנו ב-1.

## **מה חסר:**

### **היוריסטיקות:**

בבניית הסוכן השתמשנו בהיוריסטיקת שחמט (היוריסטיקה להערכת מצב, לא היוריסטיקת ה-H\_Deepening) די סטנדרטית, המתבססת על מיקום וערך הכלי ועם עוד תוספות קלות משלנו. עם זאת, יכולנו לבדוק את הסוכן על היוריסטיקות מתקדמות יותר, שדורשות מחקר מעמיק יותר ולעיתים כוח חישובי רב יותר (לאימון) כמו למשל היוריסטיקות המבוססות על למידה עמוקה. היה מעניין לבדוק את הסוכן שבנינו עם היוריסטיקות אחרות, כשיכולנו אולי לקבל תובנות רבות שאולי לא היינו מקבלים בשימוש בהיוריסטיקה הסטנדרטית. כמו כן, גם שילובי היוריסטיקות שונות בביצוע הניסויים יכלו להיות בדיקה מעניינת, שהרי כל הניסויים שעשינו, השתמשו באותה היוריסטיקה: גם עבור הסוכן הרגיל וגם עבור סוכן הבקרה (היריב). ניסויים שונים ומגוונים של סוכנים שונים עם היוריסטיקות שונות, גם הן, אולי, יכלו להוביל אותנו למסקנות ותובנות שונות מאלו שהסקנו.

## ניסויים:

בשלב הניסויים, הבדיקות שביצענו היו מעטות לעומת העולם האמיתי, כך שאם היינו רוצים להשתמש בסוכן שלנו בטורניר מקצועי, אז בניסויים של כיוון  $\theta$  ו- $\gamma$  ביוריסטיקת H\_Deepening למשל, היינו מעריכים את ציון השחקן (rank) ב-10,000 משחקים ולא ב-10. בנוסף לכך, בסדרת ניסויים כמו של איכות האסטגרטיה, היינו בודקים רמות ספיציפיות יותר של רמת KeepRate, כלומר במקום ערכים בודדים בקפיצות של 0.15, היינו בודקים ערכים רבים בקפיצות של 0.05 למשל, כך יכולנו לקבל תמונה יותר ברורה על הפרמטרים האופטימליים של השחקן. כמו כן, בדיקת הערכת איכות האסטגרטיה של השחקן נעשתה לפי "שחקן בקרה" מסוג יחיד של alphabeta. ביצוע סדרות ניסויים נוספות של שחקני בקרה מסוגים שונים (עומקים שונים ואולי אף וריאנטים שונים של alphabeta), יכלו אולי לתת לנו תמונה ברורה יותר של הסוכן שלנו. עם זאת, כל אלו כרוכים במשאב זמן גדול מאוד, אולי אף חודשים רבים של הרצות, גם אם אלו הרצות נעשות בהרצות מקבילות של הניסויים. נציין כי כל ההרצות של המשחקים בפרויקט נעשו באופן מקבילי (כל פקודה של הרצת משחק נעשתה ב-background "&"), ובחומרה חזקה מאוד ועדיין, זמן ההרצות של הניסויים ארך למעלה מכמה שבועות של הרצה (כולל ניסויים שלא צלחו מסיבות טכניות כמו קריסות או באגים).

## אופטימיזציות:

כפי שצויין בפרק של הגדרת H\_Deepening, אחד האתגרים הגדולים בפרויקט היה ליצור פונקציות יעילות בזמן ככל שניתן, על מנת שהפונקציה היוריסטית H\_Deepening, שנקראת מספר רב מאוד של פעמים, תהיה יעילה בזמן. על ידי כלים כמו מנגנוני cache, מקבילות חכמה ועוד, ניתן להפוך את הפונקציה שהגדרנו למהירה אף עוד יותר ובכך להגדיל את סף ה-KeepRate המינימלי שיכול לגרום לנו להעמקה D בזמן זהה להעמקה D-1 באלגוריתם הרגיל (כפי שמתואר במסקנה 1 בניסוי בזמנים). נציין כי ניסיונות תכנותיים רבים ושימוש ב-Profilers מתוחכמים שמזהים חוסר יעילות נעשו בכתיבת היוריסטיקה, על מנת לייעל אותה ככל שניתן. כך ש-KeepRate יהיה הגדול ביותר, כך שנוכל להעמיק ולהרוויח ייתרון על יריבנו. למשל, בניסויים בזמן הגענו לערך של 36% בעומק 6 להשגת ייתרון על יריב בעומק 5, אך ההגעה לערך הזה לוותה בביצוע ניסויים רבים שלא הניבו תוצאות טובות במיוחד - 15% ו-20% ו-22% ודרשו טיפול תכנותי בפונקציה היוריסטית וביצוע רב מאוד של הניסויים בזמנים עד שהושג ערך של 36% בניסויים המתוארים כאן.

## צפייה וניתוח משחקים של הסוכן:

את הפונקציה H\_Deepening, פיתחנו לפי בדיקה של מצבים רבים בלוח השחמט, כלומר בדקנו את התנהגות הפונקציה בכל מיני סיטואציות שונות של לוחות בשחמט. עם זאת, יכולים להיות המון מצבים שונים בהם הפונקציה שלנו דווקא פוגעת ואולי אף גורמת לאיבוד ילדים שיכולים להועיל לשחקן שלנו והדבר יכול להוביל אף להפסדו, או גם איבוד ילדים של היריב שלנו ובכך פספוס על מהלכים שהיריב מתכנן לעשות לנו. על ידי צפייה של מאות משחקים בהם הפסיד השחקן שלנו, יכול להיות שנוכל להבחין במצבים חשובים שסיננו שלא בחוכמה. למשל, ניתן לקחת משחק בו הפסדנו, עם KeepRate של 50% מהילדים, ולבדוק כל מהלך גרוע שעשינו ולהבין למה סוּן מהלך טוב יותר (אם בכלל). עם כלים כמו debugger, נוכל לשים לב אם יש מהלך שנזרק באלגוריתם (בכל עומק, לא רק בעומק 1) על אף שלא היה צריך להיזרק וכתוצאה מכך נוכל להבין את השגיאה ולשכלל את הפונקציה היוריסטית בהתאם. צפייה כזו ומעקב אחרי כל צעד שגוי יכולה להיות כמובן מעיקה ולקחת חודשים רבים של עבודה ידנית ואבחון של אותם המצבים בהם זרקנו את הילדים שלא היינו אמורים לזרוק. ניתן כמובן גם להשקיע זמן רב בפיתוח של תוכנות מתוחכמות שיבצעו אוטומטיזציה של תהליך צפייה במשחקים שבוחן את אלגוריתם ה-rb-alphabeta עם וריאנט ה-MSD בכל עומק, וזיהוי מצבים שגויים שסיננו, על ידי כך שהתוכנה תבדוק בכל צעד שעשינו, מה סוכן שחמט מאוד חכם היה בוחר ובדקת אם סיננו את המהלך החכם הזה או לא, ומודיע בהתאם ובכך נסיק בהתאם כיצד לשכלל את הפונקציה H\_Deepening.

## כיוונים להמשך המחקר:

ישנם כיוונים רבים עליהם חשבנו שאפשר לשכלל את הסוכן המבוסס rb-alphabeta עם העמקה סלקטיבית. אלו 2 הכיוונים המעניינים ביותר:

### העמקה סלקטיבית דינאמית:

סוכן המבוסס rb-alphabeta עם וריאנט ה-MSD, יבחר ב-KeepRate ראשוני. ככל שיעמיק, ה-KeepRate ייקטן יותר ויותר, כך שהמשמעות היא שבתחילה יישארו מספר רב של צעדים ולאחר מכן בכל שלב בהעמקה, ינופו מספר קטן יותר ויותר של ילדים ככל שמעמיקים. נוכל לקבוע את קצב הפחתה של KeepRate כך שכל שנעמיק, ה-KeepRate ייקטן בהתאמה לקצב הירידה.

לדוגמה, עבור קצב הפחתה 20%, נתחיל עם השארת 90% מהילדים בעומק הראשון, לכל ילד מאלו, נסנן 70% מהילדים ולכל אחד מאלו נסנן 60% וכך הלאה עד לעומק שנעמיק.

הדבר נובע מהאינטואיציה הבאה: כאשר שחקן אנושי מתבונן על הצעדים שיש לו לעשות, בדרך כלל הוא יבחן את כל הצעדים האפשריים לעומק 1 ויסנן את הילדים כפי שאנחנו עושים והצגנו גם בתחילה, אך ככל שהוא חוקר מהלך לעומק יותר, למשל בוחן את תגובת היריב למהלך שלו, הוא יבחן שיעור מסוים ממהלכי היריב שאותם יעשה, בדרך כלל מהלכים שמאוד קרובים למהלך שלו ולא יתבונן על מספר גדול יותר של צעדים שהיריב יכול לעשות.

אם קביעת קצב הפחתה של ה-KeepRate לא נעשה בתבונה, הדבר כמובן יכול להוות בעיה. קצב הפחתה גדול מדי יגרום למשל על ויתור מהלכים עתידיים של היריב כמו אכילת כלי חשוב שלנו ובכך הדבר יכול לגרום לנו לעשות צעד לא חכם שנובע מערכים היוריסטים מוטעים ולאסטרטגיה לא נכונה. קצב הפחתה קטן מדי, יגרום לבזבוז זמן רב מדי מכפי שתכננו על בדיקות היוריסטיות של הילדים בכל עומק, שעלול לגרום לנו להפסד מחוסר זמן ולהפסיד את הייתרון של העומק כנגד יריבנו.

### עומק והעמקה סלקטיבית דינמית בהתאמה לסיטואציות חמות:

גרסה אפשרית נוספת של העמקה סלקטיבית דינאמית, מורכבת אף יותר, יכולה להתרחש גם לפי מצבי המשחק השונים. כלומר, הסוכן מגדיר העמקה סלקטיבית בשיעור מסוים כמקסימלי, כאשר בכל פעם שימצא מהלך "חם", למשל כמו שח שהוא מבצע או שח שעושים עליו, KeepRate יהיה קטן מאוד, כדי שלא יפספס מהלכים אפשריים בסיטואציה ה"חמה", ובכל פעם שימצא מהלך שיראה כמהלך גרוע, למשל מהלך "סתמי", KeepRate יהיה גדול יותר, לכל היותר עד גודל השיעור המקסימלי שהוגדר מראש.

האתגר בכך הוא היכולת להבחין היוריסטית בסיטואציות "חמות" ו"קרות" בשחמט במהלך החיפוש. ניתן להוסיף לכך העמקה משתנה, כאשר אם נתקלים בסיטואציות "חמות", נגדיל את עומק החיפוש וגם נגדיל את KeepRate וההפך עבור סיטואציות "קרות", ובכך ליצור סוכן ש-2 הפרמטרים של KeepRate ו-Depth יהיו דינאמיים ומותאמים לסיטואציות שונות במשחק.

### פונקציה היוריסטית H Deepening דינאמית:

באלגוריתם rb-alphabeta עם וריאנט MSD, בכל שלב אנחנו מסדרים את הילדים לפי כללים קבועים מסוימים. הפרמטרים של המשקלים  $\theta$  ו- $\gamma$  בפונקציה היוריסטית אצלנו הם קבועים, לאחר שבניסויים מצאנו את הערכים הטובים ביותר עבורם.

אנחנו סבורים שהכללים שקבענו לא בהכרח יהיו נכונים לכל הסיטואציות במשחק, אם כי לרובם. בסיטואציות כמו מלך עם 2 צריחים אל מול מלך של היריב, לא נרצה לסנן מהלכים שיובילו את הצריח שלנו לא לעשות מט על המלך, רק משום שהם לא פותחים עבורו צעדים רבים (דבר עליו אחראי ה- $\theta$ ) ולכן נרצה להקטין את המשקל שלו בסיטואציה זו.

האתגר הוא היכולת לזהות סיטואציות כאלו ולהבין אילו ערכים הכי טובים של  $\theta$  ו- $\gamma$  יכולים להתאים. בסה"כ, נוכל לנסות לגרום לסידור ילדים הקרוב ביותר לאופטימלי בשביל ההעמקה סלקטיבית.

## סיכום:

בפרויקט זה פיתחנו את אלגוריתם ה-rb-alphabeta והוספנו לו את וריאנט ה-Minimax Selective Deepening, על מנת לייצר סוכן זהה לאלגוריתם ה-rb-alphabeta הקיים, אך עם אופצית העמקה גדולה יותר תחת אותה מגבלת הזמן, בתמורה ליותר על פיתוח חלק מהמהלכים (סינון מצבים). בדיקת וריאנט ה-MSD, התבצע על משחק השחמט. יצירת סוכן שחמט היא משימה מאתגרת מאוד, קיימים אלגוריתמים רבים ומגוונים, היריסטיקות רבות וגישות רבות לאתגר זה. זיהוי המהלכים שכדאי לסנן, מתבצע על ידי היריסטיקה H\_Deepening המשערכת לכל מהלך את הפוטנציאל שלו, כלומר כמה שווה להמשיך ולהתעמק בו והיא אתגר בפני עצמו. כתיבת היריסטיקה היוותה את אחד האתגרים הגדולים מ-2 סיבות:

1. על היריסטיקה להיות מדויקת - קשה לאדם להעריך במדויק אילו מדדים חשובים יותר וחשובים פחות במשחק כה מורכב כמו שחמט ולכן מתן משקלים לכל מדד הוא הכרחי. השגת המשקלים האופטימליים דורשת מספר רב של ניסויים (16 טורנירים, 160 משחקים) ולוקחת זמן **רב מאוד**.
2. על היריסטיקה להיות מהירה - היריסטיקה של סינון המצבים מתבצעת בכל פעם שמפתחים מצב חדש שהוא לא סופי (או בעומק המרבי). החלק המאתגר הוא לגרום לסיבוכיות זמן טובה ביותר, כאשר הדבר אינו טריוויאלי הן תיאורטית והן תכנותנית, בשל העובדה שלעיתים המון מימושים בתוכנה (API) מוסתרים מהמשתמש ונדרש לעיתים למצוא דרכים עקיפות וניסויים רבים על מנת לבצע דברים באופן מהיר - מספיקה קריאה מיותרת לפונקציה אחת שתימשך מילישנייה, כדי לגרום לתוספת של עשרות שניות בביצוע האלגוריתם בעומק 6.

כמו כן, סדרת הניסויים שביצענו כדי למצוא את הקשר בין הזמן, עומק, ערך KeepRate ודירוג השחקן, היא דבר מאתגר גם כן. הורצו מאות משחקים במשך מספר שבועות. לבסוף, רוב התוצאות שקיבלנו אכן תאמו את מה שציפינו עם תוספת של מידע כמותי חשוב לקביעת הפרמטרים של הסוכן בכל אחד מסוגי המשחקים המוגבלים בזמן. בסה"כ, הורצו אלפי משחקים, משום שחלק מהאתגר של יצירת סוכן השחמט הוא ביצוע ניסויים רבים שחלקם התבררו כשגויים בשל שגיאה תכנותית או קריסה.

## טורניר הסיום:

לסיום, ביצענו הרצה של 3 טורנירים, לכל אחד מסוגי המשחק 1-1, 5-5, 10-10, עם הסוכן שיצרנו עבור כל אחד מסוגי משחקים אלו כנגד סוכן רגיל עם עומק קטן יותר, תחת אותם מגבלות הזמן אותם חישבנו קודם לכן עבור סוכן ה-rb-alphabeta הרגיל, ולהלן התוצאות של טורניר הסיום:

### טורניר 1-1:

- סוכן: **Depth: 4** | **KeepRate: 20%**
- סוכן יריב: **Depth: 3**
- דירוג הסוכן: 2.5, מהם 2 ניצחונות, 7 הפסדים ותיקו אחד.

### טורניר 5-5:

- סוכן: **Depth: 5** | **KeepRate: 20%**
- סוכן יריב: **Depth: 4**
- דירוג הסוכן: 7, מהם 7 ניצחונות, 2 הפסדים מחוסר זמן והפסד יחיד רגיל.

### טורניר 10-10:

- סוכן: **Depth: 5** | **KeepRate: 25%**
- סוכן יריב: **Depth: 4**
- דירוג הסוכן: 5, מהם 5 ניצחונות, 4 הפסדים מחוסר זמן והפסד יחיד רגיל.

בסה"כ, נראה שהסוכן שיצרנו, המעמיק לעומק גדול יותר מיריבו, תחת אותה מגבלת זמן אך עם סינון מצבים, אכן מראה שיפור. עם זאת, על ידי אופטימיזציות מתקדמות של הפונקציה H\_Deepening, ועל ידי למידת בצורה מעמיקה יותר כפי שנכתב בפרק זה, אנו סבורים שהסוכן שלנו יוכל להיות טוב הרבה יותר מיריבו, תחת הנחת הבסיס שלנו שעומק גדול יותר ו-KeepRate מתאים, מוביל למהלכים טובים יותר ולשחקן בעל דירוג טוב יותר.