דו"ח מסכם

תרגיל בית 2

יונתן בתן

302279138

Yonibettan@gmail.com

עמרי פרוינד

301695490

omrifro@gamil.com

חלק א'

סעיף 1:

בקוד

סעיף 2:

בקוד

סעיף 3:

בהינתן gameState מסוים הסוכן ReflexPlayer מחשב את כל הצעדים האפשריים שלו, כלומר מתוך הקבוצה $\{NORT, SOUTH, EAST, WEST\}$, לאחר מכן מחשב את הערך של צעד על המצב הנתון לכל צעד אפשרי באמצעות היוריסטיקה ולבסוף בוחר בצורה רנדומלית צעד מבין הצעדים המניבים את הערך המקסימלי.

ההיוריסטיקה בה הוא משתמש מחזירה את ה- score הנתון למצב עליה היא פועלת, score זה הוא הניקוד הנוכחי ב-state כפי שהוא מופיע ב- GUI.

נבחין כי היוריסטיקה זו מניבה סוכן הבוחר את הפעולה אשר מניבה ניקוד מקסימלי מבלי לחשוב קדימה ולנסות מצבים שאינם אופטימליים לצעד הבא אבל יניבו תוצאה טובה יותר בהמשך.

חלק ב'

סעיף 1:

$$h(s)=\begin{cases} s.score & if\ \ isTerminalState(s) \\ s.score+\dfrac{1}{s.\min\_food\_dist}+\dfrac{1}{s.\min\_capsule\_dist}+s.ghost\_factor\cdot\dfrac{1}{\min\_ghost\_dist}+s.randomness & else \end{cases}$$

- הפרמטר $s.score$ מייצג את הניקוד הנוכחי של המצב כפי שהוא מופיע ב- GUI
- הפרמטר $s.min\_food\_dist$ מייצג את המרחק לחתיכת אוכל הקרובה ביותר
- הפרמטר $s.min\_capsule\_dist$ מייצג את המרחק לקפסולה הקרובה ביותר
- הפרמטר $s.ghost\_factor = \{30, -30\}$ הינו חיובי אם פקמן אכל קפסולה ושלילי אחרת
- הפרמטר $s.\min\_ghost\_dist$ מייצג את המרחק לרוח הקרובה ביותר
- הפרמטר $s.randomnes$ מוסיף 1 לערך היוריסטי בהסתברות 0.85

סעיף 2:

תחילה נבחין כי הפרמטר $s.score$ מופיע בחישוב התוצאה ושאר המספרים הינם מספרים הקטנים או שווים ל-1 ולכן מבצעים "fine tuning" להיוריסטיקה scoreEvaluationFunction.

- כפי שלמדנו, עבור מצבים סופיים ההיוריסטיקה מחזירה את הערך שמחזירה פונקציית התועלת
- הפרמטר $s.score$ מייצג את ניקוד המצב הנוכחי ולכן הוא פרמטר חשוב בפונקציה שלנו שכן מצב של ניצחון מכיל כ-500 נקודות יותר מקודמו ולכן נרצה לתת לכך משקל רב בערך היוריסטי.
- לכל אחד מהפרמטרים $s.\min\_\{food, capsule, ghost\}\_dist$ נרצה לתת בונוס גבוהה כאשר המרחק קטן ובונוס נמוך כאשר המרחק גדל ובנוסף עבור מרחקים גדולים ההבדל בין מצבים שונים נהיה זניח מבחינת פרמטר זה ולכן בחרנו בפונקציה $\frac{1}{x}$.
  - עבור המרחק מהרוח המינימלית אנו מבדילים בין המצב בו פקמן אכל קפסולה ולכן הרוח פגיעה למקרה בו פקמן לא אכל קפסולה.
    במקרה הראשון נרצה להתקרב לרוח על מנת לאכול אותה ולצבור נקודות ומיכון שאכילת רוח מזכה בפי 20 יותר נקודות מאשר חתיכת אוכל רצוי לתת פקטור של 20 אל מול הבונוסים של חתיכות האוכל, לאחר משחק עם הפרמטרים גילינו שפקטור 30 נותן את התוצאה הטובה ביותר
- הפרמטר $s.randomnes$ נועד למנוע היתקעות במינימום מקומי, כלומר כאשר ישנם 2 מצבים סמוכים כך שבמעבר לאחד מהם השני נהיה הטוב ביותר אנחנו ניתקע והאלגוריתם לא יסיים את ריצתו ולכן תוספת של רנדומליות מאפשרת לנו לצאת ממצבים אלו
  - נשים לב כי ללא הפרמטר $s.randomness$ לא היינו צריכים להפריד בין מצבים סופיים לכאלו שאינם סופיים שכן כל ערכי המינימום מאותחלים ל- ∞ ולכן עבור מצבים סופיים ערכים אלו לא מתעדכנים והיינו מקבלים $h(s) = s.score$

סעיף 3:

בקוד

חלק ג'

סעיף 1:

ההנחה שלנו הינה שהסוכנים משחקים בזה אחר זה במשחק ובפרט פקמן משחק בהתאם לצעד האחרון של רוח מס' 1 שהיא משחקת בהתאם לצעד האחרון של רוח מס' 2 וכ"ו.

הנחה זו כמובן אינה נכונה שכן הרוחות אינן בהכרח מגיבות אחת לשנייה, יתכן והן פועלות באופן בלתי תלוי אחת מהשנייה ואם במקרה הן כן מגיבות אחת לשנייה לא ניתן להבטיח שרוח מס' 1 תגיב דווקא לרוח מס' 2 ולא להפך למשל.

סעיף 2:

בקוד

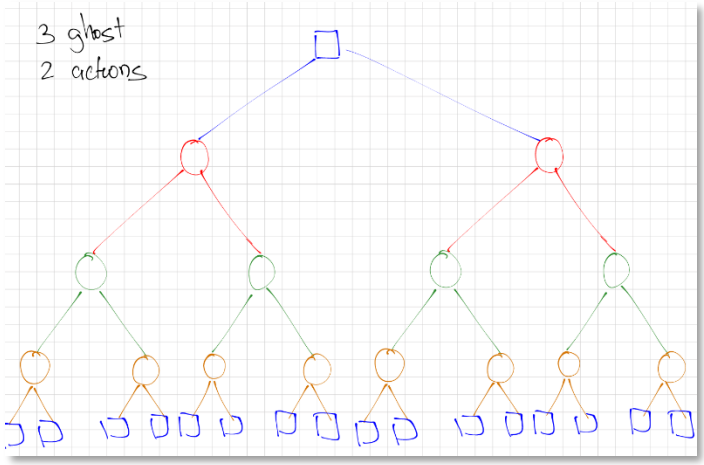
סעיף 3:

במימוש הנוכחי, צומת בשכבת מינימום i מייצג את הפעולה של הרוח ה- i בלבד על $state$ המכיל בתוכו כבר את הפעולות של הרוח ה- j לכל $j > i$ באותו התור.

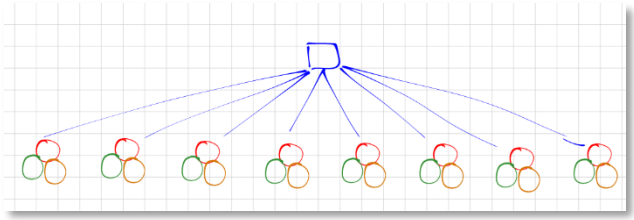
במימוש החדש תהיה לנו שכבת מינימום יחידה אשר כל מצב בה מייצג את פעולות כל הרוחות בתור מסוים.

יתרונות:

- ניתן להשתמש באלגוריתם מינימקס פשוט של 2 שחקנים אותו למדנו
- יש צורך לבצע פחות חישובים של מינימקס כי יש פחות מצבים אפשריים בשכבת המינימום מאשר בסכום שכבות המינימום
  - נניח שיש לנו k רוחות וכל סוכן יכול לבצע $action$ פעולות ואנו רוצים לחשב מינימקס עם הגבלת עומק $d = 1$
  - במימוש הנוכחי נצטרך לחשב $action^1 + \cdots + action^k = \sum_{i=1}^{k} action^i$ ערכי מינימקס.



  - במימוש החדש יש לנו רק $action^k$ מצבים בשכבת המינימום בעומק $d = 1$ (כל מצב מייצג מסלול מרוח אדומה לרוח כתומה) ולכן צריך לחשב $action^k$ ערכי מינימקס



חסרונות:

- מורכב יותר למימוש
  - חישוב מצבי הבנים דורש בניית מצב לכל קומבינציה אפשרית של פעולות הרוחות וזה מורכב יותר מאשר ליצור בן עבור כל פעולה של רוח יחידה כפי שעשינו.

חלק ד'

סעיף 1:

לא, רק פקמן נחשב כצומת מקסימום ולכן נכנס לחלק הראשון של התנאי ואילו כל שאר הרוחות נחשבות לצמתי מינימום ולכן נכנסים לחלק השני של התנאי באלגוריתם.

- גיזום α:
  - בפיתוח צומת מינימום s, אם אחד הבנים שלו מניב ערך קטן מ- α אזי מובטח כי $minimax(s) \leq \alpha$ ולכן ענף זה ייגזם בדיוק כמו במבנה עץ המכיל שכבת מינימום יחידה לכל שכבת מקסימום.

- גיזום β:
  - בפיתוח צומת מקסימום s, אם אחד הבנים שלו מניב ערך גדול מ- β אזי מובטח כי $minimax(s) \geq \beta$ ולכן ענף זה ייגזם בדיוק כמו במבנה עץ המכיל שכבת מינימום יחידה לכל שכבת מקסימום.

סה"כ אופן פעולת הגיזום אינו משתנה ולכן מבנה העץ החדש שהגדרנו אינו משפיע על האלגוריתם אלפא-ביתא.

סעיף 2:

בקוד

סעיף 3:

א.

כן.
כיוון שאנו ממיינים את הבנים בעזרת ההיוריסטיקה אנו בעצם מתקרבים לגיזום האופטימלי (במקרה של היוריסטיקה מושלמת היינו מקבלים גיזום אופטימלי) ולכן יתבצעו יותר גיזומים, משמע פחות פיתוחים ולכן זמן חישוב קצר יותר.

ב.

ברמה העקרונית לא, אך בפועל יתכנו מהלכים שונים.
"ברמה העקרונית" משמעותו שאם מדובר תיאורטית באותה ריצה, כלומר תנועת הרוחות זהה ואנו בודקים רק את התנהגות הסוכנים, ובנוסף ערכי המינימקס של הבנים הינה סדרה מונוטונית עולה ממש (אין 2 בנים עם ערך מינימקס מקסימלי זהה) וההיוריסטיקה דטרמניסטית אזי במקרה זה בחירת הצעדים של הסוכנים תהיה זהה.

לפי משפט הנכונות של אלפא-ביתא האלגוריתם מחזיר את ערך המינימקס של תת העץ שלו ולכן מובטח שהערך של כל אחד מהבנים יהיה ערך המינימקס שלו ולכן לכאורה אין שוני בין המהלכים שבוחרים הסוכנים.
כיוון שעבור בנים שונים המניבים ערך מינימקס מקסימלי זהה אלגוריתמי בחירת הצעד בוחרים צעד באופן רנדומלי (כך מימשנו את אלגוריתם בחירת הצעד) לכן סביר שבמקרה בו יש יותר מבן יחיד המניב ערך מינימקס מקסימלי 2 הסוכנים יפלו על ערך רנדומלי שונה ויבחרו בצעד שונה.
בנוסף אם מריצים את 2 הסוכנים (ריצות שונות) יתכנו תנועות שונות של הרוחות שכן איננו רשאים להניח דבר על התפלגות התנועה של הרוחות וכיוון שההיוריסטיקה שלנו מתחשבת במיקום הרוחות אזי גם במקרה זה יתכנו בחירת מהלכים שונים בין הסוכנים.

ולבסוף כיוון שההיוריסטיקה שלנו אינה דטרמיניסטית יתכנו מהלכים שונים בין הסוכנים

חלק ה'

סעיף 1:

בקוד

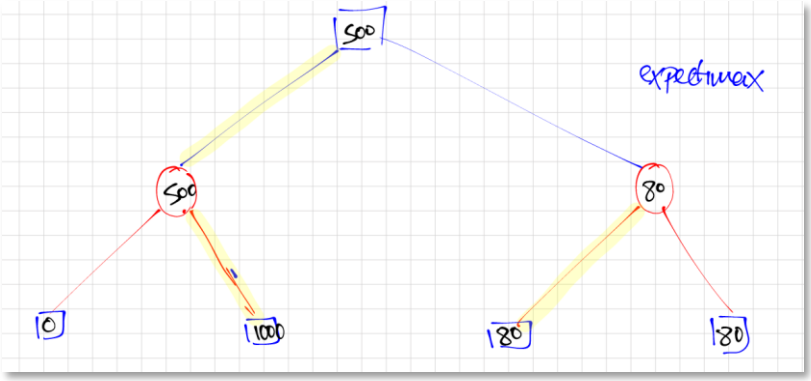
סעיף 2:

סוכנים המשתמשים באסטרטגיית מינימקס מניחים שהיריב בוחר בצעד הגרוע ביותר עבורנו, כלומר היריב עושה הכל על מנת לפגוע בנו ככל הניתן.

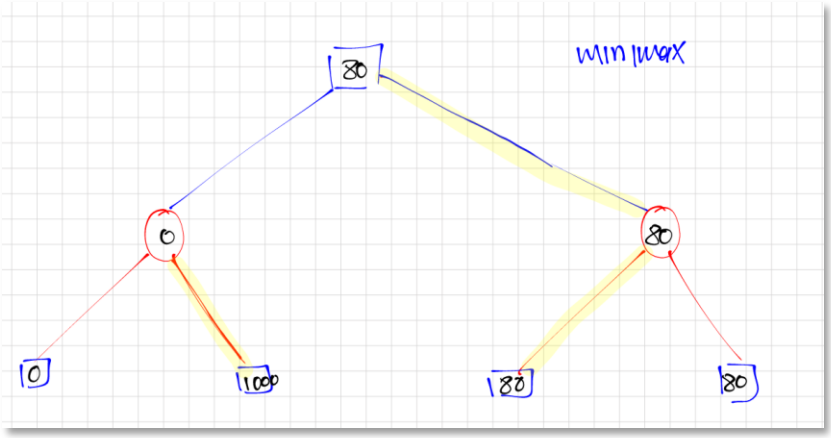
סוכן אקספקטימקס לעומת זאת חושב בצורה רציונלית יותר עבור משחקים הסתברותיים, בסעיף זה אנו מניחים שהרוח בוחרת את הצעד הבא בהתפלגות אחידה על מרחב הצעדים האפשריים שלה, כלומר צמתי הרוחות הם צמתים הסתברותיים ולכן יותר מציאותי לנבא את הצעד של הרוח כתוחלת של אפשרויות התנועה שלה בהתפלגות אחידה מאשר כצעד שיפגע בנו בצורה המקסימלית.

אנו מאמינים כי שימוש באקספקטימקס עבור רוחות רנדומליות ייתן תוצאות טובות יותר שכן מינימקס נותן חסם תחתון לפתרון האופטימלי אבל בהינתן שהרוח רנדומלית יתכן וקיימים פתרונות טובים יותר שאותם אנו עלולים לפספס עם מינימקס\אלפא-ביתא.

לדוגמא עבור רוח יחידה, נניח שהרוח בוחרת את הצעד המסומן אזי אקספקטימקס בוחר את הצעד המוביל לתוצאה המקסימלית במשחק.



ואילו מינימקס היה בוחר את הצעד השני אשר מניב תוצאה טובה הרבה פחות.



חלק ו'

סעיף 1:

- התפלגות התנועה:

ההתפלגות תלויה בפרמטר הסתברותי p.

נסמן את מספר הפעולות החוקיות ב- $\#legal\_actions$ ואת מספר הפעולות אשר מניבות ערך אופטימלי ב- $\#best\_actions$.

הרוח בוחרת בצעד אופטימלי (מתקרבת לפקמן אם אינה פגיעה ומתרחקת אחרת) בהסתברות
$P(optimal) = \frac{p}{\#best\_actions} + \frac{1-p}{\#actions}$
ובצעד שאינו אופטימלי בהסתברות
$P(not\_optimal) = \frac{1-p}{\#actions}$.

כפי שניתן לראות אין העדפה לצעד מסוים בתוך קבוצת הצעדים האופטימליים, כנ"ל עבור הצעדים הלא אופטימליים.

לצורך הבהרה נראה מה קורה בערכי הקצה:

- עבור $p = 1$ נקבל
$P(optimal\_decsision) = \frac{1}{\#best\_actions}$
$P(not\_optimal\_decsision) = 0$
כלומר הסוכן בוחר צעד בהתפלגות אחידה מבין הצעדים האופטימליים.

- עבור $p = 0$ נקבל
$P(optimal\_decsision) = \frac{1}{\#actions}$
$P(not\_optimal\_decsision) = \frac{1}{\#actions}$
כלומר הסוכן בוחר צעד בהתפלגות אחידה מבין כל הצעדים האפשריים.

כברירת מחדל מתקיים $p = 0.8$ ובכל מקום בקוד השימוש ברוח הנ"ל היא תמיד עם ערך ברירת המחדל.

- האסטרטגיה:

הרוח בוחרת בצורה רנדומלית צעד מבין הצעדים האופטימליים בהסתברות גבוהה ובוחרת בצורה רנדומלית צעד מבין הצעדים שאינם אופטימליים בהסתברות נמוכה.

צעד אופטימלי הוא צעד אשר מקרב את הרוח לפקמן באופן מקסימלי כאשר הרוח אינה פגיעה, כלומר במקרים בו פקמן לא אכל קפסולה או שחלף מספיק זמן מאז, או צעד אשר מרחיק את הרוח מפקמן באופן מקסימלי כאשר הרוח כן פגיעה.

סעיף 2:

בקוד

סעיף 3:

המימוש שלנו כמעט זהה בין 2 הסוכנים, לכל אחד מהם יש מתודה יחידה אשר מחזירה את הצעד הבא, ונעזרת במתודה rbExpectimax לצורך חישוב הערך של צמתי מינימום.

ההבדל היחיד הוא ש- RandomExpectimaxAgent משתמש במתודה זו עם הפרמטר 'random_ghost' ואילו DirectionalExpectimaxAgent משתמש במתודה עם פרמטר 'directional_ghost'.

מתודה זו מתנהגת כמו מינימקס עבור צמתי מקסימום ומחשבת את ערך התוחלת עבור צמתי מינימום כתלות בהתפלגות התנועה של טיפוס הרוח אשר מתקבל כארגומנט.

החישוב מתבצע לפי הפילוג אשר הוסבר בסעיף 1 שאל חלק זה.

סעיף 4:

- שיפור 1:
  נחשב את המרחק מהרוח לפקמן תוך התחשבות במרחק האמתי למשל ע"י שימוש ב- GameState.getWall() במקום מרחק מהנהטן שהרי יתכן שרוח תצטרך "לחזור אחורה" על מנת לעבור קיר מסוים ומרחק מנהטן מפספס עניין זה.

- שיפור 2:
  תוצאה אופטימלית לא תהיה תלויה אך ורק במרחק לפקמן אלא גם תיקח בחשבון את המרחק לחתיכות אוכל וקפסולות בידיעה שפקמן רוצה להגיע למקומות אלו ולכן תוכל לנצל זאת לטובתה.

חלק ז'

כל הרצה של המשחק באמצעות סוכן כלשהוא תניב בסיומו ניקוד מספרי. על מנת שנוכל להשוות בצורה טובה יותר בין סוכנים, נקבע את סוג הלוח, מספר הרוחות והתנהגותן (יישארו ללא שינוי לכל הסוכנים).

מרכיבים שונים במהלך המשחק הם מרכיבים סטטיסטיים ולכן התוצאה הסופית שתתקבל אינה קבועה. למעשה אם נריץ n הרצות נקבל וקטור אקראי – מדגם של n מתוך אוכלוסיית התוצאות האפשריות. על מנת להשוות בין שני סוכנים נשתמש במבחן Student's t-test על המדגמים שקיבלנו.

עבור $n \to \infty$ התוצאות אמנם מתפלגות נורמלית אך לצורך המבחן מכיוון שיש ברשותנו n דגימות נשתמש בסטטיסטי של התפלגות t על המדגם.

יהיו A_1, A_2 שני סוכנים, נגדיר השערת האפס והשערה חלופית:

$$\begin{cases} H_0 : \mu_1 > \mu_2 \\ H_1 : \mu_1 \le \mu_2 \end{cases}$$

עפ"י מבחן t נדחה את השערת האפס אם מתקיים $\frac{\bar{X}_1 - \bar{X}_2}{\sqrt{S_1^2 + S_2^2 / n}} < t_{(n-1),1-\alpha}$ כאשר מספר

ההרצות (דרגות החופש) $n = 20$, השונות מתוך המדגם הינה $S_k^{\,2} = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{X}_k)^2$ והסף $\alpha = 0.05$.

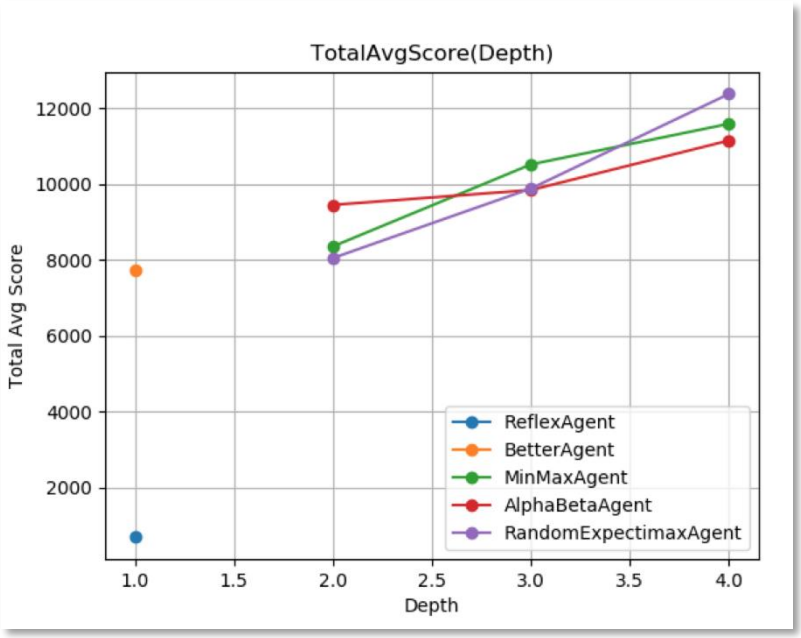
חלק ח'

סעיף 1:

בקוד

סעיף 2:

א.



ב.

Score Table

Agent	depth=1	depth=2	depth=3	depth=4
ReflexAgent	691.2857142857142			
BetterAgent	7740.714285714285			
MinMaxAgent		8357.857142857143	10522.285714285714	11592.857142857141
AlphaBetaAgent		9454.714285714284	9847.142857142857	11152.285714285714
RandomExpectimaxAgent		8052.285714285716	9886.57142857143	12381.857142857141

סעיף 3:

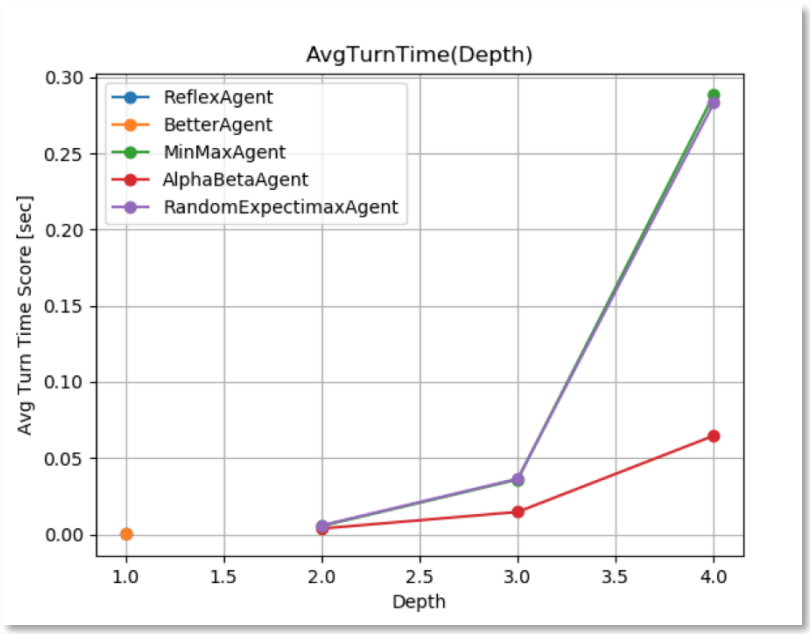
עבור הסוכן רפלקס קיבלנו את התוצאה המצופה, ניתן לראות שהביצועים של ההיוריסטיקה המשופרת שלנו אכן טובים יותר מההיוריסטיקה הבסיסית, דבר זה אינו מפתיע אותנו שכן בנינו אותה מראש כך שתהיה טובה יותר.

האלגוריתמים אלפא-ביתא ומינימקס הינם זהים מבחינת איכות הפתרון שלהם, שכן ההבדל בניהם הינו בזמן הריצה ומספר הפיתוחים, ולכן כמצופה אנו רואים תוצאות מאוד דומות בין השניים.

כפי שכבר הוסבר בשלב מוקדם יותר בדוח האלגוריתם אקספקטימקס נותן לנו פתרון קצת יותר "ריאלי" שכן הרוחות לא תמיד מבצעות את הצעד הגרוע ביותר עבורנו, כפי שמינימקס\אלפא-ביתא היו מצפים, ולכן ניתן לצפות מצד אחד לקבל תוצאות טובות יותר עבור אקספקטימקס למשל אם לרוח 2 צעדים אפשריים ושניהם נותנים ערך אקספקטימקס יחסית דומה במקרה זה התוחלת "מעדנת" את ההחלטה שהסוכן לוקח אך במקרה אחר בו צעד אחד של הרוח מעולה עבורנו ואילו הצעד השני הינו אסון במקרה זה מינימקס יניב החלטה טובה יותר שכן הוא יוצא מנקודת הנחה שהרוח תבצע את הצעד הגרוע ביותר עבורנו וזה בדיוק מה שאנו רואים בגרף, כלומר ישנם מקרים בהם אקספקטימקס מתעלה על מינימקס ולהפך.

סעיף 4:

א.



ב.

Time Table

Agent	depth=1	depth=2	depth=3	depth=4
ReflexAgent	0.00011099074600834519			
BetterAgent	0.0003564636634625545			
MinMaxAgent		0.005395868062065866	0.03597770848020868	0.2882491518820357
AlphaBetaAgent		0.0037176512968693514	0.014634655453485567	0.06466116002962823
RandomExpectimaxAgent		0.005789807476051831	0.0364018611691398	0.28303091473559655

סעיף 5:

כאן התוצאות כבר הרבה יותר חד משמעיות.

עבור הסוכן רפלקס ניתן לראות כי אנו מקבלים תוצאה זהה ומינימלית עבור ההיוריסטיקה הבסיסית והמשופרת שכן האלגוריתם הוא זהה וכל ששינינו הוא הדרך בה אנו מחשבים את הערך ההיוריסטי, ובנוסף אנו לא מפתחים עץ לכל מהלך אלא מריצים חישוב פשוט לצורך הערכה של טיב המצבים הבאים.

הסוכנים מינימקס ואקספקטימקס מבצעים את אותו אלגוריתם, ההבדל היחיד בניהם הוא השימוש שלהם בתוצאה, מינימקס מחשב מקסימום ומינימום ואילו אקספקטימקס מחשב מקסימום ותוחלת, ולכן כמצופה אנו מקבלים ביצועים זהים.

וכמובן אלפא-ביתא שהינו הגרסה המהירה של מינימקס, כמצופה, נותן לנו את הביצועים הטובים ביותר לכל עומק.

בנוסף ניתן להבחין שככל ש- d גדל כך חישוב כל צעד נעשה איטי יותר שכן יש לפתח יותר צמתים ולרדת עמוק יותר בעץ המשחק לכל צעד, נקודה זו משתלבת עם התיאוריה ואינה מפתיעה אותנו כלל.

נקודה מעניינת נוספת היא שהגדלת הגבלת העומק משפיעה הרבה יותר (כמעט 2 סדרי גודל) על מינימקס ואקספקטימקס לעומת אלפא-ביתא שכן ככל שהעומק גדול יותר כל הגיזום אפקטיבי יותר כיוון שלכל ענף שנגזם יש יותר צאצאים. כמובן שבכל זאת נקבל פגיעה כלשהי בביצועים כיוון שתחילה הגיזום אינו תמיד אופטימלי וגם אם כן לפחות ענף אחד יצטרך לרדת עמוק יותר בחישוב כל צעד ולכן אנו עדיין מצפים לפגיעה בביצועים אם כי לא חמורה כמו עבור 2 האלגוריתמים האחרים.

סעיף 6:

Expectimax Table

agent - result_type	RandomGhost	DirectionalGhost
RandomExpectimaxAgent - avg_score	2139.2	1311.8
DirectionalExpectimaxAgent - avg_score	1803.4	1935.8
RandomExpectimaxAgent - avg_turn_time	0.6945473630566242	0.7490055000747986
DirectionalExpectimaxAgent - avg_turn_time	0.6219373358693698	0.8313097872617596

```
(ai_hw1) ybettan hw2 (devel) $
```

כפי שניתן להבחין המצב המניב את התוצאה הנמוכה ביותר הינו הסוכן RandomExpectimaxAgent עם הרוח DirectionalGhost.

דבר זה מתיישב עם התיאוריה שלנו שכן RandomExpectimaxAgent מיועד לרוץ עם RandomGhost ולכן ציפינו לקבל תוצאה גבוהה, מאותן סיבות DirectionalExpectimaxAgnet אשר רץ עם DirectionalGhost מניב תוצאה גבוהה.

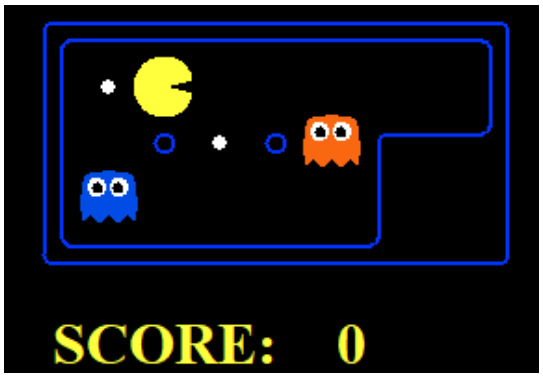
הסוכן DirectionalExpectimaxAgnet מניח שהרוח תעשה צעד גרוע עבורו כלומר תתקרב כאשר היא חזקה או תתרחק כאשר היא חלשה ולכן הנחת העבודה שלו מחמירה יותר מאשר RandomExpectimaxAgent ולכן הוא מתמודד יפה על רוחות אשר לוקחות החלטות פחות רעות עבורו.

מבחינה אינטואיטיבית, אך לא מדויקת, DirectionalExpectimaxAgnet הוא סוג של סוכן מינימקס אם יוצאים מנקודת הנחה שההתקרבות של הרוח היא הצעד הגרוע ביותר עבורו ולכן מניב תוצאה טובה גם כאשר הרוח הינה רנדומלית ולא בהכרח פוגעת בנו באופן מקסימלי.

- הריצה התבצעה עם 2 רוחות

סעיף 7:

לא התייחסנו כאן לסוכנים RflexAgent ו- BetterAgent כיוון שהם אינן רצים עם מגבלת עומק ולכן עבורם המגבלה של $d = 4$ המצוין בשאלה אינה רלוונטית




```
Agents preformence on minimaxClassic layout
+----------------------+--------------------+
|        Agent         |       Score        |
+----------------------+--------------------+
|     MinMaxAgent      | 368.57142857142856 |
|    AlphaBetaAgent    | 367.57142857142856 |
| RandomExpectimaxAgent | 515.1428571428571  |
+----------------------+--------------------+
```

ניתן לשים לב שתוצאת הסוכן RandomExpectimaxAgent טובה מזו של MinimaxAgent. הסיבה לכך נובעת מהעבודה שעבור לוח קטן מאוד (ביחס לעומק החיפוש 4) כמו minimaxClassic עליו בחנו את הסוכנים, ההשפעה של ההחלטה אותה תבצע כל רוח כבר בתור הראשון קריטית לניצחון (או להפסד) במשחק. MinimaxAgent מניח שהרוחות יקבלו את ההחלטה הגרועה ביותר עבור Pacman לעומת RandomExpectimaxAgent המניח הסתברות שווה לכל ההחלטות ואכן ההסתברות להחלטה הגרועה ביותר קטנה כך שבמשך 7 משחקים קורים יותר הפסדים.

סעיף 8:

לא התייחסנו כאן לסוכנים RflexAgent ו- BetterAgent כיוון שהם אינן רצים עם מגבלת עומק ולכן עבורם המגבלה של $d = 4$ המצוין בשאלה אינה רלוונטית


```
Agents preformence on trappedClassic layout
+----------------------+-------------------+
|        Agent         |       Score       |
+----------------------+-------------------+
|     MinMaxAgent      |       -501.0      |
|    AlphaBetaAgent    |       -501.0      |
| RandomExpectimaxAgent | 384.2857142857143 |
+----------------------+-------------------+
```

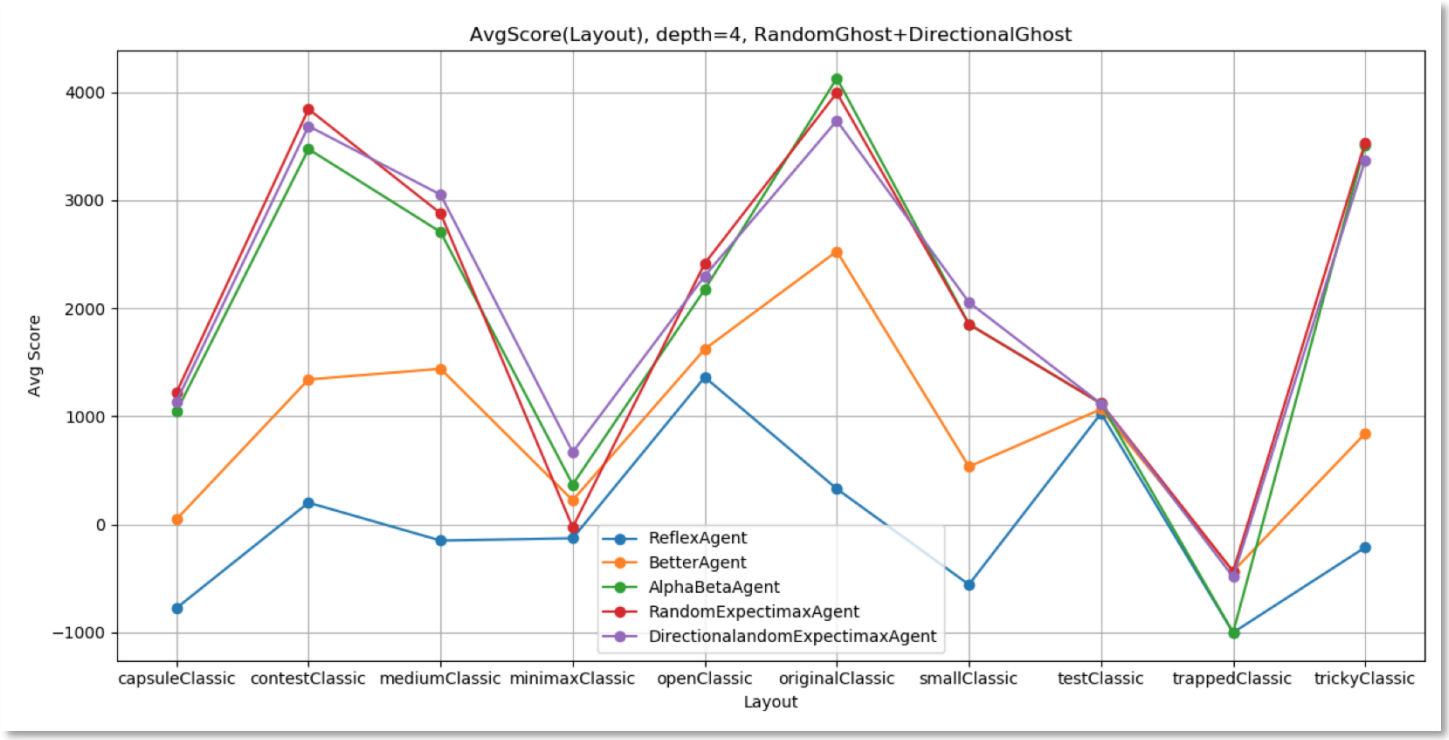
התשובה בסעיף זה דומה לתשובה מסעיף 7 רק שבמקרה זה נכנס גורם נוסף. פרט לגודלו הקטן של הלוח, עבור חישוב בעומק 4 הסוכן רואה כביכול את סוף המשחק. סוכן הMinimaxAgent מניח שהרוחות יסגרו עליו ומבין שלא יכול לנצח – לכן, הציון הגבוה ביותר שיקבל יהיה אם ילך הכי פחות לפני שיתקל ברוח (נפסל בכוונה). סוכן ה-RandomExpectimaxAgent לעומתו רואה סיכוי (הסתברותי) שהרוחות לא יסגרו עליו ובמקרה זה הרוח לניצחון עולה וגורם לו לנסות לנצח וברוב המקרים גם להצליח.

סעיף 9:

- **חלק מהטענות נכתבו כאן בתמצות כיוון שהורחבו בסעיפים המתאימים להם בשלבים מוקדמים יותר בדו"ח**

- כפי שראינו והסברנו בסעיפים קודמים ReflexAgent אינו נבדק עם מגבלת עומק, MinimaxAgent, AlphBetaAgent ו- RandomExpectimaxAgent מקבלים תוצאות טובות יותר ככל שמגדילים את העומק אך נקודה זו באה על חשבון הביצועים שכן כעט חישוב כל צעד הוא איטי יותר. בנוסף, ראינו כי AlphaBetaAgnet מושפע בצורה חלשה יותר מהגדלת עומק החיפוש מאשר MinimaxAgent ו- AlphaBetaAgent שכן ככל שהעומק גדול יותר כך הגיזום גוזם יותר צאצאים בתת העץ הנגזם.

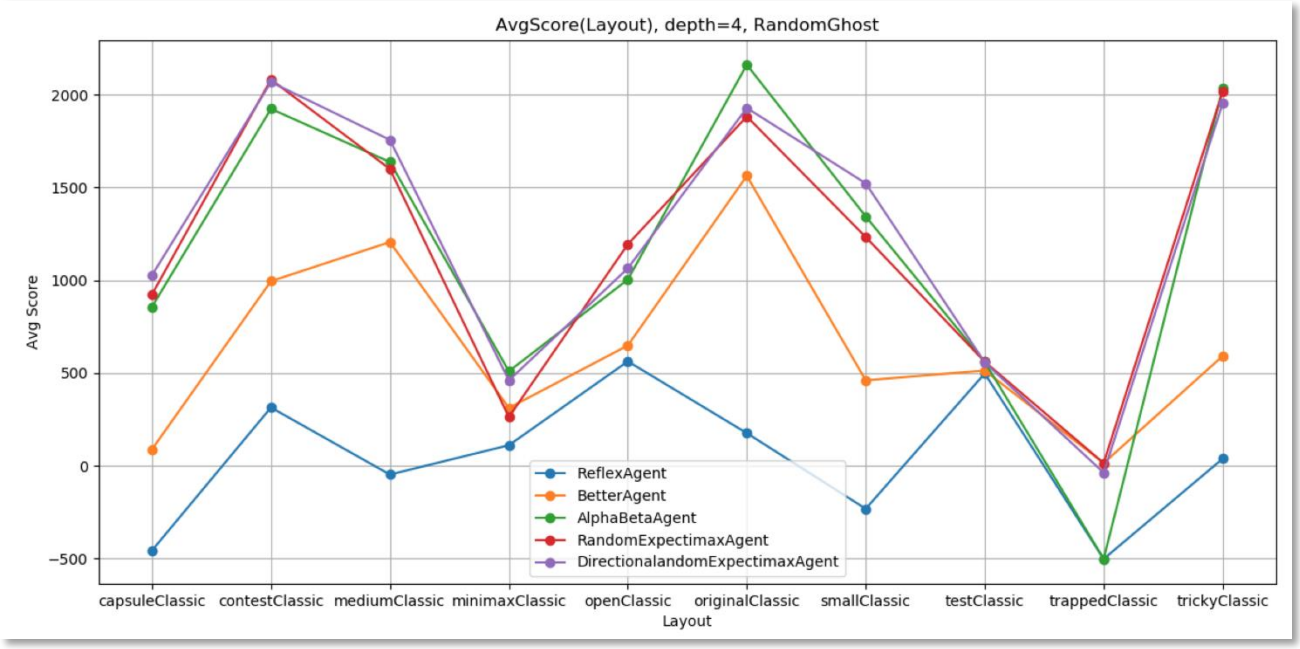
- נראה איך כל לוח משפיע על הסוכנים בעזרת הגרף הבא המתאר את התוצאה של כל סוכן כפונקציה של הלוח עליו הוא עובד, התוצאה הינה הסכום של רוח רנדומלית ורוח כיוונית.



- ניתן לראות כי ההיוריסטיקה שלנו מתגברת על ההיוריסטיקה הבסיסית בכל אחד מהלוחות וזאת כיוון שתכננו את ההיוריסטיקה שלנו בקפידה בשלבים מוקדמים של התרגיל בכדי להבטיח שכל האלגוריתמים שלנו, אשר מסתמכים על היוריסטיקה זאת, יתנו תוצאות טובות.

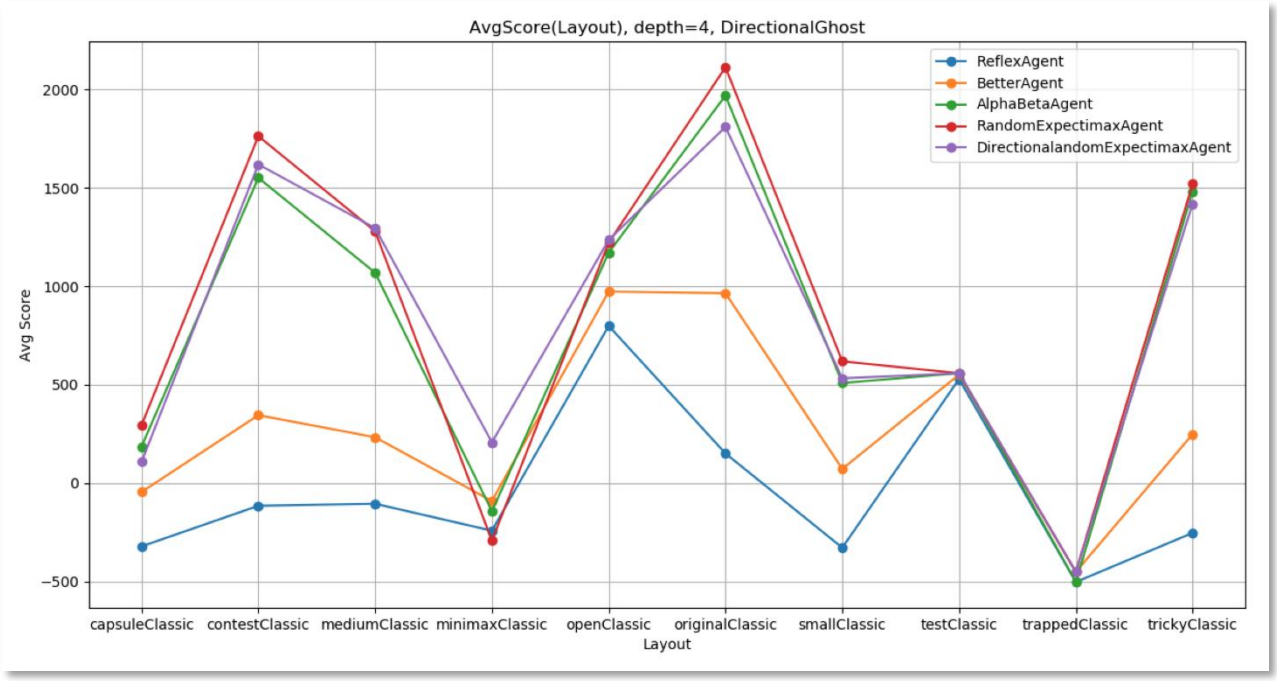
- נשים לב בי הלוח testClassic גורם לכל הסוכנים להניב את אותה תוצאה $500 \approx$ (צריך לחלק ב-2), וזאת כיוון שהלוח יחסית קטן, מכיל רוח יחידה לכל היותר ולכן הסכנה היחידה היא להיאכל ע"י הרוח ועבור רוח יחידה גם הסוכנים הכי פשוטים שלנו אשר סופרים את ה- score יבחרו לא ללכת למצב בו הרוח אכלה אותנו כיוון שהפסד הניקוד מפגישת רוח שווה בערך לניקוד הכולל של אכילת כל המזון בלוח.

- נבחין כי בממוצע על 2 סוגי הרוחות הסוכנים AlphaBeta, RandomExpectimax, DirectionalExpectimax מתנהגים בצורה יחסית דומה, למעט trappedClassic שם סוכני Expectimax טובים יותר מ- AlphaBeta כפי שהסברנו בסעיף הקודם. נצלול קצת פנימה לכל סוג רוח על מנת להבין נקודה זו לעומק.



- עבור לוח מספיק קטן AlphaBeta יכול לפתח את עץ המשחק בשלמותו ולכן רואה שבמקרה הגרוע הרוחות סוגרות עליו לכן "מתאבד" על מנת להפסיד כמה שפחות נקודות על תנועה לעומת סוכני ה- Expectimax אשר מבינים שבהסתברות טובה הרוחות לא יסגרו עליו ולכן מנסה לנצח ובכך מגדיל את התוצאה הסופית שלו.

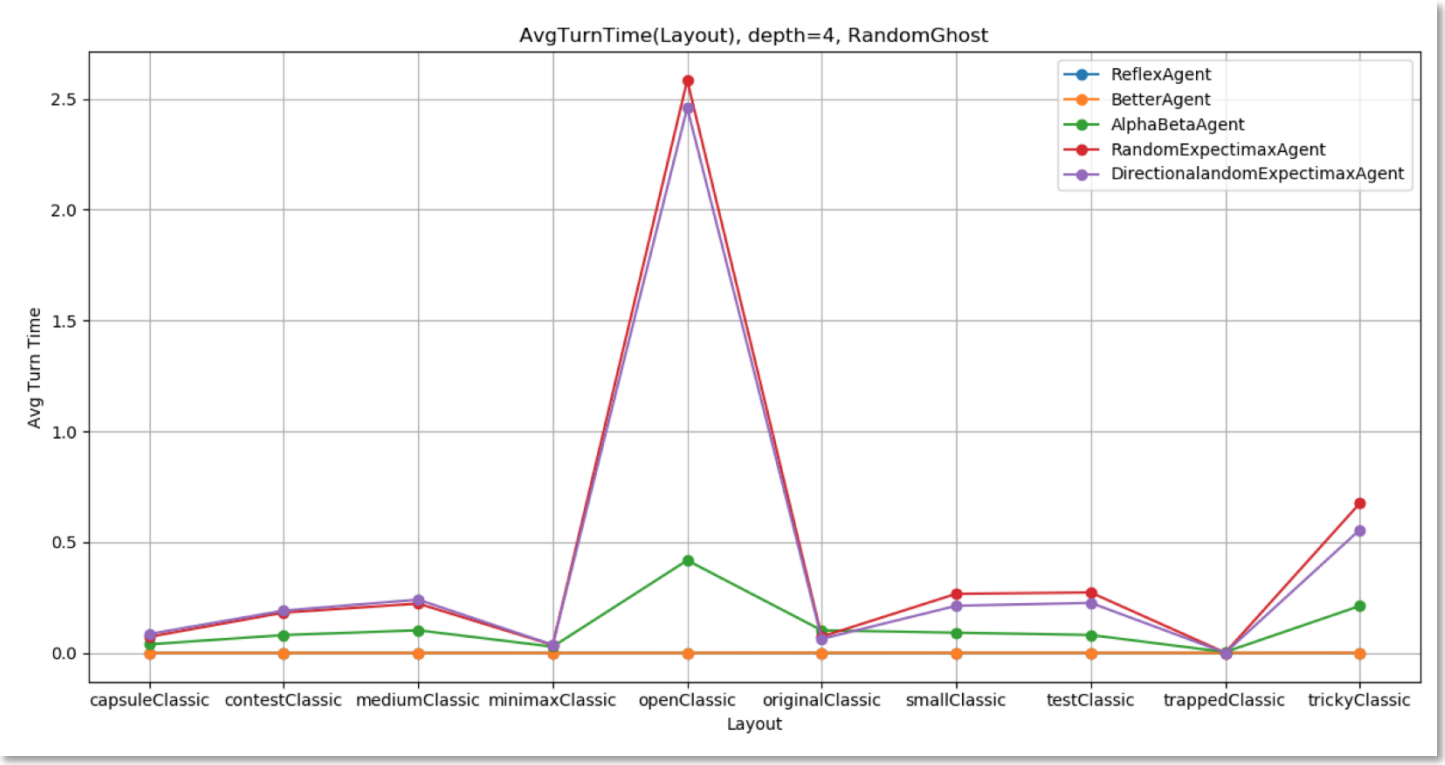
- דבר נוסף מעניין שקורא פה הוא ש- BetterAgent מספיק חכם כדי להגבר על מכשול זה כיוון שהוא לא מנבא את סוף המשחק אלא רק מנסה לאכול אוכל ולברוח מרוחות ולכן התנהגות זו מזדהה עם סוכני ה- Expectimax בלוח זה.



- כצפוי עבור רוחות Directional הרוחות מעצם מבצעות את "הצעד הגרוע ביותר" עבור הסוכן ולכן AlphaBeta מנבא את ההתנהגות נכונה ובמקרה זה סוכני ה- Expectimax אינם טובים ממנו שכן לפקמן אין סיכוי ללא תלות בסוכן

- כפי שראינו והוסבר בחלק ח' מגבלת העומק משפיע על כל הסוכנים התלויים בה, כלומר DirectionalExpectimaxAgent, AlphaBetaAgent, RandomExpectimaxAgent, ככל שהעומק גדל כך התוצאות של 3 הסוכנים הנ"ל משתפרת וזאת מכיוון שככל שהעומק גדל כך הסוכן "צופה" יותר מהלכים קדימה (למעט trappedClassic שם צפיית מהלכים קדימה דווקא מכשילה חלק מהסוכנים) ולכל עם היוריסטיקה טובה הוא יהיה סוכן טוב יותר.
אילו הייתה מגבלת זמן במקום מגבלת עומק היינו ממשים את האלגוריתמים בצורת iterative deepening עם שילוב של שמירת סדר הפיתוח האופטימלי לכל עומק על מנת ליצור גיזום אופטימלי באיטרציה הבאה.

- שמנו לב כי גם ללוחות יש השפעה על משך הזמן שלוקח לסוכן לחשב את הצעד הבא.



- מאוד בולט לעין כי openClassic מגדיל משמעותית את זמן חישוב צעד ממוצע, הסיבה לכך היא שבלוח זה אין קירות כלומר כמעט בכל צעד לכל סוכן יש 4 כיוונים אפשריים לתנועה בשונה מלוחות המכילים קירות שם לרוב לכל סוכן יש אפשרות אחת או אולי 2 להמשך תנועה, דבר זה מגדיל בצורה אקספוננציאלית את עץ המשחק המפותח לכל צעד ולכן מעט מאוד את אלגוריתם בחירת הצעד.
הלוח trickyClassic מכיל אזור פתוח ולכן גם הוא נפגע מאותם סיבות אך באופן פחות חזק.
סוכנים אשר אינם מפתחים עץ משחק כמו ReflexAgent, BetterAgent אינם נפגעים מבעיה זו.

חלק ט'

לטובת תכנון סוכן מיטבי לתחרות השתמשנו במידע שצברנו במהלך התרגיל עד כה, ובפרט בגרפים מחלק ח'.
על פי הנתונים היה ניתן להבחין (למעט מקרים בודדים) בשיפור התוצאה ככל שמגדילים את העומק - לכל הלוחות ולכל הסוכנים.
היות והתחרות מוגבלת בעומק, בחרנו את עומק השחקן להיות העומק המקסימאלי המותר - עומק 4.

הבדל משמעותי נוסף אליו שמנו לב הוא ההבדלים בתוצאות הסוכנים השונים כתלות בשני גורמים מרכזיים:

- סוג הלוח – ה-Layout
- סוג הרוחות – Directional/Random

סוכן התחרות מנסה לדלות מידע אודות סוג הרוחות והלוח אנו רצים ובעזרת מידע זה לבחור את הסוכן האופטימלי, מבין הסוכנים שפיתחנו, עבור סיטואציה זו.

בנוסף יתכן ושחקן התחרות יחליף את הסוכן בו הוא משתמש באמצע משחק בהתאם למצבים מסוימים.

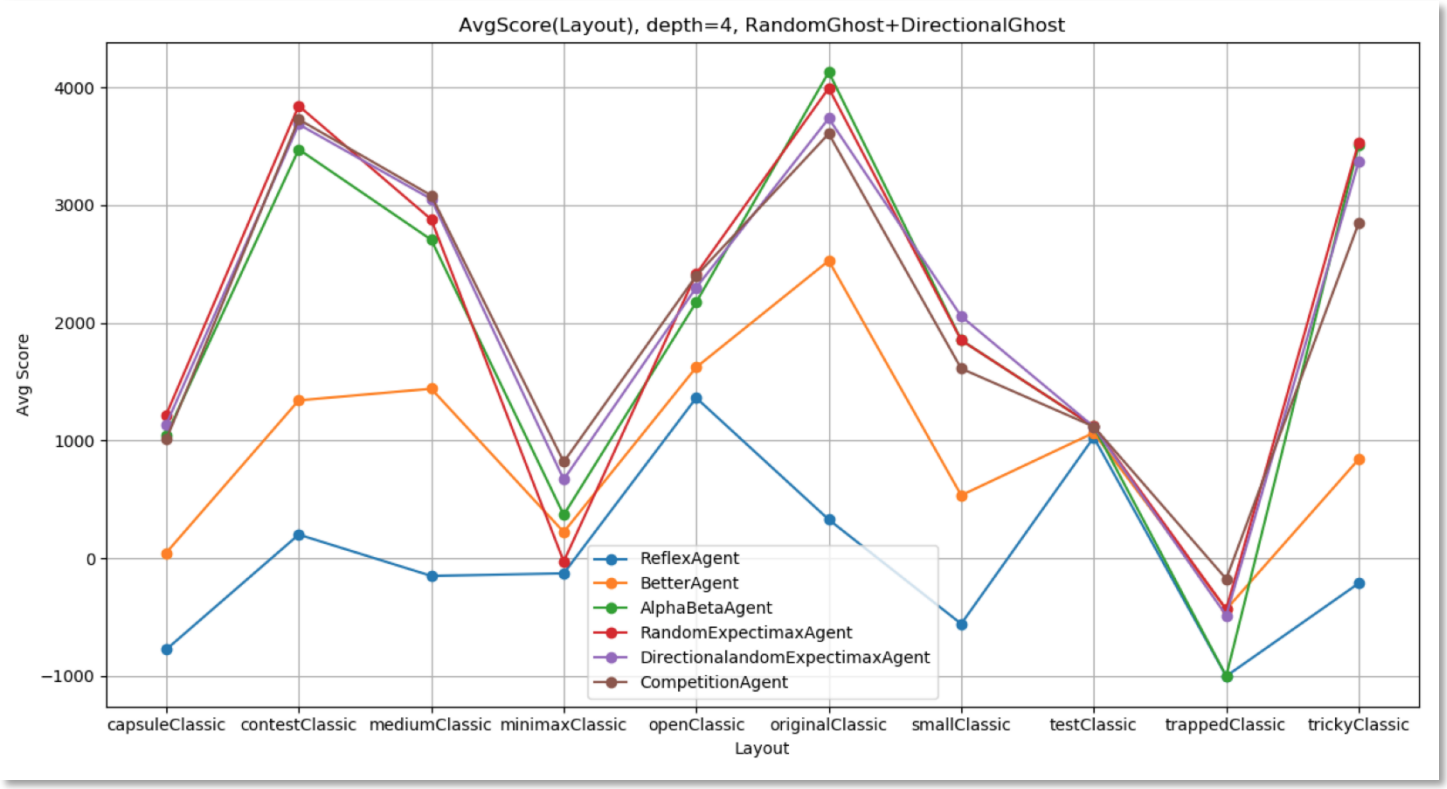
<u>שערוך הלוח</u> – מצב המשחק (gameState) נתון לנו בתור פרמטר, לכן השתמשנו במתודה ()getWalls על מנת למצוא התאמה בין הגובה והרוחב של רשימת הקירות לבין הלוחות שאנו מכירים. מכיוון שייתכנו לוחות שאנו לא מכירים, השארנו אפשרות דיפולטית של unknown בשני גדלים, גדול וקטן.

<u>שערוך הרוחות</u> – משימה זו לא פשוטה משום שגם RandomGhost וגם DirectionalGhost בוחרות אקראית מבין המהלכים האפשריים לכל רוח כאשר עבור DirectionalGhost מדובר בהסתברות גבוה יותר לצעד המקרב את הרוח לפקמן לעומת RandomGhost בה ההתסברויות שוות. על מנת לשערך יצרנו סוכן מסוג DirectionalGhost המקבל את המצב וע"י כך שאנחנו מבקשים ממנו לקבל את הצעד הבא שלו ומשווים לצעד שהרוח ביצעה בפועל אנחנו מצליחים לקבל אינדיקציה לא רעה על סוג הרוח מולה אנו מתמודדים.

נסביר בקצרה על הבחירות המרכזיות שהסוכן שלנו עושה, ישנן עוד מקרי קצה אשר לא יפורטו כאן אך ניתן לראותם בקוד עצמו.

- הסוכן הדיפולטי איתו אנו רצים הינו AlphaBetaAgent בעומק 4, במידה ולא נגיע למסקנה חד משמעית אנו נשאר עם סוכן זה.
- עבור הלוח trappedClassic ראינו ש- AlphaBetaAgent לא מספק את הסחורה ולכן כאשר נזהה שאנו רצים על לוח זה הסוכן יוחלף להיות RandomExpectimax בעומק 4.
- עבור הלוח openClassic ראינו שהסוכנים אשר מפתחים עץ משחק מלא אינם עומדים במגבלת הזמן ולכן היינו צריכים להתפשר, עבור מקרים בהם הרוח רחוקה (מרחק גדול יותר מעומק הפיתוח) נבחר להשתמש ב- AlphaBetaAgent בעומק 3 ואילו במצבים בהם הרוחות קרובות נבחר ב- RandomExpectimax בעומק פיתוח 4.

לסיום ניתן לראות שאנו מקבלים ביצועים לא רעים בכלל עבור CompetitionAgent:



ובנוסף ניתן לראות כי אנו עומדים במגבלת הזמנים עבור כל אחד מהלוחות: