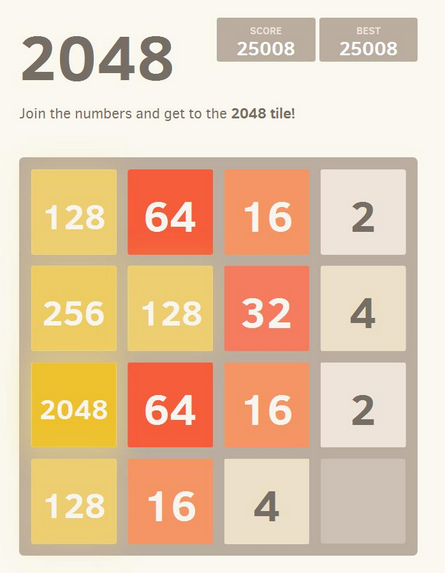
# אלגוריתם אבולוציוני

# לפיתוח שחקן למשחק 2048



# ירדן פלג

# נעה לוי

# תוכן עניינים

* [על המשחק 2048](#_המשחק_2048)
  + [לוח המשחק](#_לוח_המשחק)
  + [מהלך במשחק](#_מהלך_המשחק)
  + [הפסד במשחק](#_הפסד_במשחק)
  + [קצת על המשחק](#_קצת_על_המשחק)
* [שחקנים קיימים ל2048](#_שחקנים_קיימים_ל-2048)
* [ייצוג הפתרון](#_ייצוג_פתרון_(Solver))
* [תיאור האלגוריתם](#_תיאור_האלגוריתם)
  + [האלגוריתם](#_האלגוריתם__(Standard)
  + [פרמטרים](#_פרמטרים)
* [היוריסטיקות](#_היוריסטיקות)
* [סקירת הקוד](#_סקירת_התוצאות) והאופרטורים שיושמו באבולוציה
* [תוצאות](#_תוצאות)
  + [אופן בדיקת התוצאות](#_אופן_בדיקות_התוצאות)
  + [סקירת התוצאות](#_סקירת_התוצאות)
    - ריצה ראשונה
    - ריצה שניה
    - ריצה שלישית
    - ריצה רביעית
    - ריצה חמישית
    - ריצה שישית
* [ממצאים](#_מסקנות) מעניינים
* ביביליוגרפיה

# המשחק 2048

## לוח המשחק

לוח משבצות של 4\*4.

בכל שלב של המשחק, כל משבצת ריקה או מכילה אריח עם ערך מספרי מחזקות 2. כלומר, ערכים אפשריים לאריחים הם 2,4,8,16,32,64,128,256,512,1024,2048,4096 ועוד.

## מטרת המשחק

להגיע ללוח בו מופיע האריח 2048. כמו כן, קיימות ווריאציות רבות למשחק המאפשרות המשך משחק לאחר האריח 2048, ולהגיע לחזקות גבוהות יותר. גרסת המשחק שלנו מאפשרת זאת גם היא. תיאורתית, בגירסה שכזו, ניתן להגיע לכל היותר לאריח בעל ערך 217=131072 (מעשית הדבר לא סביר כלל ותלוי מאוד בגורם האקראי של הופעת אריחים וערכם).

## מהלך המשחק

* בכל צעד במשחק, השחקן יכול להניע את הלוח ימינה, שמאלה, למעלה, או למטה. לדוגמה, כאשר השחקן מניע את הלוח ימינה, כל האריחים מחליקים ימינה אל המשבצות הפנויות בכיוון זה, וכל זוג אריחים בעלי ערך זהה הנפגשים, מוחלפים באריח עם חזקה אחת גדולה יותר של 2.
* עבור יותר מ-2 אריחים באותה שורה/עמודה שמוזזים לכיוון הקיר, האריחים שנמצאים קרוב יותר לקיר מתאחדים.
* לאחר כל צעד, נוסף אריח אקראי במשבצת פנוייה: בעל הערך 2 בהסתברות של 0.9 או הערך 4 בהסתברות של 0.1. מיקומו של האריח החדש נבחר בהתפלגות אחידה מתוך כל המקומות הפנויים.

## הפסד במשחק

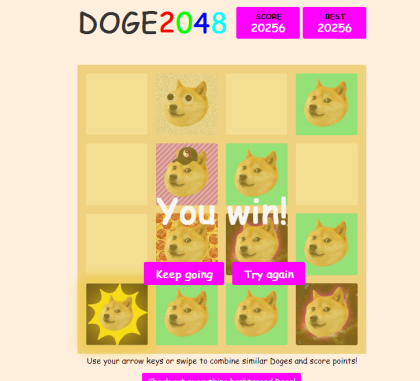
מפסידים במשחק כאשר אף תנועה (ימינה, שמאלה,למעלה, למטה) לא תשנה את מצב הלוח.

## קצת על מקור המשחק [1]

ב- 6 לפברואר 2014 שוחרר המשחק Threes על ידי Asher Vollmer,Greg Wohlwend ו- Jimmy Hinson.

כעבור 21 יום, ב- 27 לפברואר יצא משחק "מאוד" דומה בשם 1024.

רק עשרה ימים לאחר מכן, בחור בן 19 בשם Gabriele Cirulli שיחרר את המשחק 2048.

לטענתו, מאחר וקיבל את ההשראה למשחק מהמשחק 1024 ו- Threes והרגיש לא בנוח להרוויח ממנו כסף, הוא החליט לפתח אותו כ open source. ע"י שימוש בקוד פתוח שהוא פירסם, התחילו אנשים רבים ליצור עותקים שונים ורבים של המשחק.

כיום למשחק יש צורות שונות ומשונות, ממשחקים עם לוחות גדולים יותר מ- 4\*4 למשחקים עם תמונות של כלבים במקום מספרים. הרעיון בכולם דומה - איחוד אריחים עד להשגת האריח הנכסף מהסוג ה- 11 (11^2 = 2048).

# שחקנים קיימים ל-2048

בעקבות פופלריות המשחק, נוצר שרשור ארוך באתר stack-overflow [4], אליו העלו מתכנתים שונים שחקני AI שפותרים את המשחק, ואת התוצאות הטובות ביותר אליהן הגיעו באמצעותם.

לשם השוואה ונקודת ייחוס להצלחת השחקן האבולוציוני שלנו, השוונו את תוצאותינו לתוצאות המוצגות שם.  
נציג חלק מהשחקנים והתוצאות שהם השיגו:

### ov3y (ovolve) / 2048-AI - [Matt Overlan](https://github.com/ov3y/2048-AI)

**האלגוריתם ([5],[6])**

* שימוש mini-max search with alpha-beta pruning
* זמן חישוב לצעד של 100 מילי-שניות
* שימוש ביוריסטיקות ההבאות:
  + Monotonicy- מודדת על מימוש הכלל הבא - כל האריחים מסודרים בסדר עולה לכיוון אחד הקירות, כאשר האריח הגדול ביותר ממוקם בפינה.
  + Smoothness- מודדת את ההפרש בין אריחים שכנים ומנסה לצמצם את ההפרש הזה. לוח "חלק" לגמרי הוא לוח שבו כל האריחים באותו מספר.
  + Number of free tiles- לוח שיש לו מעט מדי משבצות פנויות מקבל קנס - הורדה בערכו.

**תוצאות**

* 90% מהריצות הגיעו למספר 2048, כלומר מתוך 1000 ריצות, 900 מהן הגיעו ל-2048.

### Nneonneo / 2048-AI

**האלגוריתם ([7])**

* שימוש ב- expectimax optimization וחיפוש maximization על כל אפשרויות הריצה, על ידי חישוב תוחלת כל האפשרויות שבהן אריח יופיע אקראית ובחירה בכיוון הטוב ביותר.
* הכותב אומר שלדעתו אי אפשר לבצע pruning ל- expextimax optimization פרט אולי לענפים ממש לא סבירים. לכן האלגוריתם עובר brute force על כל האפשריות עם חישוב התוחלת עבור מיקום האריח הנוסף, ובוחר את האפשרות הטובה ביותר. עומק עץ החישוב נע בין 4 ל-8 רמות חישוב.
* שימוש ביוריסטיקות הבאות:
  + כל אריח פנוי הקנה תוספת ציון 20000+
  + לכל שורה ועמודה - אם האריח הגבוה ביותר בשורה או בעמודה צמוד לקיר אז תוספת ציון 20000+.

**תוצאות**

הריצו את המשחק 100 פעמים והתוצאות הן:

* 100% מהפעמים הגיעו ל-2048
* 97% מהפעמים הגיעו ל-4096
* 76% מהפעמים הגיעו ל-8192
* 13% מהפעמים הגיעו ל-16384

\*הכותב הציג את התוצאה הטובה ביותר של הגע ללוח עם האריחים 16384,8192,4096,2048,512,256 והוסיף שהיא לקחה 37 דקות עד שנגמר המשחק.

### Ronenz – Randomly solver (Pure Monte Carlo game search)

**האלגוריתם ([8],[9](**

* שימוש באסטרגיה מאוד פשוטה אך במפתיע, מאוד יעילה.
* עבור כל צעד שרוצים לבחור (למעלה, ימינה, שמאלה, למטה) בוחרים אקראית צעדים עד סוף המשחק, ומחשבים ציון לתוצאה שהתקבלה. חוזרים על התהליך X פעמים. הצעד אותו אנו בוחנים יקבל את הציון הממוצע של התוצאות שהתקבלו.
* הצעד עם ממוצע הניקוד הגבוה ביותר נבחר.

**תוצאות:**

עבור x=100 האלגוריתם מגיע ל:

* 80% מהפעמים לאריח 2048
* 50% מהפעמים לאריח 4096

עבור x=10000 האלגוריתם מגיע ל:

* 100% מהפעמים לאריח 2048
* 70% מהפעמים לאריח 4096
* 1% מהפעמים לאריח 8192

המתכנת ציין כי לאחר ריצות רבות הוא שם לב כי קיים חסם על התוצאות ממש סמוך להופעת האריח 8192 כאשר הניקוד הוא בסביבות 80000. הגדלת x לא משפרת דרסטית את הביצועים.

# ייצוג פתרון (Solver)

לאלגוריתם הוגדרו מספר יוריסטיקות הנותנות ציון למצב הלוח; הציון של כל יוריסטיקה נורמל לערך בין 0 ל-1.

כל הפרטים באוכולוסיה משתמשים באותן היורסטיקות, ומה שמבדיל בין פרט לפרט הוא כמה משקל נותן כל אחד מהם ליוריסטיקות השונות. לכן חשוב היה לנרמל את ציוני היורסיטיקות – כך שלמשקלים הניתנים ע"י הפתרונות תהיה משמעות.

אם כן, כל פתרון מיוצג באמצעות מערך משקלים, כך שכל אינדקס i מייצג משקל הניתן ליוריסטיקה i.

החלטנו שהמשקלים ההתחלתיים שיקבעו לפרט ינועו בין 0 ל1000 לכל יוריסטיקה.

# תיאור האלגוריתם

## האלגוריתם (Standard Genetic Algorithm, Heuristiclub)

1. אתחל אוכלוסיה בגודל p, עם מערך משקלים הנעים בטווח WieghtRange.
2. חשב פונקצית fitness לכל אחד מן הפרטים באוכלוסיה הראשונית
3. בכל דור, ועד לדור numOfGenerations
   1. בצע סלקציה לפי Selection Operator
   2. בצע crossover על כל הפרטים עם הXO Operator
   3. בהסתברות Pm בצע מוטציה עם הMutation Operator
   4. חשב פונקצית fitness לכל אחד מן הפרטים
   5. בחר לדור החדש כמות הורים כמתוארת בפרמטר Elitism
   6. השלם את הדור החדש מתוך הפרטים החדשים בסוף שלב d.

## פרמטרים

כל ההרצות בוצעו עפ"י האלגוריתם שתואר לעיל, אך ההרצות נבדלות ביניהן עפ"י הפרמטרים המתוארים כאן:

### אופרטורים

|  |  |
| --- | --- |
| XO Operator | אופרטור הcrossover |
| Mutation Operator | אופרטור הMutation |
| Selection Operator | אופרטור הSelection לבחירת הורים לדור הבא |
| Fitness Function | פונקצית הfitness |

### ערכים

|  |  |
| --- | --- |
| Population | גודל האוכלוסיה |
| NumOfGeneratinos | מס' הדורות לחישוב |
| (k) Group size for the Tournament Selector | גודל קבוצה לTournament Selector |
| WieghtRange | טווח אתחול משקל במערך המשקלים |
| (Pm) probability for mutation | הסתברות לביצוע מוטציה על פרט |
| Elitism | כמות ההורים שמועברים כפי שהם לדור הבא |

### חישוב הfitness

|  |  |
| --- | --- |
| Number Of Games | כמות המשחקים שהפרט מבצע בחישוב התאמתו  (התוצאה היא חציון התוצאות) |
| Heuristics | מערך היוריסטיקות בהם משתמש הפרט לבחירת צעד |
| LookAhead | מספר הצעדים שהפרט מסתכל קדימה בבחירת צעד |
| Expectation of Random cell | בסמלוץ צעד, האם לחשב תוחלת לוחות לאחר תוספת אריח אקראי וחשב ציון צעד לפיה. |

# היוריסטיקות

|  |  |  |
| --- | --- | --- |
| id | יוריסטיקה | תיאור |
| 0 | NumOfEmptyCells | מספר התאים הריקים בלוח |
| 1 | AddedScore | סך הנקודות שנוספו לניקוד |
| 2 | NumberOfIdenticalTiles | מספר האריחים הזהים בלוח |
| 3 | DistancesAndValues | סכום המרחקים בין אריחים לעומת ערכם |
| 4 | ProximateSameTiles | מספר האריחים השכנים בעלי אותו ערך |
| 5 | Proximate1PowerDiffTiles | מספר האריחים השכנים בעלי הבדל ערך של חזקה אחת |
| 6 | LocationCorner | מיקום אריחים בפינות הלוח |
| 7 | LocationSides | מיקום אריחים בצידי הלוח (צמוד לקירות, לא בפינה) |
| 8 | LocationMiddle | מיקום אריחים באמצע הלוח |
| 9 | TrapedTile | אריח שכלוא בין אריחים גדולים יותר (יוריסטיקה שלילית) |

## NumOfEmptyCells

**תיאור**

יוריסטיקה זו מחשבת את מספר התאים שאין בהם אריחים מבין 16 התאים, ונותנת ציון גבוה ביחס ישר למספר התאים ריקים.

לדוגמה, בלוח המוצג כאן, ינתן הציון הבא (לפני נרמול): 14

**מוטיבציה**

אחד הקשיים שמציב המשחק ככל שמתקדמים הוא שהלוח מתמלא אריחים ומופחת מרחב התמרון (קיימות פחות אפשרויות תזוזה מבין ארבעת הכיוונים), וכן מוגבלת האפשרות "לחלץ" אריחים גבוהים במיקומים לא אסטרטגיים. לכן, בשלבים מסויימים של המשחק, כמות התאים הריקים מתארת תמרון נכון של הלוח כך שאריחים זהים יתמזגו לאריח גבוהה יותר.

## AddedScore

**תיאור**

יוריסטיקה זו מחשבת את הניקוד שהתווסף בעקבות הצעד שבוצע, ונותנת ציון גבוה ביחס ישר למספר הנקודות שהתווספו.

**מוטיבציה**

הניקוד הוא חלק ממדד ההצלחה של המשחק, ולכן יוריסטיקה זו מתבקשת. כמו כן, הניקוד מתאר את ערך האריח הגבוה ביותר אליו הגיע השחקן, וגדל ככל שקיימים יותר אריחים כאלו. לכן תוספת ניקוד גבוה מנבאת את גודל התקדמות השחקן במטרתו להגיע לאריח 2048 ולנצח (ואף לאריחים גבוהים יותר).

## NumberOfIdenticalTiles

**תיאור**

יוריסטיקה זו סוכמת את כמות האריחים הזהים בלוח, ונותנת משקל פרופורציונלי לחזקת ערך האריח ע"י ספירת האריחים עבור כל ערך והכפלה בחזקה המתאימה של 2.

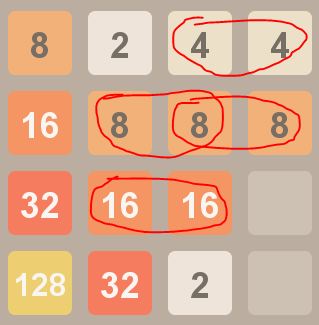
לדוגמה, בלוח המוצג כאן, ינתן הציון הבא (לפני נרמול):

**מוטיבציה**

מאחר והתקדמות במשחק מושגת ע"י חיבור של אריחים בעלי אותו הערך בלבד, ככל שיש יותר אריחים עם אותו ערך על גבי הלוח, יש יותר סיכוי לחברם. כמובן שככל שערך האריחים הללו גבוה יותר, כך גובר הסיכוי להתקדם לעבר התוצאה הרצויה.

## ProximateSameTiles

**תיאור**

****יוריסטיקה זו סופרת את כמות האריחים השכנים (מ-4 הכיוונים) בעלי אותו הערך, ונותנת משקל פרופורציונלי לחזקת ערך האריח ע"י ספירת האריחים עבור כל ערך והכפלה בחזקה המתאימה של 2.

לדוגמה, בלוח המוצג כאן, ינתן הציון הבא (לפני נרמול):

**מוטיבציה**

יוריסטיקה זו מאפשרת הסתכלות צעד אחד קדימה על זוגות אריחים צמודים שניתן לאחד לחזקה גבוהה יותר. כמובן שככל שערך האריחים הללו גבוה יותר, כך גובר הסיכוי להתקדם לעבר התוצאה הרצויה.

## Proximate1PowerDiffTiles

**תיאור**

בדומה ליוריסטיקה הקודמת, יוריסטיקה זו סופרת את כמות האריחים השכנים (מ-4 הכיוונים) בעלי ערך עם הפרש חזקה אחת של 2, ונותנת משקל פרופורציונלי לחזקת ערך האריח ע"י ספירת האריחים עבור כל ערך והכפלה בחזקה המתאימה של 2.

לדוגמה, בלוח המוצג כאן, ינתן הציון הבא (לפני נרמול):

**מוטיבציה**

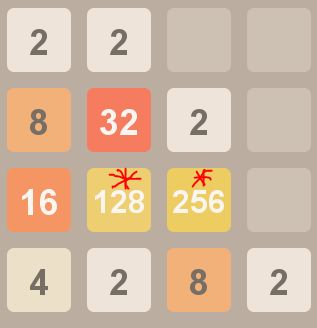
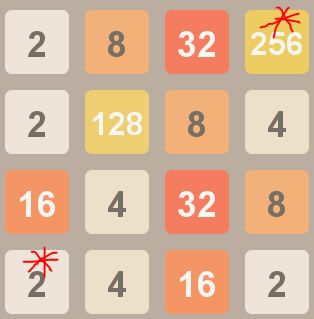
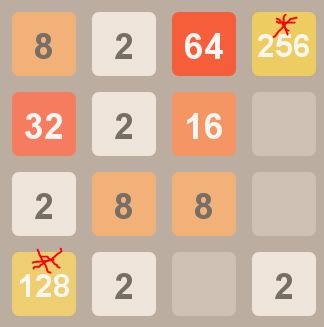
יוריסטיקה זו מאפשרת הסתכלות כמה צעדים קדימה על זוגות אריחים צמודים שבשלבים קרובים יהיה ניתן לאחד לחזקה גבוהה יותר. כמובן שככל שערך האריחים הללו גבוה יותר, כך גובר הסיכוי להתקדם לעבר התוצאה הרצויה.

## DistancesAndValues

**תיאור**

יוריסטיקה זו סוכמת עבור כל שני אריחים על הלוח (שכנים או לא) את המרחק ביניהם + הפרש החזקה בין ערכיהם, ונותנת משקל פרופורציונלי לחזקת ערך האריח הגבוה מביניהם ע"י חישוב עבור כל ערך והכפלה בחזקה המתאימה של 2. היוריסטיקה נותנת ציון **נמוך** לערך גבוה של החישוב. (כלומר הציון המוחזר הוא 1/calculation)

לדוגמה, בלוחות המוצגים כאן, יתווסף הציון הבא עבור האריחים המסומנים (לפני נרמול):



**מוטיבציה**

יוריסטיקה זו מאפשרת הסתכלות כללית על הלוח ולמעשה משלבת במובן רחב כמה מן היוריסטיקות הקיימות. ככל שהמרחק בין אריחים קטן יותר, וכן הפרש החזקה ביניהם, כך גובר הסיכוי לחברם ולהגיע לחזקות גבוהות יותר. כמובן שככל שערך האריחים הללו גבוה יותר, כך גובר הסיכוי להתקדם לעבר התוצאה הרצויה.

## LocationCorner

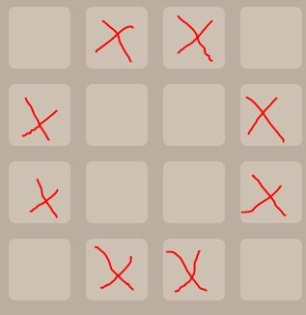
**תיאור**

יוריסטיקה זו מחשבת את האריחים שנמצאים בפינה. היא סוכמת את חזקות האריחים הנמצאים בפינות. כלומר, ככל שערכו של האריח שנמצא בפינה גבוה יותר, אז היוריסטיקה תחזיר ערך גבוה יותר.

**מוטיבציה (יוריסטיקות מבוססות מיקום)**

יוריסטיקות אלו נותנת חשיבות למיקום האריח, מאינטואיציה של שחקנים רבים ששיחקו את המשחק מתברר שהם מגיעים לתוצאות טובות אם האריח הגבוהה ביותר נמצא באחת מפינות הלוח. בכדי לא להגביל את השחקן לשיטת משחק אחת, על פיה מצמידים את האריח לפינה, קיימות יורסטיקות לאזורים שונים של הלוח.

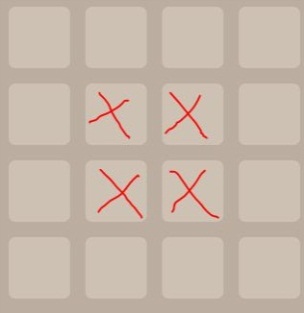
## LocationSides

**תיאור**

יוריסטיקה זו מחשבת את האריחים שנמצאים ליד קיר אך לא בפינות. היא סוכמת את חזקות האריחים שנמצאים ליד קיר אך לא בפינות. כלומר, ככל שערכו של האריח שנמצא ליד קיר ולא בפינות גבוה יותר, אז היוריסטיקה תחזיר ערך גבוה יותר.

**מוטיבציה (יוריסטיקות מבוססות מיקום)** ראה ביוריסטיקה LocationCorner

## LocationMiddle

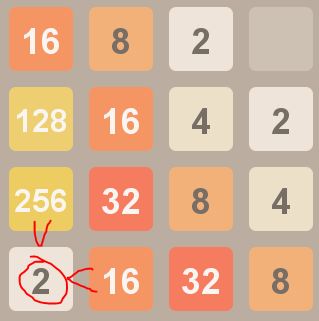
**תיאור**

יוריסטיקה זו מחשבת את האריחים שנמצאים באמצע הלוח. היא סוכמת את חזקות האריחים שנמצאים באמצע הלוח. כלומר, ככל שערכו של האריח שנמצא באמצע הלוח גבוה יותר, אז היוריסטיקה תחזיר ערך גבוה יותר.

**מוטיבציה (יוריסטיקות מבוססות מיקום)** ראה ביוריסטיקה LocationCorner

## TrapedTile

**תיאור**

יוריסטיקה זו בודקת עבור כל אריח האם הוא כלוא, ואם כן קונסת את הלוח על פי חזקה הופכית\* של האריח. אריח נחשב כלוא אם אין לו משבצת פנוייה לידו, וגם אם ערך כל שכניו גדול משלו.

\* חזקה הופכית – הנחנו שהחזקה הגבוהה ביותר שאנו מגיעים אליה היא

4096, ומיפינו כל חזקה לחזקה ההופכית ביחס לחזקות הקיימות. לדוגמה 2 הופכי ל-10 (ה-3 מימין וה-3 משמאל), וכן 5 הופכי ל7.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 4096 | 2048 | 1024 | 512 | 256 | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 0 |
| 12 | 11 | 10 | 9 | 8 | 7 | **6** | 5 | 4 | 3 | 2 | 1 | 0 |

לדוגמה, בלוח המוצג כאן, ינתן הציון הבא (לפני נרמול):

**מוטיבציה**

לאריחים כלואים אין מסביבם אף אריח אליו יכולים לחבור, ובכך הם תופסים שטח גדול מלוח המשחק בכך שהם מונעים איחוד של אריחים שכן יכלו להתאחד.

# סקירת קוד ה- Evaluator ואופרטורים

לשם סמלוץ משחק, כתבנו קוד למשחק 2048 – הסתמכנו על קוד open source הכתוב בjava והמרנו אותו לקוד C# ע"מ שיעבוד עם תוכנה בשם Heuristiclab[10] המקומפלת בשפה זו. כמו כן, הורדנו את גרפיקת המשחק ע"מ לזרז את חישוב המשחק, וכתוצאה מכך את חישוב הfitness של פרט (Quality במושגי Heuristiclab).

לשם הרצת האלגוריתם האבולוציוני, השתמשנו בתוכנה בשם Heuristiclab, ובחרנו בפורמט Genetic Algorithm, שאופן פעולתו מתוארת בחלק הדו"ח "האלגוריתם".

יצרנו קובץ C# בשם Evaluator אותו לאחר מכן קישרנו לספריות Heuristiclab וקימפלנו את הפרויקט עם קוד זה. מטרת הEvaluator לקבל פרט ולשערך את ערך הfitness שלו.

אופן חישוב Quality של פרט:

1. סמלץ **משחק** number Of Games פעמים:
   1. עד לנצחון או הפסד בצע **צעד**:
      1. בכל חישוב צעד משחק **סמלץ את כל אחד מארבעת הצעדים** – ימינה, שמאלה, למטה, למעלה, וחזור למצב הלוח הקודם.
         1. לכל צעד חשב את הציון על פי הנוסחה

אם Expectation of Random Cell, חישוב הציון של הצעד יבוצע על פי תוחלת ציוני לוח עבור כל **אפשרויות הנחת האריח האקראי** בהסתברות הזהה לזו הקיימת במשחק, ויקח אותה כציון הצעד.

* + 1. בחר את הצעד עם **הציון הגבוה ביותר** ובצע אותו.
    2. המשך לצעד הבא.
  1. שמור את ציון המשחק.

1. החזר כQuality (Fitness)- את הערך **החציוני/עליון** של ציוני המשחק שהתקבלו.

אופרטורים:

Average Crossover- בוחר מספר הורים ומחשב עבור כל תא בוקטור של הצאצא את ממוצע תאי ההורים.

Discrete Crossover- בוחר עבור על תא בוקטור הצאצא לקחת את הערך מההורה הראשון או השני.

MultiRealVectorManipulator- בוחר באופן רנדומלי אחד מהאופרטורים המוגדרים אצלו ומפעיל אותם על הפרט שנבחר למוטציה. (BreederGeneticAlgorthm, SelfAdaptiveNormalAllPositions, PolynomialAllPosition, PolynomialOnePosition, UniformOnePosition). מכיוון שכל ה

TurnamentSelection - Selector שבוחר איזה פרטים יעברו לדור הבא על ידי בחירה של קבוצת פרטים מגודל k מתוך האוכלוסיה הנוכחית באופן רנדומלי, ובחירת הטוב ביותר מביניהם עד קבלת דור חדש.

Elitisem – קבוצה של פרטים שעוברים לדור הבא ללא מעבר בתהליך ה- selection.

# תוצאות

## אופן בדיקות התוצאות

מכיוון ששינינו את פונקציית ה- evaluation בכל פעם כדי לבדוק את שיפור הפרטים בין הריצות השונות, ובין תחילת הריצה לסיום הריצה, בדקנו את התוצאות כך:

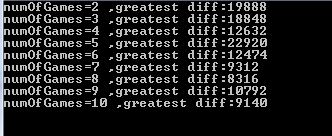
* בדיקת ערכו של פרט בוצעה ע"י סימלוץ 1000 משחקים על פי הEvaluator על פיו פותח הפרט, ושמירת הערכים הבאים –
  + Quality (חציון הscore של 1000 המשחקים)
  + כמות הפעמים שהפרט הגיע ל-1024
  + כמות הפעמים שהפרט הגיע ל-2048
  + כמות הפעמים שהפרט הגיע ל-4096
* בכל ריצה, נבדק ערכם של הפרטים הבאים –
  + הפרט הטוב ביותר שהאבולוציה השיגה בדור 0.
  + הפרט הטוב ביותר בדור האחרון של האבולוציה.

כך שמרנו על יחס מסויים בין הריצות השונות, שאיפשר לנו להשוות ביניהן, על פי הפרמטרים שצויינו לעיל.

\*הערכים של כל פרט באוכלוסיה (תאי הוקטור) הם ממשיים. לשם הצגת הדוח עיגלנו את הערכים למספרים שלמים.

\* ביצענו עשרות ריצות ונסיונות, וכאן הבאנו ריצות מובחרות ומעניינות לדיון.

\*מדעו נבחר החציון של 1000 משחקים כ- Quality של פרט – הרצנו חישוב עבור פרט טוב כלשהו, שבו בדקנו מהיא הסטייה בחישוב עבור כמות משחקים מסויימת. הרצנו 100 פעמים כל פעם עם numOfGames שונה ושמרנו את ה- Quality שהתקבל מחציון ה- numOfGames. לקחנו את ערך ה- Quality המקסימלי והמינימלי שקיבלנו, והדפסנו את ההפרש של שניהם. אפשר לראות מהתוצאות כי כאשר לוקחים 1000 משחקים לפרט, ההבדל בין ה- Qualities הינו בסדר גודל של כמה מאות נקודות.









## סקירת התוצאות

### ריצה ראשונה – נסיונות ראשונים

|  |  |
| --- | --- |
| **אופרטורים** | |
| XO Operator | Average Crossover |
| Mutation Operator | MultiRealVectorManipulator |
| Selection Operator | Tournament Selector |
| Fitness Function | ניקוד הלוח הגבוה מתוך סדרת משחקים |

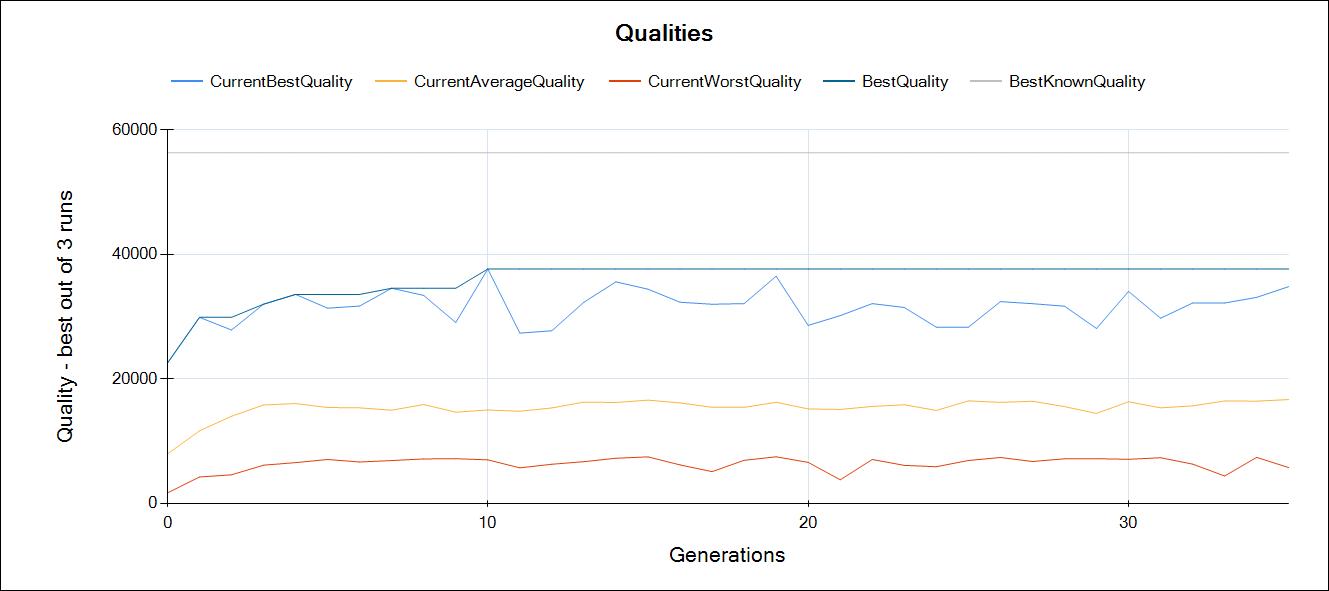
|  |  |
| --- | --- |
| **ערכים** | |
| Population | 100 |
| NumOfGeneratinos | 35 |
| (k) Group size for the Tournament Selector | 5 |
| WieghtRange | 0-1000 |
| (Pm) probability for mutation | 10% |
| Elitism | 3 |

|  |  |
| --- | --- |
| **Fitnessחישוב ה-** | |
| Number Of Games per player | 3 |
| Heuristics | 0-9 |
| LookAhead | 1 |
| Expectation of Random cell | no |

**מטרת הריצה**

קבלת מושג על סדרי גודל של שיפור כתוצאה מהאבולוציה ונקודת ייחוס לניסויים אחרים. כמו כן בחרנו ערך של Quality כמשחק הטוב ביותר משום שרצינו לקבל את המשחקים שהצליחו להגיע הכי רחוק. (דבר שהתברר כטעות – ראה ריצה רביעית)

|  |  |  |
| --- | --- | --- |
| פרמטרים | פרט טוב ביותר בדור 0 | פרט טוב ביותר בדור 35 |
| הפרט עצמו | לא נשמר ביוריסטיקלאב | |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 460 | 221 | 623 | 488 | 423 | 760 | 529 | 340 | 605 | 608 | | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
| כמות 1024 | 434 | 492 (49%) |
| כמות 2048 | 34 | 45 (4.5%) |
| כמות 4096 | 0 | 0 (0%) |
| Quality | 8884 | 10270 |



**ניתוח תוצאות**

ניתן לראות שיפור קל בפרט הטוב ביותר בריצה האחרונה. הפסקנו את הריצה משום שרצינו לקבל תוצאות טובות יותר על ידי הרצה של כמות גדולה יותר של פרטים.

### **ריצה שנייה – הגדלת האוכלוסיה וכמות המשחקים לפרט,**

### **ולקיחת חציון**

|  |  |
| --- | --- |
| **אופרטורים** | |
| XO Operator | Average Crossover |
| Mutation Operator | MultiRealVectorManipulator |
| Selection Operator | Tournament Selector |
| Fitness Function | ניקוד הלוח **החציוני** מתוך סדרת משחקים |

|  |  |
| --- | --- |
| **ערכים** | |
| **Population** | **200** |
| NumOfGeneratinos | 172 |
| (k) Group size for the Tournament Selector | 5 |
| WieghtRange | 0-1000 |
| (Pm) probability for mutation | 10% |
| Elitism | 3 |

|  |  |
| --- | --- |
| **fitnessחישוב ה** | |
| **Number Of Games per player** | **20** |
| Heuristics | 0-9 |
| LookAhead | 1 |
| Expectation of Random cell | no |

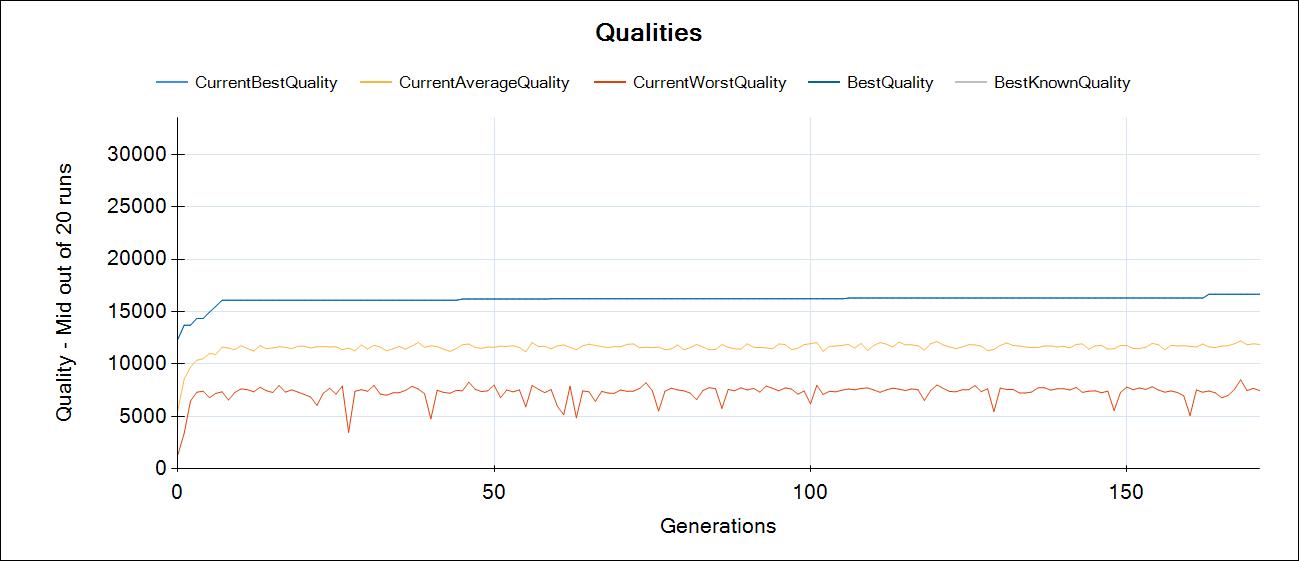
**מטרת הריצה**

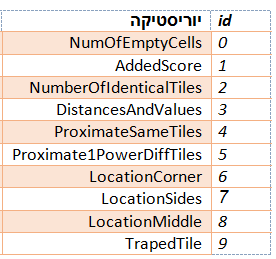
מכיוון שהשיפור לא היה משמעותי כפי שציפינו בריצות עד כה:

* הגדלנו את **כמות הריצות** לפרט כדי לקבל שיערוך מדוייק יותר של הפרטים באוכלוסיה וכך להבדיל בין פרטים טובים יותר לטובים פחות.
* הגדלנו את **גודל האוכלוסייה** מכיוון שהבנו שמרחב החיפוש הוא גדול מאוד (100010) וכדי לקבל כיסוי רחב יותר נזדקק ליותר פרטים באוכלוסיה.
* החלטנו לבחור **ערך חציוני** של המשחקים של פרט כדי לקבל Quality יותר מדוייק עבור אותו פרט ולא להסתמך על תוצאה טובה יחידה.

|  |  |  |
| --- | --- | --- |
| פרמטרים | פרט טוב ביותר בדור 0 | פרט טוב ביותר בדור 172 |
| הפרט עצמו | לא נשמר ביוריסטיקלאב | |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 482 | 208 | 590 | 445 | 314 | 598 | 612 | 703 | 640 | 474 | | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
| כמות 1024 | 457 | 509 (50%) |
| כמות 2048 | 33 | 52 (5%) |
| כמות 4096 | 0 | 0 (0%) |
| Quality | 9050 | 10732 |

\*ההבדל בין ה- Quality שרואים בגרפים בריצה זו ובזו שלפניה נובע מכך שבריצה הקודמת לקחנו את ציון המשחק הגבוה ביותר כ- Quality בעוד בריצה הזו לקחנו את הערך החציוני, ושבאופן ברור נמוך יותר. לכן כאן, ההשוואה בין הריצות נעשית על פי הטבלה ולא על פי הגרף.

**ניתוח תוצאות**

ניתן לראות שיפור קל מאוד בהשוואה לריצה הקודמת. בניגוד לציפיות לפיהן הריצה תהיה משמעותית טובה יותר, שכן קיבלה יותר דורות לרוץ, יותר פרטים באוכלוסיה ושיערוך פרט היה מדויק יותר (יותר משחקים לפרט).

### **ריצה שלישית - שינוי ה- Crossover**

|  |  |
| --- | --- |
| **אופרטורים** | |
| **XO Operator** | **DiscreteCrossover** |
| Mutation Operator | MultiRealVectorManipulator |
| Selection Operator | Tournament Selector |
| Fitness Function | ניקוד הלוח החציונימתוך סדרת משחקים |

|  |  |
| --- | --- |
| **ערכים** | |
| Population | 200 |
| NumOfGeneratinos | 70 |
| (k) Group size for the Tournament Selector | 5 |
| WieghtRange | 0-1000 |
| (Pm) probability for mutation | 10% |
| Elitism | 3 |

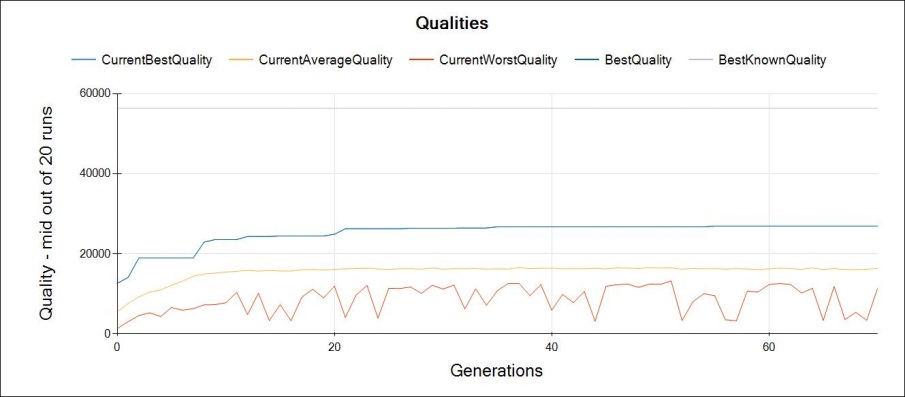
|  |  |
| --- | --- |
| **fitnessחישוב ה** | |
| Number Of Games per player | 20 |
| Heuristics | 0-9 |
| LookAhead | 1 |
| Expectation of Random cell | no |

**מטרת הריצה**

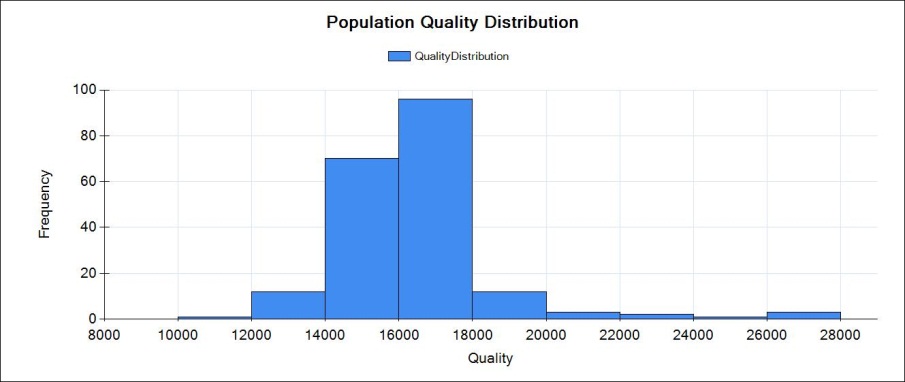
מכיוון שבריצה הקודמת ראינו כי שינוי גדול באיכות הריצה (גודל אוכלוסיה,מספר דורות,כמות ריצות לפרט) הקנה רק שיפור מזערי, חשדנו שמשהו מכנס אותנו למקסימום מקומי. לכן החלטנו לשנות אופרטור:

* **שינוי ה- Crossover** ל- DiscreteCrossover שיגרום לגיוון רב יותר באוכלוסיה.

|  |  |  |
| --- | --- | --- |
| פרמטרים | פרט טוב ביותר בדור 0 | פרט טוב ביותר בדור 70 |
| הפרט עצמו | לא נשמר ביוריסטיקלאב | |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 36 | 63 | 535 | 387 | 166 | 989 | 810 | 316 | 435 | 715 | | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
| כמות 1024 | 394 | 752 (75%) |
| כמות 2048 | 30 | 211 (21%) |
| כמות 4096 | 0 | 5 (0.5%) |
| Quality | 9050 | 15456 |



**ניתוח תוצאות**

ניתן לראות ששינוי ה- crossover והכנסת הגורם האקראי שב-DiscreteCrossover שיפר את הפרט, הרבה יותר מאשר בריצות האחרות. כלומר הצלחנו לצאת מהמקסימום המקומי אליו נקלענו.

### 

### **ריצה רביעית - הורדת כמות המשחקים לפרט**

|  |  |
| --- | --- |
| **אופרטורים** | |
| XO Operator | DiscreteCrossover |
| Mutation Operator | MultiRealVectorManipulator |
| Selection Operator | Tournament Selector |
| Fitness Function | ניקוד הלוח החציונימתוך סדרת משחקים |

|  |  |
| --- | --- |
| **ערכים** | |
| Population | 200 |
| NumOfGeneratinos | 146 |
| (k) Group size for the Tournament Selector | 5 |
| WieghtRange | 0-1000 |
| (Pm) probability for mutation | 10% |
| Elitism | 3 |

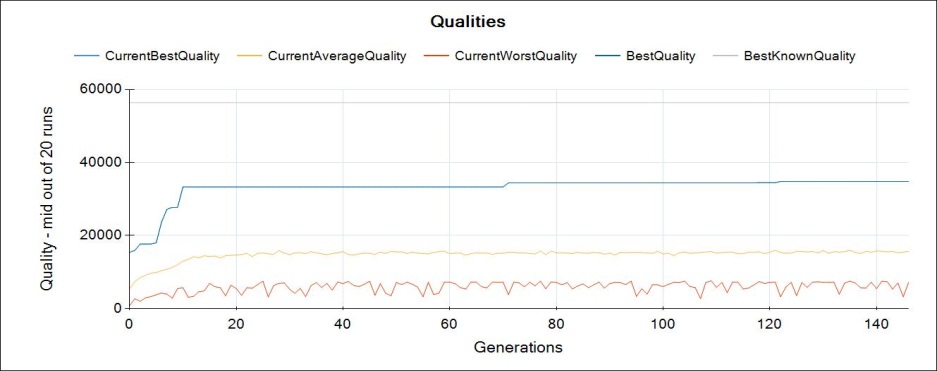
|  |  |
| --- | --- |
| **fitnessחישוב ה** | |
| **Number Of Games per player** | **5** |
| Heuristics | 0-9 |
| LookAhead | 1 |
| Expectation of Random cell | no |

**מטרת הריצה**

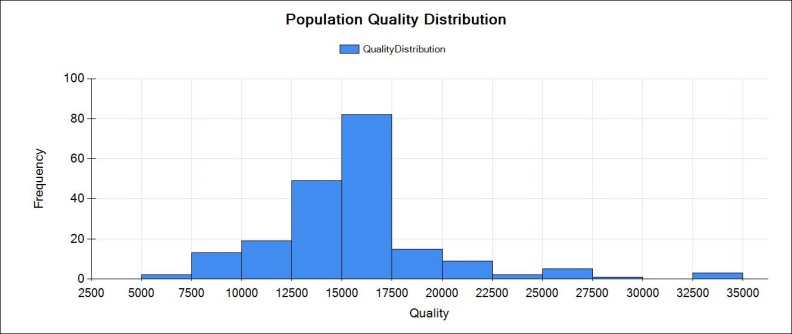
הריצה הקודמת לקחה כ-17 שעות. וכמו כן רצינו לשפר את אלגוריתם ה- evaluation (ע"י חישוב תוחלת אפשרויות הופעת אריחים אקראיים בחישוב כל צעד) דבר אשר יגדיל את זמן הריצה בצורה משמעותית. לכן כדי לנסות להוריד את זמן הריצה בוצע השינוי הבא:

* **הקטנה של כמות המשחקים** עבור כל פרט ל- 5 משחקים במקום 20. (אמור להקטין את זמן הריצה פי רבע)

|  |  |  |
| --- | --- | --- |
| פרמטרים | פרט טוב ביותר בדור 0 | פרט טוב ביותר בדור 146 |
| הפרט עצמו | לא נשמר ביוריסטיקלאב | |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 51 | 118 | 767 | 486 | 194 | 978 | 864 | 226 | 583 | 615 | | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
| כמות 1024 | 234 | 711 (71%) |
| כמות 2048 | 4 | 183 (18%) |
| כמות 4096 | 0 | 2 (0.2%) |
| Quality | 7121 | 14890 |



**ניתוח תוצאות**

זמן הריצה של האבולוציה היה 07:50 שעות והוא הגיע לדור 146 בהשוואה לריצה הקודמת שרצה כ-17 שעות והגיעה לדור 70 בלבד. בהשוואה לריצה שבה החלפנו את ה- Crossover:

* כמות משחקים שהגיעו ל- 1024 קטנה ב- 41.
* כמות משחקים שהגיעו ל- 2048 קטנה ב- 28.
* כמות משחקים שהגיעו ל- 4096 קטנה ב- 3.
* Quality קטנה ב- 566.

לכן קיבלנו שחקן פחות טוב בריצה זאת. ולכן הורדה של כמות המשחקים לפרט פוגעת בתוצאה שנקבל. בנוסף אם מסתכלים על התפלגות ה- Quality בדור האחרון:

* בריצה זו ה- WorstQuality היה 7124. בריצה הקודמת ה- WorstQuality היה 11382 אפשר לראות את ההבדל גם בגרף ה- Quality.
* בריצה זו ה- avrageQuality היה 15567. בריצה הקודמת ה- averageQuality היה 16355 קצת יותר גבוה לאורך כל הריצות.

כלומר האוכלוסיה כולה בריצה הקודמת (זאת שהחלפנו crossover) הגיעה לנקודה יותר טובה מאשר האוכלוסיה בריצה הזאת.

### **ריצה חמישית - האם ערך עליון יותר טוב מאשר חציון**

|  |  |
| --- | --- |
| **אופרטורים** | |
| XO Operator | DiscreteCrossover |
| Mutation Operator | MultiRealVectorManipulator |
| Selection Operator | Tournament Selector |
| Fitness Function | ניקוד הלוח **הגבוה** מתוך סדרת משחקים |

|  |  |
| --- | --- |
| **ערכים** | |
| Population | 200 |
| NumOfGeneratinos | 53 |
| (k) Group size for the Tournament Selector | 5 |
| WieghtRange | 0-1000 |
| (Pm) probability for mutation | 10% |
| Elitism | 3 |

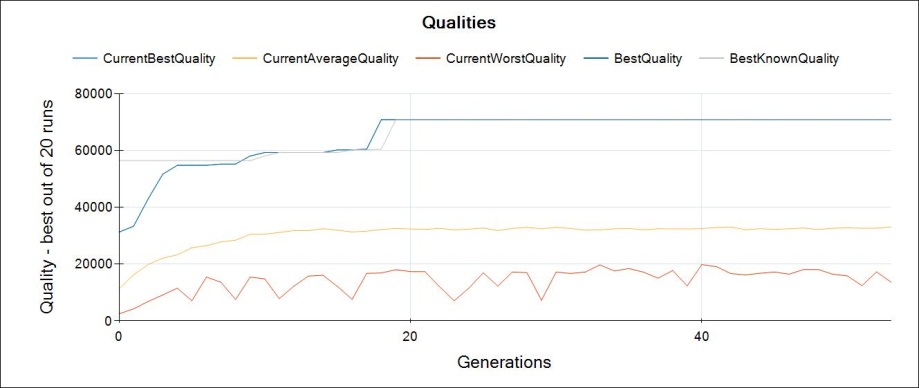
|  |  |
| --- | --- |
| **fitnessחישוב ה** | |
| **Number Of Games per player** | **20** |
| Heuristics | 0-9 |
| LookAhead | 1 |
| Expectation of Random cell | no |

**מטרת הריצה**

לבדוק האם חישוב Quality כערך הגבוה עבור כל המשחקים של הפרט יתן תוצאות יותר טובות מאשר לקיחת חציון.

* **שינוי כמות המשחקים לפרט** חזרה ל-20 כדי לקבל תוצאות טובות כמו לאחר שינוי ה- crossover, ולשם השוואה אליו על פני השוואה לריצה עם חמשת המשחקים לפרט.
* **שינוי חישוב ה- Quality** של פרט על ידי לקיחת ה- Score הכי גבוה מתוך 20 המשחקים של כל פרט.

|  |  |  |
| --- | --- | --- |
| פרמטרים | פרט טוב ביותר בדור 0 | פרט טוב ביותר בדור 53 |
| הפרט עצמו | לא נשמר ביוריסטיקלאב | |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 73 | 150 | 948 | 577 | 381 | 967 | 998 | 669 | 876 | 158 | | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
| כמות 1024 | 469 | 700 (70%) |
| כמות 2048 | 36 | 171 (17%) |
| כמות 4096 | 0 | 4 (0.4%) |
| Quality | 9434 | 14540 |

\*ההבדל בין ה- Quality שרואים בגרפים נובע מכך שלקחנו בריצה הזאת ערך גבוה ביותר כ- Quality, ובריצה שבה החלפנו Crossover לחקנו ערך חציוני. לכן כאן, ההשוואה בין הריצות נעשתה על פי הטבלה.

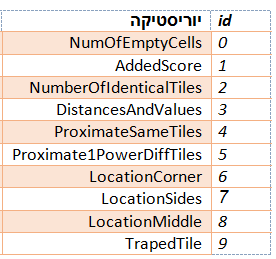
**ניתוח תוצאות**

בהשוואה לריצה שבה שינינו את ה- Crossover:

* כמות משחקים שהגיעו ל- 1024 קטנה ב- 52
* כמות משחקים שהגיעו ל- 2048 קטנה ב- 40.
* כמות משחקים שהגיעו ל- 4096 קטנה ב- 1.
* Quality קטנה ב- 916.

קיבלנו שחקן פחות טוב מאשר בריצה בה שינינו את ה-crossover. כלומר לקיחת ערך עליון עבור סדרת משחקים של פרט מסויים פחות טובה מאשר לקיחת חציון.

### **ריצה שישית - ביצוע צעד על פי תוחלת מיקום אריח אקראי**



|  |  |
| --- | --- |
| **אופרטורים** | |
| XO Operator | DiscreteCrossover |
| Mutation Operator | MultiRealVectorManipulator |
| Selection Operator | Tournament Selector |
| Fitness Function | ניקוד הלוח **החציוני** מתוך סדרת משחקים |

|  |  |
| --- | --- |
| **ערכים** | |
| Population | 200 |
| **NumOfGeneratinos** | **100** |
| (k) Group size for the Tournament Selector | 5 |
| WieghtRange | 0-1000 |
| (Pm) probability for mutation | 10% |
| Elitism | 3 |

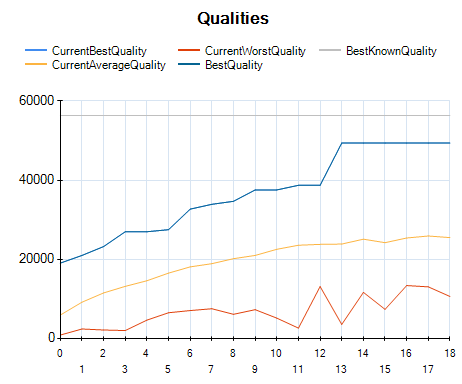
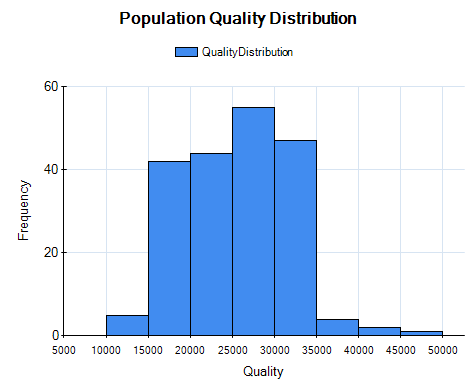
|  |  |
| --- | --- |
| **fitnessחישוב ה** | |
| **Number Of Games per player** | **10** |
| Heuristics | 0-9 |
| LookAhead | 1 |
| **Expectation of Random cell** | **Yes** |

**מטרת הריצה**

לראות כיצד חישוב תוחלת הופעת אריח אקראי על הלוח לאחר ביצוע צעד תורמת לפרט לפתור בצורה טובה יותר את המשחק.

* **שינוי כמות המשחקים לפרט** חזרה ל-10 למרות שכבר ראינו ש-20 ריצות נותנות quality מדוייק יותר ומכאן תוצאות טובות יותר. נאלצנו בכל זאת להוריד את כמות המשחקים ל-10 על מנת להוריד את זמן החישוב שהתארך פי כמה והפך את האבולוציה ללא יעילה מבחינת זמן ריצה
* **שינוי חישוב ה- Quality** של פרט על ידי לקיחת ה- Score החציוני מתוך 10 המשחקים של כל פרט. בעקבות מסקנות קודמות, לפיהן הערך החציוני עדיף על פני הערך הגבוה ביותר.
* **חישוב תוחלת הופעת אריח אקראי לאחר ביצוע צעד, ועל פי חישוב זה מתן ציון לצעד טרם בחירתו.** חישוב זה נותן לפרט ראיה רחבה על השפעת הצעד שלו על מצב הלוח, דבר שלא היה קיים בעבר. לפני כן, "הניח" הפרט שהאריח מוקם במקום אקראי כלשהו, ולפי כך ביצע צעד, שאחריו מוקם מחדש האריח (אולי באותו מקום, אך בהסתברות גבוהה בהרבה יותר – במיקום אחר). ציפינו ששינוי זה יקפיץ את החישוב האבולוציוני כמה צעדים קדימה.

|  |  |  |
| --- | --- | --- |
| פרמטרים | פרט טוב ביותר בדור 0 | פרט טוב ביותר בדור 18 |
| הפרט עצמו | לא נשמר ביוריסטיקלאב | |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 25 | 219 | 858 | 457 | 86 | 995 | 571 | 859 | 620 | 791 | | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
| כמות 1024 | 404 | 781 (78%) |
| כמות 2048 | 18 | 236 (24%) |
| כמות 4096 | 0 | 6 (0.6%) |
| Quality | 8534 | 15958 |

\*לא ניתן להשוות את הגרף לגרפים אחרים שכן הquality נמדד רק על פי 10 משחקים, ולכן הוא "מקל" על הציון לעומת 20 משחקים ו"מקשה" לעומת 3-5 משחקים. 

**ניתוח תוצאות**

התוצאות מאד דומות לריצה בה שינינו את הcrossover – אולם שם הגענו לדור 70 וכאן רק לדור 18.

הריצה הזו לקחה זמן רב מאד – יומיים ו-3 שעות, וכעבור יום וחצי נראה היה שהאבולוציה החלה להתכנס למקסימום מקומי (ניתן לראות בגרף שלעיל). בניגוד לציפיותינו - לעומת ריצות אחרות, ריצה זו לא התחילה עם פרטים מוצלחים יחסית בזכות שיפור ידע השחקן על השלכות הצעד שהוא בוחר לבצע. דבר שהבהיר לנו את חשיבותו של יחס נכון בין המשקלים.

מצד אחד, איכות הפרט לא השתנתה רק במעט ביחס לריצה בה שינינו את הcrossover, ומצד שני הגענו לפרט איכותי יותר, תוך שיפור עקבי של האבולוציה כבר מתחילת ההרצה. דבר המעיד על שיפור איכות המשחק ועל פוטנציאל ריצות המתבססות על תוחלת, אולם הריצה לוקחת זמן רב מדי, וכן התכנסנו למקסימום מקומי.

# ממצאים מעניינים

## יחס משקלי היוריסטיקות

מיינו את תוצאות הריצות מהגרועה ביותר לטובה ביותר – כלומר בסדר הבא מימין לשמאל6,3,4,5,2,1 , והצגנו את הפרט הטוב ביותר של כל ריצה בדור האחרון באמצעות עמודות משקלים כמתואר בגרף הבא:

### Trapped Tile

ניתן לראות שהיוריסטיקה Trapped Tile מקבלת משקל נמוך מאד. יוריסטיקה זו הינה יוריסטקיה שלילית (כלומר קונסת את הquality). אנו משערים ששילובה עם היורסיטיקות האחרות היה בעייתי, והוא פגם בביצועי השחקן.

### Proximate Heuristics (and Distances&Values)

הופתענו לגלות שהיוריסטיקה Proximate 1 diff זו קיבלה משקל נמוך כמעט בכל ההרצות, והלכה וקטנה ככל שהתוצאות השתפרו.

לעומתה, אחותה – Proximate Same קיבלה תמיד משקל משמעותי. אנו מניחים כי Proximate 1 Diff גרמה לשחקן להעדיף מהלכים פחות טובים, בהם מצמידים שני אריחים בהפרש חזקה אחת לעומת הצמדת אריחים זהים. ע"מ לחזק השערה זו, התבוננו גם במשקל היוריסטיקה Distances & Values, אשר "מחברת" את הגיון שתי היוריסטיקות Proximate, אך לא מעדיפה 2 אריחים צמודים בהפרש חזקה אחת על פני זהים.

בנוסף, הבחנו בכך שבריצה ה5 (מוצגת בגרף כריצה 3 –לקיחת כQuality את כערך הגבוה ביותר במקום חציון) משקל היוריסטיקה Proximate 1 diff דווקא גדול משמעותית לעומת ריצות אחרות, ואנחנו משערים שמשמעות הדבר היא שהצמדת שני אריחים עם הפרש חזקה אחת, לעיתים מובילה את השחקן לאריחים גבוהים יותר, אולם על פי ממוצע המשחקים היא אינה טובה יותר.

### Location Heuristics

שמנו לב ש-3 יוריסטיקות המיקום קיבלו ערכים שונים מאד, כאשר ההעדפה היא לפי הסדר:

1. Sides
2. Corner
3. Middle

תוצאות אלו מתיישבות היטב עם ההגיון של שחקני 2048 (אנושיים) רבים, שעל פיו צריך לשאוף להצמיד את האריח בעל הערך הגבוה ביותר לקיר. על אף שהיינו מודעים לסגנון משחק זה, החלטנו לתת לאבולוציה להחליט בשבילנו מהו מיקום מוצלח, ואכן היא בחרה בSides (וcorner).

### Empty Cells ו- Added Score

שתי יוריסטיקות אלה קיבלו משקלים משמעותיים, באופן לא מפתיע. הן היוריסטיקות היחידות שמכוונות את השחקן לאחד אריחים, מכיוון שהScore גדל רק בפעולת איחוד, ומאחר ובכל צעד נוסף אריח אקראי, רק איחוד תורם לשימור כמות התאים הריקים. הדבר מסתדר עם אינטואיציית השחקנים לנקות את הלוח כמה שאפשר.

## ממוצע המשקלים ב- Crossover אינו מוצלח

האופרטור Average Crossover לוקח מספר הורים ומחשב עבור כל תא בוקטור את הממוצע של אותו התא אצל ההורים ומעביר אותו לצאצא. פעולה זו גורמת לכך שהאוכלוסיה תהיה לא מגוונת, ובכך תתכנס למקסימום מקומי ועצירת האבולוציה. אפשר לראות שיפור גדול באיכות הפרט הטוב ביותר בין הריצות השניה והשלישית שבהן הדבר היחידי ששונה הוא ה- crossover. כאשר ה- Quality בריצה השלישית הוא 15456 לעומת ה- Quality בריצה השנייה שעמד על 10732.

|  |  |  |
| --- | --- | --- |
|  | **ריצה שניה** | **ריצה שלישית** |
| **כמות 1024** | 509 (50%) | 752 (75%) |
| **כמות 2048** | 52 (5%) | 211 (21%) |
| **כמות 4096** | 0 (0%) | 5 (0.5%) |
| **Quality** | 10732 | 15456 |

## חציון נכון יותר מערך גבוה

במהלך הריצות השונות, ניסינו לבדוק האם עדיף לקחת את הערך העליון של מספר משחקים של פרט (המשחק הטוב ביותר) או האם כדאי לקחת את החציון. מתוצאות הריצות השלישית והחמישית (ערך חציוני וערך גבוה בהתאמה), ראינו שערך חציוני מוביל אותנו לפרט מוצלח יותר. לכן אנו משערים כי חציון מדייק את ה- Quality ומשקף נכונה את איכות המשחק של הפרט.

|  |  |  |
| --- | --- | --- |
|  | **ריצה שלישית** | **ריצה חמשית** |
| **כמות 1024** | 752 (75%) | 700 (70%) |
| **כמות 2048** | 211 (21%) | 171 (17%) |
| **כמות 4096** | 5 (0.5%) | 4 (0.4%) |
| **Quality** | 15456 | 14540 |

## כמות משחקים גדולה לפרט מדייקת את שערוך הפרט

השוני בין הריצות השלישית והרביעית הוא כמות המשחקים בחישוב ה- Quality. בריצה השלישית כל שחקן שיחק 20 משחקים ונבחר החציון ובריצה הרביעית כל שחקן שיחק 5 משחקים ונבחר החציון. אפשר לראות מתוצאות הריצות כי מהריצה השלישית, עם פי 4 משחקים, קיבלנו פרט טוב יותר מאשר בריצה הרביעית. דבר אשר מחזק את ההשערה שלנו שכמות גדולה יותר של משחקים מובילה לערך Quality מדוייק יותר של הפרט. ניתן לקבל חיזוק נוסף להשערה זו מהבדיקה שעשינו על ידי הרצת כמות משחקים שונה לכל פרט ובדיקת הפרשי ה- Quality שהתקבלו (כפי שמוסבר בחלק - אופן בדיקת התוצאות).

כמו כן יש לציין שהגדלת כמות המשחקים לפרט מגדילה משמעותית את זמן הריצה של דור, לכן כאשר הוספנו את לוגיקת Expected Tile נאלצנו להתפשר ולהוריד את כמות המשחקים על מנת להגיע לתוצאות במסגרת הזמן הנתון לפרוייקט.

|  |  |  |
| --- | --- | --- |
|  | **ריצה שלישית** | **ריצה רביעית** |
| **כמות 1024** | 752 (75%) | 711 (71%) |
| **כמות 2048** | 211 (21%) | 183 (18%) |
| **כמות 4096** | 5 (0.5%) | 2 (0.2%) |
| **Quality** | 15456 | 14890 |

## תוחלת אריח רנדומלי

עד לריצה השישית (שבה הוספנו חישוב של תוחלת אריח אקראי), שחקן הניח שלא מתווסף אריח חדש, וחישב ציון לצעד על פי לוח ללא אריח חדש. רק בביצוע הצעד בפועל, התווסף אריח אקראי לפי חוקי המשחק. כלומר, החסרנו מידע מהשחקן, מידע שיכול להכריע בהחלטה על ביצוע צעד. בריצה השישית, הוספנו את יכולת החישוב הזו לשחקן, על ידי כך שמחושבת תוחלת הלוחות לאחר ביצוע הצעד והופעת אריח בכל המיקומים האפשריים. ברור היה לנו ששינוי זה הוא בעייתי מכיוון שהוא מכפיל את זמן חישוב צעד בכמות התאים הריקים (כפול 2 עבור כל אריח), לכן הורדנו את כמות המשחקים ל-10 כדי להקל על חישוב האבולוציה.

לאורך הדורות הראשונים ראינו נסיקה עקבית בערכי ה-Quality של הפרטים, באופן קצת יותר עקבי מאשר בגרפים של הריצות הקודמות. בהן, כפי שניתן לראות, הייתה קפיצה קטנה ולאחריה התכנסות כאן ראינו פוטנציאל באבולוציה המשתפרת בכל דור בצורה ניכרת ועקבית גם עבור פרטים עם יחס משקלים מוצלח פחות, אולם בדור 13 הפסיקה הנסיקה והתכנסנו לערך שקיבלנו עד לדור 18, ואנו מניחים שהאבולוציה הייתה ממשיכה בהתכנסות זו לולא עצרנו.

כל דור לקח כשעתיים חישוב, ואף בדורות המאוחרים, שבהם השחקנים היו טובים יותר, המשחקים נמשכו זמן רב יותר. לכן נאלצנו לעצור **לאחר 3 ימי ריצה.**

|  |  |  |
| --- | --- | --- |
|  | **ריצה שלישית** | **ריצה שישית** |
| **כמות 1024** | 752 (75%) | 781 (78%) |
| **כמות 2048** | 211 (21%) | 236 (24%) |
| **כמות 4096** | 5 (0.5%) | 6 (0.6%) |
| **Quality** | 15456 | 15958 |

## הצעות לשיפור האלגוריתם

לקראת הגשת הפרויקט עלו לנו רעיונות רבים לשיפור האלגוריתם, אותם לא הספקנו ליישם ולבדוק. כאן נמנה כמה מהם, ואת המוטיבציה מאחוריהם.

מאחר וכל ההרצות שלנו התכנסו למקסימום מקומי בשלב יחסית מוקדם (סביבות דור 20-40), הגענו למסקנה שיתכן והאלגוריתם שלנו משתמש יותר בניצול (Exploitation) ע"י ביצוע Elitism, ו- Crossover לעומת האיתור (**Exploration**) המבוצע ע"י הMutation. על אף שלמדנו שמוטציה צריכה להתבצע בהסתברות נמוכה, חשבנו שהגדלת אחוז זה יכולה להוציא את האלגוריתם שלנו מהמקסימום המקומי אליו הוא נקלע – ואולי אף שינוי הMutator לבצע מוטציות משמעותיות יותר יעזור לחיפוש נרחב יותר במרחב החיפוש. כמו כן, הורדת ערך הelitism עשוי גם הוא לתרום לגיוון האוכולוסיה מדור לדור.

על אף שתוספת **חישוב תוחלת האריח האקראי** לא שיפרה באופן ניכר את התוצאות, אנו מאמינים ששיפור זה מאפשר לפרט - יהיו אשר יהיו משקליו- לשחק טוב יותר בהסתכלות קדימה למיקום האריח החדש. אנו מאמינים שהמשך עבודה עם אלגוריתם משופר זה תוביל אותנו לתוצאות טובות אם נצליח להתגבר על ההתכנסות למקסימום המקומי כפי שקרתה בכל ההרצות, וכן על זמן הריצה הארוך מדי של חישוב דור.

על מנת להפחית את זמן הריצה הארוך, חשבנו להשתמש במנגנון ה**Early Stop** **[3]** המוכר בספרות, ולעצור חישובים של פרטים שאינם נראים לנו מבטיחים לפי התוצאות הראשוניות (למשל של 3 המשחקים הראשונים שלהם מתוך ה-10).

אחד הדברים המשמעותיים שלדעתנו כיוון את האבולוציה להתכנסות מקומית בריצה זו (תוחלת אריח אקראי), הוא הורדת כמות המשחקים לשם הפחתת זמן הריצה. כבר הראנו בדרכים שונות שכמות המשחקים משמעותית ביותר לחישוב נכון של הfitness, ועל כן מעניין לנסות את אותה הריצה עם **יותר משחקים לפרט**.

כמו כן, דבר שיכול לתרום מאד לתוצאות האבולוציה, היא להכניס כמות מסויימת של פרטים מוצלחים שמצאנו לתוך 200 הפרטים באוכלוסיה הראשונית ולהתחיל את ההרצה מאוכלוסיה זו. בתחום האלגוריתמים האבולוציונים יש את האסכולה שגורסת שצריך להתחיל עם אוכלוסיה אקראית ללא ידע מוקדם, בעוד אסכולה אחרת גורסת שניתן לעזור לשיפור התוצאות ע"י **הכנסת הידע המוקדם לאוכלוסיה הראשונית**. אנו חושבים שניסיון שכזה היה מוביל אותנו לתוצאות טובות בהרבה יותר.

כמו כן, כבר בתחילת הדרך חשבנו על המרת הGA שלנו ל**GP**, ולהכניס קצת יותר שכל לייצוג פרט או לחישוב המשחקים. בין הרעיונות שעלו, היה הרעיון **לייצג את הפרטים ע"י שני מערכי משקלים**: האחד לתחילת המשחק (כלומר כל עוד הscore נמוך יחסית), והשני לשלב מתקדם יותר. כך יכולים הפרטים ליישם שתי אסטרטגיות משחק שונות. שינוי זה נשמע הגיוני מאד למשחק כמו 2048, מאחר ובשלבים מתקדמים של המשחק יש פחות תאים ריקים וגיוון רחב יותר בערכי האריחים, כלומר הלוח נראה שונה לחלוטין.

רעיון נוסף שעלה היה הגדלת כמות המשחקים לפרט עבור פרטים מתקדמים (כלומר בעלי fitness גבוה). המוטיבציה מאחורי שינוי זה היא הפחתת זמן הריצה עבור פרטים פחות טובים, וכמו כן מתן סיכוי טוב יותר להתחרות מול פרטים בעלי fitness נוכחי גבוה יותר, וכך להתקדם ולהוציא אותנו מההתכנסות למקסימום המקומי. בשינוי זה עדיף לתמוך ע"י חילוק האוכלוסיה לשכבות גיל שונות (**Age Layer Structure [3]**), כך שכל קבוצת גיל תתחרה בינה לבין עצמה ותעלה את הפרטים הטובים ביותר לשכבה הבאה. שינוי זה, בנוסף לזריעת השכבה הצעירה בפרטים אקראיים חדשים יכול לטעמנו לתרום מאד לחיפוש נרחב יותר במרחב החיפוש ומכאן לתוצאות טובות יותר.

רעיון נוסף היה **הוספת רמות לעץ החישוב של המשחק**, ובכך לשפר את הסיכויים של השחקן להגיע למצבים טובים במשחק. כדי להתמודד עם תוספת האריח הרנדומלית יכולנו, לכל צעד, לחשב את התוחלת של כל האפשרויות להוספת אריח, דבר אשר יגדיל בצורה אקספוננציאלית את זמן הריצה, לכן הוספת רמות חישוב לעץ מחייבת שיפור זמן ריצת אלגוריתם האבולוציה.

# ביביליוגרפיה

[1] היסטוריית 2048 – בלוג של הכותבים של המשחק Threes:  
<http://asherv.com/threes/threemails/>

[2] אתר עם קישורים למשחקים רבים ושונים של 2048:

<http://allthe2048.com/>

[3] על עצירה מוקדמת ומודל השכבות (מאמר במסגרת הקורס):

<http://idesign.ucsc.edu/papers/bongard_gecco10b.pdf>

שחקנים אחרים:

[4] השירשור ב- stack overflow שבוא אנשים פירסמו את התוצאות שלהם לשחקנים שהם כתבו:

<http://stackoverflow.com/questions/22342854/what-is-the-optimal-algorithm-for-the-game-2048>

[5] המשחק של [Matt Overlan](https://github.com/ov3y/2048-AI) <http://ov3y.github.io/2048-AI>

[6] הקוד של [Matt Overlan](https://github.com/ov3y/2048-AI) <https://github.com/ov3y/2048-AI>

[7] הקוד של [nneonneo](http://stackoverflow.com/users/1204143/nneonneo) <https://github.com/nneonneo/2048-ai>

[8] המשחק של [Ronenz](http://stackoverflow.com/users/632039/ronenz) <http://ronzil.github.io/2048-AI/>

[9] הקוד של [Ronenz](http://stackoverflow.com/users/632039/ronenz) <https://github.com/ronzil/2048-ai-cpp>

[10] האתר של התוכנה HeuristicLab

<http://dev.heuristiclab.com/trac/hl/core>

מצגות במסגרת הקורס:

[11] [Game Playing: Adversarial Search](http://www.cs.umbc.edu/~ypeng/AI/471/lecture-notes/Ch06.ppt)

<http://www.cs.umbc.edu/~ypeng/AI/471/lecture-notes/Ch06.ppt>

[12] [Working with Evolutionary Algorithms](http://www.cs.vu.nl/~gusz/ecbook/slides/Working_with_EAs.ppt)

<http://www.cs.vu.nl/~gusz/ecbook/slides/Working_with_EAs.ppt>