



מיני פרויקט

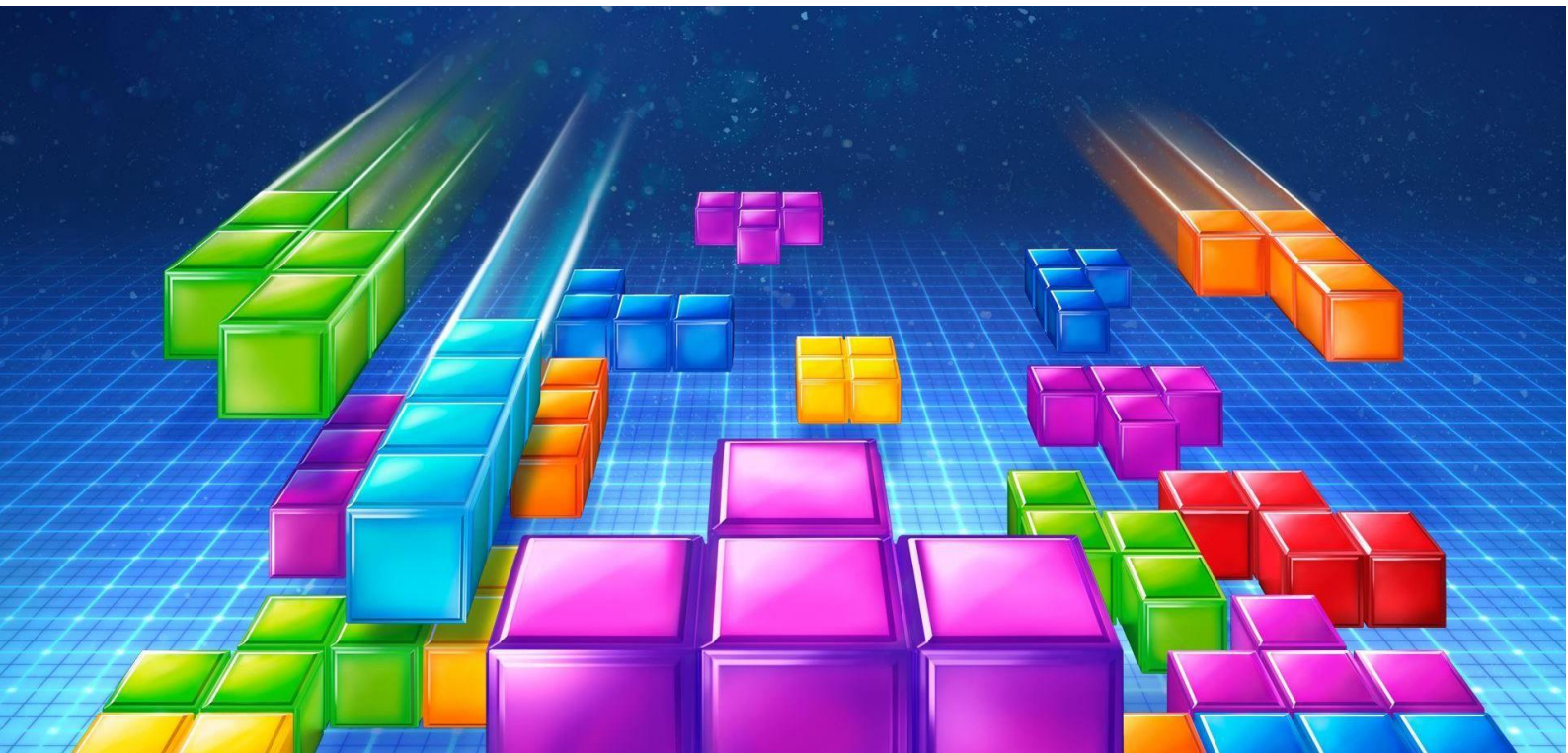
Tetris Genetic Algorithm

מגישים:

יובל אליאס

אדם לופז מועלם

רוני קדרון





תוכן עניינים:

| | |
|----------|---------------------------------|
| 3..... | מבוא |
| 5..... | תיאור הפרויקט |
| 7..... | תיאור התוכנה |
| 9..... | ניסויים, מהלך האבולוציה ותוצאות |
| 12 | מסקנות |
| 13 | הוראות הפעלה |
| 14 | ביבליוגרפיה |

מבוא

הקדמה - טטריס :


טטריס הינו המשחק משלב את האתגר של חשיבה מראש ותזמון, תוך כדי התמודדות עם חלקים מתפרסים במהירות גוברת.

מהלך המשחק :


במהלך המשחק על השחקן לכוון צורות המורכבות מקוביות בצבעים שונים אשר נופלות במהירות הולכת וגדלה מחלקו העליון של המסך .


- המשחק מורכב מלוח משחק מלבני בגודל 10×25 .


- ישנם 7 צורות שונות :

I (הטטרומינו הישר) - ארבעה ריבועים ברצף על קו אחד.  ○


O (הטטרומינו המרובע) - ארבעה ריבועים המרכיבים ריבועים של 2 על 2.  ○

T - שלושה ריבועים בקו כאשר מעל הריבוע האמצעי מוצב ריבוע נוסף.  ○

J - קו של שלושה ריבועים כאשר ריבוע מוצב מתחת לריבוע הימני ביותר בקו.  ○

L - קו של שלושה ריבועים כאשר ריבוע מוצב מעל לריבוע הימני ביותר בקו.  ○

S - שני ריבועים בקו הנמצאים מעל זוג ריבועים אחר ומוסטים ימינה.  ○

Z - שני ריבועים בקו הנמצאים מעל זוג ריבועים אחר ומוסטים שמאלה.  ○

- כל צורה מתחילה באמצע הלוח בחלק העליון.

- הצורות נבחרות בצורה אקראית.

- בכל רגע נתון ניתן לראות את הצורה הבאה שתגיע.

- הציון עולה בכל צורה חדשה שמונחת ובכל מילוי שורה שנמחקת.

המטרה היא לסדר את הצורות בתחתית המסך במידת האפשר ללא רווחים ברצף של קוביות. כשהקוביות יוצרות שורה שלמה ללא רווח, השורה נעלמת, והקוביות שנערמו מעליה נופלות "קומה" אחת. כשהקוביות נערמות עד לקצה העליון של המסך המשחק מסתיים.

שמו של המשחק נגזר מהמילה היוונית "טטרה", שפירושה "ארבע" משום שהצורות במשחק מורכבות מארבעה ריבועים. כל אחת מן הצורות הנופלות מטה במשחק הטטריס מהוות טטרומינו, כלומר צורות גאומטריות המורכבות מארבעה ריבועים בלבד.



הקדמה - אלגוריתם גנטי :

אלגוריתם גנטי הוא אלגוריתם חישובי המשתמש המבוסס על מושגים מתחום הגנטיקה והאבולוציה הביולוגית כדי לפתור בעיות אופטימיזציה ולמצוא את הפתרונות הטובים ביותר לבעיה הנתונה.

ריצת האלגוריתם מתחילה עם אוכלוסייה אקראית של פתרונות אפשריים המיוצגים כרצף של גנים. כל גן מייצג מאפיין או משתנה של הפתרון הפוטנציאלי והפתרונות מיוצגים ע"י קבוצה של גנים כאלה.

האלגוריתם מבצע סדרה של פעולות גנטיות , כמו בחירה, שחלוף ומוטציה, כדי ליצור את הדור הבא של פתרונות עם מאפיינים משופרים בהשוואה לדור הקודם.

בזכות השילוב של התהליכים הגנטיים האלו, האלגוריתם הגנטי יכול לחקור ביעילות את מרחב הפתרונות ולהתכנס לפתרונות אופטימליים או כמעט אופטימליים לבעיות מורכבות ומגוונות שונות.

תיאור הפרויקט

מטרת הפרויקט:

מימוש אלגוריתם גנטי של משחק הטטריס אשר יגיע לציון המיטבי במשחק.

מרחב הפתרונות האפשריים:

מרחב הפתרונות בטטריס מורכב מהצורות האפשריות, אפשרויות למיקום ולסיבוב של הטטרומינואים בתוך לוח המשחק, הקשר בין הצורות וכו'. מרחב זה הופך לאקספוננציאלי ככל שהמשחק מתקדם, מכיוון שכל טטרומינו יכול להימצא במספר מצבים ומיקומים שונים.

תנאי זכייה:

המשחק מחשיב משחק כניצחון אם הניקוד עולה על ציון מקסימלי מוגדר מראש (max_score), אשר מוגדר ל-22,000 נקודות כברירת מחדל באלגוריתם. אם תנאי זה מתקיים לולאת המשחק מסתיימת.

הבעיות שניצבות מול השחקן:

- ניהול מרחב: האלגוריתם עם מאפייני הפרטים צריך לנהל את המרחב המוגבל בצורה יעילה, כדי למנוע מצב שבו אין אפשרות להוסיף טטרומינואים נוספים.
- תכנון וחזיון: האלגוריתם צריך להעריך את התוצאות האפשריות של כל פעולה שהוא עושה, ולבחור באופציה הטובה ביותר לפי מאפייני הפרט שלפיו הוא רץ.

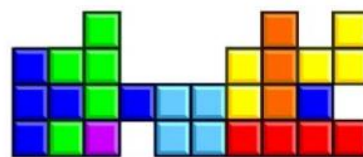
מבנה ומאפייני הפרט:

כל פרט באוכלוסייה מיוצג על ידי וקטור של מספרים ממשיים באורך 7, כאשר כל ערך בו שייך ל $[-1.0, 1.0]$. כל אחד מהערכים בוקטור יכול לייצג פרמטר שונה שמשפיע על התנהגות האלגוריתם במשחק הטטריס. הפרמטרים הם:

- מספר חורים:** חורים מוגדרים כחללים ריקים בלוח שבהם יש לפחות תא מלא אחד מעל. המטרה היא לצמצם את מספר החורים מכיוון שפעולה זו יכולה להקל על מילוי הלוח עם החלקים הבאים במשחק.
- מספר הריבועים מעל חורים:** גורם זה מייצג את מספר התאים המלאים שנמצאים ישירות מעל חורים. בלוקים מעל חורים יכולים להקשות על מחיקת שורות מלאות ולהגדיל את הגובה הכללי של הלוח.
- גובה מקסימלי:** זהו גובה העמודה הגבוהה ביותר על הלוח. שמירה על גובה הלוח נמוך חשובה מכיוון שהיא מספקת יותר גמישות להנחת חלקים במהלכים המשחקים הבאים.
- מספר שורות מלאות שהוסרו:** כאשר יש שורה מלאה, היא מוסרת מהלוח. מספר השורות שהוסרו על ידי הנחת הטטרומינו הנוכחי נחשב כציון חיובי מכיוון שהוא מפנה מקום על הלוח.
- שכנים מצדדי הטטרומינו:** מציין את מספר הצדדים של החלק הנוכחי שנמצאים במגע עם חלקים אחרים או עם קצוות הלוח.
- מספר הריבועים בתחתית הלוח:** מציין את מספר הצדדים של הבנייה הנוכחית שנמצאים במגע עם החלק התחתון של הלוח.

7. מספר הריבועים הנמצאים במגע עם צידי הלוח: מציין את מספר הצדדים של הבנייה הנוכחית שנמצאות במגע עם קירות הלוח.

מספר חורים = 2
 מספר הריבועים מעל חורים = 3
 גובה מקסימלי = 4
 מספר הריבועים בתחתית הלוח = 9
 מספר הריבועים הנמצאים במגע עם צידי הלוח = 6



אתחול האוכלוסייה:

נאתחל m (גודל האוכלוסייה שנבחר) וקטורים באורך 7 (מספר הגנים) כך ש: $vector[i] = k \in [-1.0, 1.0]$ בשלב האתחול כל k נבחר באקראי.

נדגיש כי אופן פעולת האלגוריתם מתבצעת כך שבדור הראשון שמאותחל ישנם $2m$ פרטים ולאחר מכן כל דור מכיל m פרטים.

חישוב fitness:

פונקציית הפיטנס הינה פשוטה, המשחק מתקיים והניקוד מצטבר עד ההפסד. בעת סיום המשחק, ערך פונקציית הפיטנס של הפרט הינה הניקוד של המשחק. המשחק מקבל ציונים על ידי תגמול הדרגתי של כל טרמינו חדש, עם בונוסים משמעותיים עבור מחיקת שורות מלאות, במיוחד עבור מחיקת שורות מרובות במהלך אחד באופן הבא:

- תוספת ניקוד התחלתית: בכל פעם שנעשה שימוש בטורמינו חדש (כלומר, כאשר צורה חדשה נוצרת ומתחילה ליפול), הניקוד גדל ב-1. זוהי תוספת בסיסית המתרחשת ללא קשר לפעולות הננקטות עם הבנייה.
- הוספת נקודות עבור כל שורה שנמחקה בצורה הבאה:
 - ניקוי שורה אחת במהלך בודד: + 40 נקודות
 - ניקוי 2 שורות במהלך בודד: + 120 נקודות
 - ניקוי 3 שורות במהלך בודד: + 300 נקודות
 - ניקוי 4 שורות במהלך בודד: + 1200 נקודות

יצירת דור חדש:

כדי ליצור פרטים חדשים ומשופרים נבצע מוטציות ו crossover נבצע על שני וקטורים של פרטים באוכלוסייה בהסתברות 0.5. על כל וקטור באוכלוסייה מוטציה בהסתברות של p . בהמשך נציג ביצוע ניסויים על מנת למצוא את ה- p האופטימלי. פונקציית המוטציה בוחרת 3 פרמטרים מתוך הוקטור ומבצעת עליהם מוטציה אקראית. את האוכלוסייה החדשה נבחר על ידי ביצוע טורניר בגודל 4, בין הפרטים באוכלוסייה הקודמת.

תיאור התוכנה

הפרויקט משתמש באלגוריתם שמריץ את משחק הטטריס עם תוספת של אינטליגנציה מלאכותית המבוססת על אלגוריתמים גנטיים באמצעות הספרייה pygame לצורך הדמיית המשחק והספרייה eckity לחישובים אבולוציוניים.

תיאור הקבצים:

- **eckity_main.py**: משמש כנקודת הכניסה העיקרית לשימוש בספריית EC-KitY מאתחל את תהליך האלגוריתם הגנטי, הגדרת הסביבה, הפרמטרים והלולאה הגנטית.
- **tetris_evaluator.py**: בהינתן הווקטור של הפרט מבצע את חישוב הפיטנס ומחזיר את ציון הריצה.
- **main.py**: במחלקה זו מתבצע הניתוב לקטע הקוד המתאים לפי בקשת המשתמש - האם להריץ את האלגוריתם הגנטי או להריץ משחק עבור שחקן בודד.
- **tetris_plot.py**: קובץ המכיל מחלקה Information בה אנו משתמשים כדי ליצור גרף מהנתונים של ריצה.
- **tetris_base.py**: קובץ זה הוא מרכז הריצה של המשחק. אחראי על הצגת המשחק באמצעות ספריית pygame ופונקציות שונות של הצגת המשחק. בנוסף, אחראי על אתחול המשחק, משתמש בפונקציות למכניקת משחק כגון תנועת טרמינו, סיבוב וזיהוי התנגשות, הפעלת לולאת המשחק עבור שחקן בודד או אלגוריתם גנטי. מכיל פונקציות המסייעות בהרצת המשחק וחישוב הציון, ופונקציות הערכת מהלכים:
 - **remove_complete_rows**: פונקציה הבודקת לאחר התור האם ישנם שורות מלאות, מסירה אותן, מורידה שורה את הריבועים שמעליהם ומחזירה את מס' השורות שנמחקו.
 - **get_blank_board**: פונקציה המאתחלת לוח משחק חדש וריק.
 - **is_valid_position**: הפונקציה בודקת אם ניתן למקם כלי נתון במיקום מסוים על לוח המשחק מבלי להתנגש בכלים קיימים או לצאת מחוץ לתחום.
 - **calc_move_info**: הפונקציה מעריכה את התוצאות הפוטנציאליות של הצבת כלי טטריס במיקום מסוים ובסיבוב על לוח המשחק.
 - **calc_sides_in_contact**: קובעת את מספר הצדדים של חתיכת טטריס במגע עם בלוקים אחרים, הרצפה והקיר על לוח המשחק.
 - **calc_heuristics**: מחשבת פרמטרים הכוללים את המספר הכולל של החורים, ספירת הריבועים מעל כל חור, והגובה המרבי של העמודות.
- **tetris_ai.py**: הקובץ האחראי לסימולציית המשחק לפי האלגוריתם הגנטי. מכיל בתוכו את הפונקציות הבאות:
 - **run_game(chromosome)**: הפונקציה מאתחלת את המשחק באמצעות tetris_base.py ומדמה ריצות משחק שבהן המהלך הטוב ביותר של הפרט נקבע על ידי הכרומוזום. משחק ברציפות את המשחק, מעריך כל מהלך על סמך האסטרטגיה של הפרמטרים בכרומוזום עד שמגיעים לניקוד מקסימלי או שהמשחק מסתיים, מחזיר סטטיסטיקות לגבי הריצה של משחק בודד כמו מספר החלקים בשימוש, שורות שהוסרו, ציון סופי והאם המשחק זכה.
 - **chromosome.py**: קובץ זה מכיל את המחלקה שמייצגת את תכונות הפרט ומכילה את ווקטור הכרומוזום וציונו, ובנוסף במחלקה זו מתבצע חישוב המהלכים לפי הווקטור. למחלקה פרמטרים של וקטור וניקוד. מחלקה זו מכילה פונקציות לחישוב מהלכים לפי הווקטור:



- **calc_best_move**: פונקציה המחשבת את המהלך הכי טוב לפי הפרמטרים שנתונים בווקטור של הפרט.

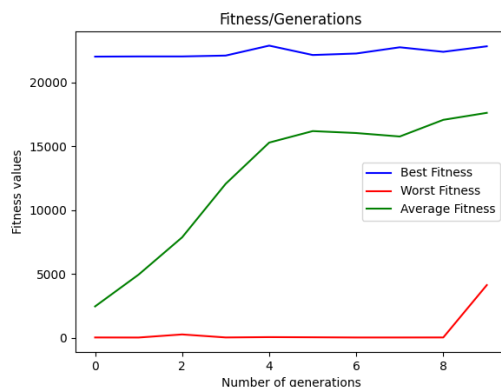
ניסויים, מהלך האבולוציה ותוצאות

מציאת מס' הדורות האופטימלי לריצת האלגוריתם:

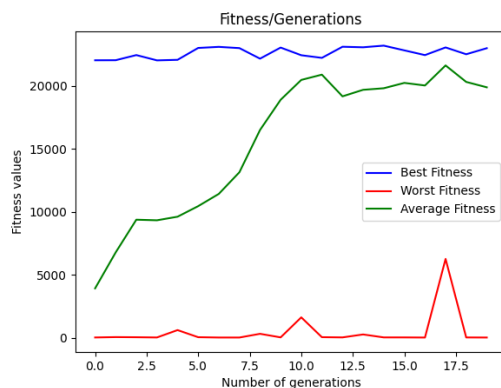
תחילה ניסינו להבין אילו פרמטרים של גודל אוכלוסיה ומספר דורות אנו צריכים לבחור בכדי לקבל תוצאה אופטימלית עבור האלגוריתם.

התחלנו את הניסויים עם גודל אוכלוסיה קבוע של 100 וריצה על מס' דורות 10,20,40

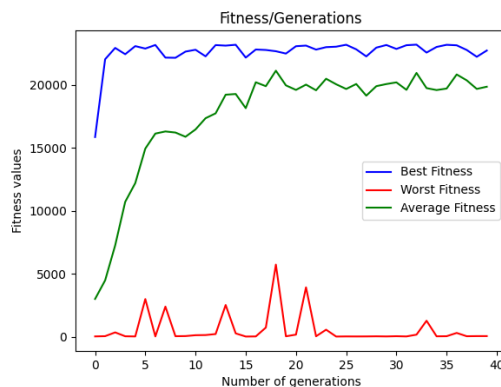
10 דורות:



20 דורות:



40 דורות:

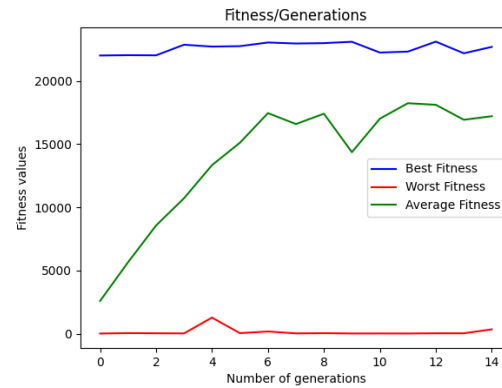


ניתן לראות כי לאחר 15 עד 18 דורות בהרצות המופרטות מעלה, ציון המשחק הממוצע מתחיל לשאוף לערך מסוים לכן ניתן להעריך על בסיס נתונים אלו כי מס הדורות האופטימלי הוא 15 דורות.

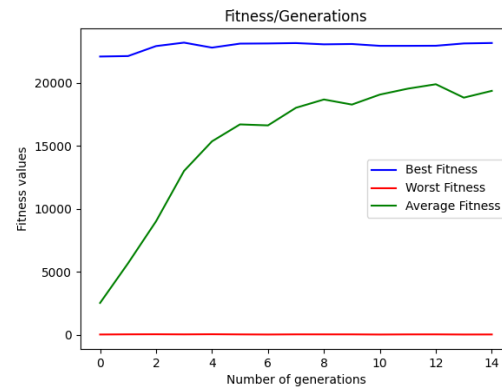
מציאת גודל האוכלוסייה האופטימלי לריצת האלגוריתם:

בנוסף, ראינו כי עבור גודל אוכלוסייה קטן יחסית מקבלים תוצאות שאינן מתכנסות לערך מסוים ולכן החלטנו לבדוק גדלים שונים של אוכלוסיות. הרצנו את האלגוריתם האבולוציוני עם גדלי האוכלוסיות הבאים: 100, 400, 1000.

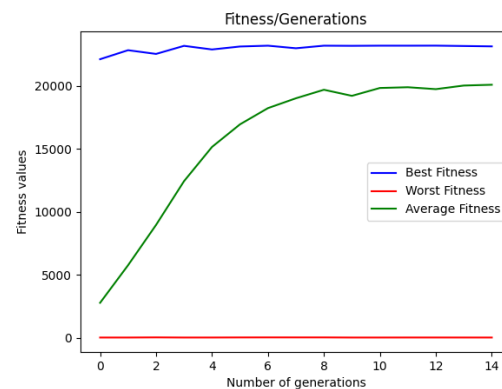
אוכלוסייה של 100:



אוכלוסייה של 400:



אוכלוסייה של 1000:



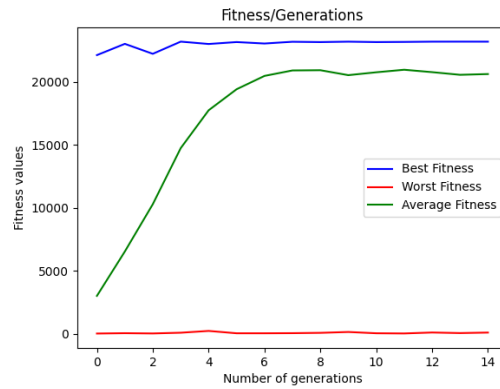
ניתן לראות כי ככל שהאוכלוסייה גדולה יותר, הערך הממוצע מתקרב לערך \max_score שהוגדר עבור המשחק.

מציאת ההסתברות האופטימלית לביצוע מוטציה על פרט באוכלוסייה:

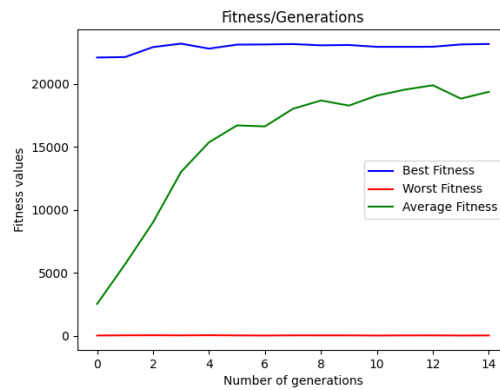
מטרה נוספת הייתה לבדוק ערך הסתברותי אופטימלי לביצוע מוטציה על פרט באוכלוסייה.

בדקנו את ההסתברויות הבאות: 0.01, 0.05, 0.09. עבור אוכלוסייה בגודל 400 פרטים ו15 דורות.

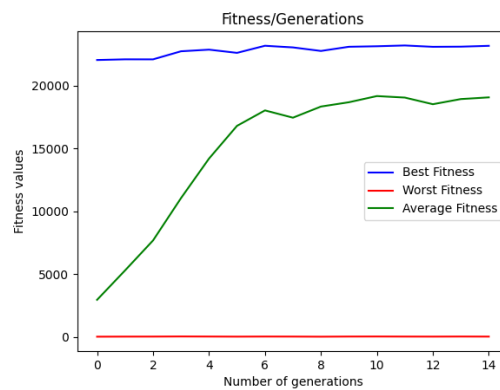
הסתברות למוטציה $p = 0.01$:



הסתברות למוטציה $p = 0.05$:



הסתברות למוטציה $p = 0.09$:



ניתן לראות מהגרפים כי ככל שההסתברות למוטציה קטנה יותר השאיפה לערך \max_score תתחיל לאחר מס' קטן יותר של דורות.

מסקנות

מס הדורות הנדרש להתכנסות לערך \max_score :

עבור גודל אוכלוסייה של 400 פרטים הרצנו את האלגוריתם מספר רב של פעמים עם מספר דורות טווח בין 10-40 וקיבלנו כי אין התכנסות חד משמעית לערך המיטבי אך ככל שמס' הדורות גדל, הערכים שהתקבלו נמצאו יחסית בקרבתו.

נציין כי מספר קטן של דורות לא מאפשר לאלגוריתם מספיק "זמן" על מנת להגיע לפתרון כמעט אופטימלי.

לכן החלטנו להגדיל את גודל האוכלוסייה ולבצע את הניסויים הבאים עם 14 דורות.

גודל אוכלוסייה אופטימלי:

מהבדיקות שביצענו על מנת למצוא את גודל האוכלוסייה האופטימלי ניתן לראות שככל שגודל האוכלוסייה גדול יותר יש מגוון פרטים יותר גדול בכל דור ולכן מגיעים לציון הכמעט אופטימלי באופן יותר מהיר.

עבור אוכלוסיות בגודל 400 ו-1000 ראינו כי ההתכנסות לערך המיטבי יותר משמעותי עבורם. לכן לניסויים הבאים בחרנו בגודל אוכלוסייה 400, החיסרון העיקרי לבחירה זו הוא שלא אלגוריתם נדרש כמעט 3-4 דורות יותר על מנת להתכנס לערך האופטימלי, אך זה יותר יעיל מבחינת זמן הריצה מאשר ריצה עם 1000 פרטים לכל דור.

ערך הסתברותי אופטימלי ליצירת מוטציה בפרט:

מהגרפים שקיבלנו, ראינו כי שככל שההסתברות ליצירת מוטציה גדולה יותר - כך מס הדורות הנדרש לשאיפה לערך המיטבי גדול יותר.

ייתכן כי אם היינו מריצים את האלגוריתם על מס דורות ואוכלוסיות גדול משמעותית, היינו מקבלים תוצאות שונות.

הוראות הפעלה

- הורידו את התיקייה, ופתחו command prompt בתקיה הראשית.
 - התקינו את החבילות הנדרשות : `pip install -r requirements.txt`
 - הריצו את התוכנית באחת משתי האפשרויות :
 - הרצת אלגוריתם גנטי : `python main.py --ga`
 - הרצה למשחק בודד : `python main.py --game`
- הוראות הפעלה למשחק בודד :
- חץ שמאלה או מקש A : הזז את היצירה שמאלה.
 - חץ ימינה או מקש D : הזז את היצירה ימינה.
 - חץ למעלה או מקש W : סובב את היצירה בכיוון השעון.
 - חץ למטה או מקש S : להאיץ את מהירות נפילת היצירה.
 - מקש רווח : הנחתה מיידית של היצירה מטה.
 - מפתח P : השהה את המשחק.
 - לחיצה על סגירת החלון : עצירת המשחק.



ביבליוגרפיה

1. "Tetris Guideline" (2018) - [Link](#)
2. "Genetic and Evolutionary Algorithms and Programming: General Introduction and Application to Game Playing" (2009) - [Link](#)
3. EC-KitY – Python toolkit for evolutionary computation - [Link](#)
4. Pygame – Python modules tailored for video game development - [Link](#)
5. "Playing Tetris with Genetic Algorithms" (Machine Learning - Stanford University, 2015) - [Link](#)
6. "Genetic Tetris: A Python Implementation of Tetris using Genetic Algorithms" by Thyoma J. (2018) - [Link](#)