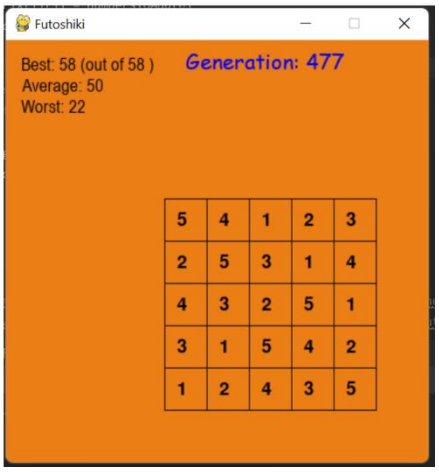
מגישים: רביב חכם, 208387951
פלג חכם, 208387969

ביולוגיה חישובית – תרגיל 2

קישור לגיט:
(בגיט מופיעים כל קבצי הפיתון המרכיבים את התוכנית שבנינו, הדוח הנ"ל, ובפרט קובץ ההרצה main.exe)
https://github.com/peleg5050/Biologi2/tree/master

הנחיות והסבר הרצה:
ישנן שתי דרכים להריץ את התוכנית שבנינו:
דרך ראשונה (הקלה): יש ליצור ולהיכנס לתיקייה בה מופיע קובץ ההרצה main.exe וקובץ הטקסט (שמתאר את מצב הלוח), יש לפתוח את ה cmd עבור נתיב זה (ע"י כתיבה "cmd" בשורת הנתיב) ובטרמינל שנפתח יש לכתוב: main.exe filename.txt
דרך שנייה: ע"י ביצוע ההרצה מהטרמינל של סביבת הפיתוח pycharm ע"י הרצת הפקודה: py main.py filename.txt

קצת על הפרויקט:
יצרנו מעיין אפליקציית desktop אשר מריצה אלגוריתם גנטי שפותר לוח חידה פוטושיקי (בהתאם להנחיות שסופקו לנו בתרגיל).
כמובן שבהתאם לקוד שכתבנו ניתן להריץ כל לוח משחק בכל גודל שרוצים.



מסכים:
ישנו דף עם תצוגה גרפית, נוחה וידידותית למשתמש, שמריצה ברצף את כל סוגי האלגוריתמים הגנטיים שבנינו, כך שבמרכז המסך נמצא לוח המשחק בגודל N X N תאים, והלוח מלא במספרים מ 1 עד N כך שבתאים אשר נתונים כקלט אכן קיימות אותן הספרות הקבועות לכל אורך הניסיונות (שכן זהו אילוץ קשה). על המסך מופיע למעשה הפתרון הטוב ביותר מבין 300 הפתרונות שהאלגוריתם מריץ עבור הדור וסרגל מידע בחלקו השמאלי של המסך, המכיל את מספר האיטרציה הנוכחית, הניקוד של הפתרון הטוב ביותר, הניקוד שקיבל הפתרון הגרוע ביותר וכן את הציון הממוצע שהתקבל. כאשר הריצה מסתיימת נפתחים 2 גרפים שמציגים את ציון הפתרון הטוב ביותר והציון הממוצע שהתקבלו כתלות במספר האיטרציה עבור כל שלושת האלגוריתמים השונים. כך נראה הלוח עבור הרצה של האלגוריתם הגנטי הרגיל (הבסיסי) עם הלוח שפורסם כדוגמה בתוך התרגיל עצמו:

סעיף א :

בסעיף זה היה עלינו לייצר אלגוריתם גנטי אשר מנסה למצוא פתרון לקלט כלשהו של לוח משחק, בהתאם לחוקי משחק הפוטושיקי ובהתאם לאילוצי הלוח הנוכחי. יש לציין שביצענו מספר רב של ניסיונות בשיטת ניסוי וטעיה עד שמצאנו את האלגוריתם שמביא לתוצאה הטובה ביותר.
האלגוריתם שמימשנו משתמש באוכלוסייה בגודל 300 (כלומר בכל דור מריץ 300 פתרונות במקביל) ועובד באופן הבא:

1. **ייצוג הפתרונות:**
הפתרונות במשחק מיוצגים ע"י מטריצה (כלומר מערך דו ממדי) בגודל N X N תאים (כאשר N הינו הגודל של הלוח הנוכחי שקיבלנו כקלט מהמשתמש). נשים לב כי כל תא בלוח הינו מספר מ 1 עד N כך שבתאים אשר נתונים כקלט עבור לוח המשחק הנוכחי אכן קיימות אותן הספרות הקבועות לכל אורך הניסיונות (שכן זהו אילוץ קשה אותו לא ניתן להפר), ובסה"כ בלוח כולו קיימות כל הספרות שצריכות להיות (כלומר מצב בו כל מספר מ 1 עד N מופיע בדיוק N פעמים), ובמידה ולא, אנו מחליפים את הספרות אשר מופיעות יותר פעמים מהכמות שאמורה להיות, בספרות החסרות, כך שמבצעים את פעולת בחירה זו באופן רנדומלי. באופן כללי, על המסך מופיע למעשה אחד מבין 300 הפתרונות שהאלגוריתם מריץ עבור הדור הנוכחי, שזהו גם כמובן הפתרון הטוב ביותר. נשים לב כי בכל שלב ישנם 300 פתרונות שרצים במקביל, והדבר מיוצג ע"י מערך שמכיל את כל המטריצות הללו.

2. **פונקציית ההערכה:**
במימוש שלנו לפונקציית ההערכה קראנו getGrade. פונקציה זו מקבלת פתרון (מטריצה דו ממדית מבין הפתרונות בדור), רצה עליו תחילה לפי שורות כך שעבור כל ספרה שונה בשורה הפונקציה מעלה את הניקוד ב1. לאחר מכן רצה לפי עמודות כך שעבור כל ספרה שונה בעמודה הניקוד גדל ב1. לבסוף הפונקציה עוברת על כל אילוצי הגדול שווה ועבור כל אילוץ שמתקיים הפונקציה מעלה את הניקוד ב1. באופן זה מחושב ניקוד הלוח. כך ככל שהציון גבוה יותר, האלגוריתם מתקרב יותר לפתרון.

3. **הכלאה (cross-over) בין פתרונות שונים:**
במימוש שלנו לפונקציית ההכלאה קראנו createCrossover. פונקציה זו (אשר נקראת רק בסיכוי מסוים אשר נקבע מראש להיות 30 אחוז) מקבלת 2 פתרונות אשר נבחרו מהדור הקודם, לאחר מכן מגרילה מספר שורה (מספר בין 1 ל N) ולוקחת את השורות שהם עד השורה שהוגרלה מהפתרון הראשון, ואת השורות שהינם מהשורה שהוגרלה מהפתרון השני (ועד הסוף) ובכך יוצרת פתרון חדש (כלומר מטריצה אשר מהווה פתרון) אך כפי שכבר כתבנו, אנו דואגים שתמיד הפתרון יכיל אך ורק את הספרות החוקיות שאמור להכיל. לכן לאחר שנוצר לנו הפתרון החדש אנו עוברים עליו ובודקים שאכן קיימות בו כל הספרות שצריכות להיות, ובמידה ולא, אנו מחליפים את הספרות, אשר מופיעות יותר פעמים מהכמות שאמורה להיות, בספרות החסרות, כך שמבצעים את פעולת בחירה זו באופן רנדומלי, ובכך הפתרון נשאר חוקי.

4. **מוטציות:**
במימוש שלנו לפונקציית שיוצרת את המוטציות קראנו createMutation. פונקציה זו (אשר נקראת רק בסיכוי מסוים אשר נקבע מראש להיות 70 אחוז, אך יורד במהרה ל15 אחוז) מקבלת פתרון כלשהו אשר נבחר מהדור הקודם, עוברת על כל התאים בו, כך שעבור כל תא

שאינו תא שערכו נקבע מראש (כחלק מהגדרת הלוח) מבצעת בסיכוי של 30 אחוז הגרלה של מספר (בטווח שבין 1 ל N) וקובעת את הערך שהוגרל להיות הערך החדש של התא, ובאופן זה יוצרת פתרון חדש (כלומר מטריצה אשר מהווה פתרון) אך כפי שכבר כתבנו, אנו דואגים שתמיד הפתרון יכיל אך ורק את הספרות החוקיות שאמור להכיל. לכן לאחר שנוצר לנו הפתרון החדש אנו עוברים עליו ובודקים שאכן קיימות בו כל הספרות שצריכות להיות, ובמידה ולא, אנו מחליפים את הספרות, אשר מופיעות יותר פעמים מהכמות שאמורה להיות, בספרות החסרות, כך שמבצעים את פעולת בחירה זו באופן רנדומלי, ובכך הפתרון נשאר חוקי.

5. **בעיית ההתכנסות המוקדמת:**

כדי לפתור את בעיית ההתכנסות המוקדמת העלנו את אחוז ה cross-over והמוטציות ל 60 אחוז עבור כל 1200 דורות, וזה נשאר כך למשך 20 דורות, ולאחריהם החזרנו את האחוז להיות כפי שהיה. בנוסף בדקנו מתי הציון נשאר קבוע, ובכך במידה והציון נשאר קבוע למשך יותר מ 100 דורות, הגדלנו את אחוז המוטציות ל 40 אחוז וזה נמשך למשך 10 דורות.
בנוסף הגדרנו כי כאשר אנו במצב בו במשך 100 דורות רצופים הציון הטוב ביותר נשאר זהה, ניקח רק פתרון אחד (הפתרון הטוב ביותר) לדור הבא ונבצע זאת למשך 15 דורות ואז נתחיל ספירה מחודשת של דורות "תקועים" (מצב בו במשך 100 דורות רצופים הציון הטוב ביותר נשאר זהה). כמו כן, במידה ועברו יותר מ 600 דורות אזי כל 150 דורות אנו מבצעים את יצירת הדור החדש באופן מעת שונה, בכך שאנו קודם כל יוצרים 100 פתרונות חדשים לגמרי ולהם יוצרים הכלאה עם 20 הפתרונות הטובים ביותר שהיו בדור הקודם. את שאר הפתרונות שנשאר להוסיף אנו מוסיפים בדיוק באותו אופן של המקרה הרגיל. בעזרת שיטות אלו ניסינו ליצור גיוון בפתרונות השונים ובכך לצאת מבעיית ההתכנסות המוקדמת.

6. **בחירת הדור הבא:**

במימוש שלנו לפונקציית שיוצרת את הדור הבא קראנו initNewGen. בכל דור אנו לוקחים את חמשת הפתרונות הטובים ביותר שהיו בדור הנוכחי, כפי שהם, אל הדור הבא. בנוסף הגדרנו כי כאשר אנו במצב בו במשך 100 דורות רצופים הציון הטוב ביותר נשאר זהה, ללא שינוי, ניקח רק פתרון אחד (הפתרון הטוב ביותר) לדור הבא ונבצע זאת למשך 15 דורות ואז נתחיל ספירה מחודשת של דורות "תקועים" (מצב בו במשך 100 דורות רצופים הציון הטוב ביותר נשאר זהה).
כעת, את יתר הפתרונות שנשארו להוסיף (כדי להשלים לכמות הלוחות שיש בכל דור) אנו בוחרים באופן הבא: אנו מגרילים לוח פתרון מהדור הקודם, ובוחרים אותו בסיכוי של הציון שקיבל חלקי הציון המקסימלי שיכול להתקבל (ובכך פתרון עם ציון גבוהה יבחר בסיכוי גבוהה ולהיפך). במידה והפתרון לא נבחר נחזור על פעולה זו עד אשר פתרון כלשהו יבחר. כעת בסיכוי של 10 אחוז נבצע פעולת הכלאה לפתרון זה עם פתרון אחר (שנבחר בדיוק באותו האופן שתיארתי).
על הלוח שהתקבל זה עתה, בסיכוי של 15 אחוז נבצע מוטציה. במידה ועברו יותר מ 600 דורות אזי כל 150 דורות אנו מבצעים את יצירת הדור החדש באופן מעת שונה, בכך שאנו קודם כל יוצרים 100 פתרונות חדשים לגמרי ולהם יוצרים הכלאה עם 20 הפתרונות הטובים ביותר שהיו בדור הקודם. את שאר הפתרונות שנשאר להוסיף אנו מוסיפים בדיוק באותו אופן שתיארנו במקרה הרגיל.

7. **משך הרצת האלגוריתם:**

נשים לב שאנו מריצים את האלגוריתם השונים ברצף, כך שכל אחד מהם רץ למשך 3000 דורות (גם אם הגיע להצלחה אפשרנו לו לרוץ לצורך ניתוח הגרפים בצורה טובה). פעמים רבות על מנת להגיע להצלחה נדרשו יותר מ 3000 דורות, ובאופן כללי הדבר תלוי הרצה (כמצופה מאלגוריתם גנטי). כמו כן, אנו מציגים את הפתרון הטוב ביותר שהתקבל בדור ואת הציון שלו, וניתן לעצור את האלגוריתם בכל שלב שרוצים (באופן כזה שעדיין רואים את הפתרון) ע"י לחיצה על מקש ה space. נדגיש כי כל ריצה עשויה להיות שונה במידת ההצלחה ומשך זמן עד התכנסות שכן האלגוריתם מתבסס על רינדום.

סעיף ב :

נרצה להשוות בין האסטרטגיות השונות (האלגוריתם הגנטי הרגיל, האלגוריתם הדארוויני והאלגוריתם הלאמארקי). ראשית נשים לב להבדל בין האלגוריתם הגנטי הדארוויני ללאמארקי. בשני האלגוריתמים ביצענו אופטימיזציה על כל הפתרונות ורק לאחר מכן נקבע את הfitness של הפתרון. השוני הינו שבאלגוריתם הדארוויני הדור הבא נוצר על פי הפתרון המקורי **לפני** האופטימיזציה ואילו בלאמארקי הדור הבא נוצר על פי הפתרון המקורי **אחרי** האופטימיזציה.

תיאור האופטימיזציה:

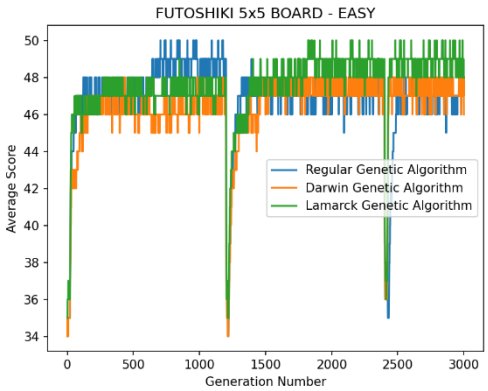
נשים לב כי מספר צעדי האופטימיזציה שבצענו בכל דור הינו 5. כחלק מתהליך האופטימיזציה שבצענו על כל פתרון, בצענו אופטימיזציות בלולאה שרצה כל עוד לא ממשנו את כל צעדי האופטימיזציה שמותרים לנו ועומדים לרשותנו, לכן סרקנו תחילה את כל התאים שהינם חלק מאילוצי הגדול שווה, בדקנו מי לא מקיים את התנאי והוספנו את תאים אלו למערך, כך שבמידה ואחד מהתאים היה תא מקובע (שהוגדר מראש כי בתא זה חובה שיהיה מספר כלשהו) אזי לא הוספנו את זוג תאים אלו אל המערך.
כעת רצנו על המערך הנ"ל ועבור כל זוג תאים שהפרו את האילוץ ביצענו החלפה וחישבנו את הציון שמתקבל. כעת האילוץ אומנם מתקיים אך אין זה מחייב כי ציון הלוח גדל בעקבות החילוף. לכן בדקנו אם ציון הלוח השתפר, ואם כן עדכנו את ציון הלוח להיות הציון החדש (שלאחר האופטימיזציה). נדגיש- עבור האלגוריתם הדרוויני יצרנו את הדור הבא על סמך הפתרון המקורי שהיה לפני האופטימיזציה ואילו בלאמארקי הדור החדש נוצר על סמך הפתרון המקורי שאחרי האופטימיזציה (במידה והדבר היה כדאי).
במידה ובשלב זה לא ממשנו את כל צעדי האופטימיזציה שעומדים לרשותנו בצענו אופטימיזציה נוספת בה נבחר זוג תאים באופן רנדומלי ובמידה ושני תאים אלו אינם תאים קבועים ביצענו ביניהם חילוף, חישבנו את הציון החדש שהתקבל ובמידה והציון השתפר עדכנו את הfitness של הפתרון (ושוב במידה ואנו בלאמארקי יצרנו את הדור הבא על סמך הפתרון המקורי שאחרי האופטימיזציה).

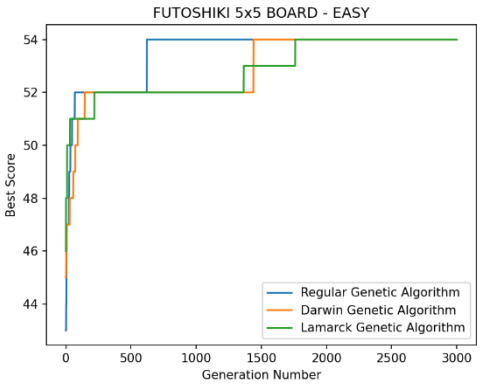
השוואה בין שלושת סוגי האלגוריתמים:

בס"ד

את ההשוואה נבצע עבור דרגות קושי שונות של הלוחות כלומר השווה בשתי רמות קושי: easy ו tricky ובשלושה גדלים 5*5, 6*6, 7*7.
בכל פעם נציג גרף של ה best ולצידו גרף של הממוצע. נדגיש כי התוצאות משתנות מהרצה להרצה בגלל רנדומיות. כמו כן נתנו לכל האלגוריתמים לרוץ במשך 3000 דורות (כך שגם אם לוח הגיע לפתרון הוא המשיך לנסות לפתור וכך נוצרה אחידות).

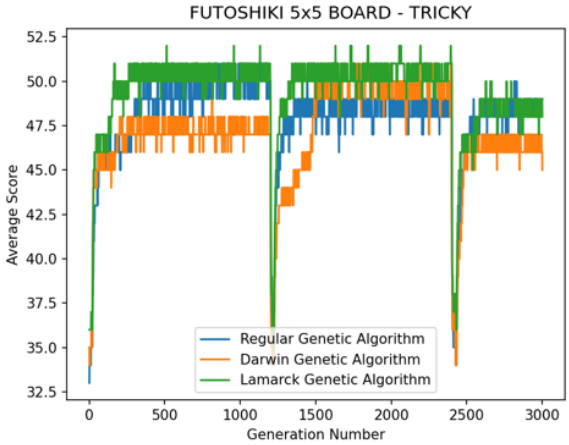
Average Score Per Generation Graph

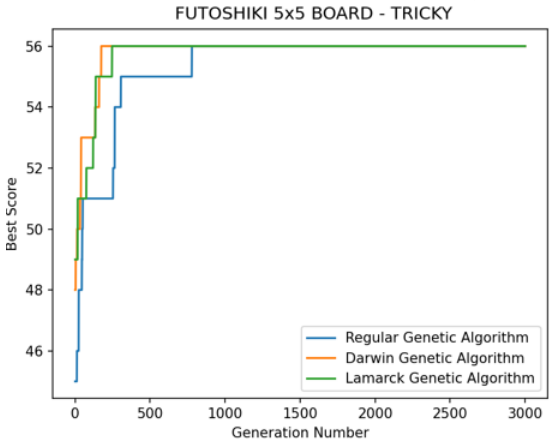




רמת קושי easy - לוח 5*5 :

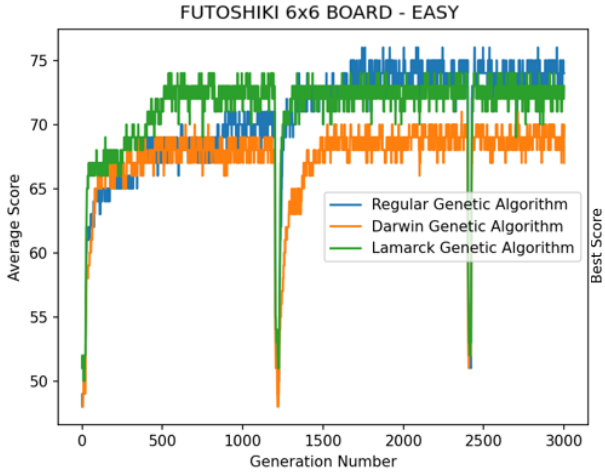
אפשר לראות **שבגרף ה best (הימני)** כל האלגוריתמים הצליחו לפתור את הלוח. יחד עם זאת נציין כי האלגוריתם הגנטי הרגיל התכנס יותר מהר מהדרוואני והלאמארקי, כך שתוך 600 דורות פתר את הלוח. האלגוריתם הדרוואני פתר תוך כ 1500 דורות ולאחריו גם הלאמארקי פתר תוך 1750דורות. סה"כ כלל האלגוריתמים הגיעו לפתרון (ואף האלגוריתם הגנטי הרגיל התכנס מהר יותר) דווקא כי מדובר בלוח קל שלא צריך לוגיקה מסובכת, לכן הרנדומיות שלו והעובדה שאינו עובד לפי לוגיקה מסובכת גרמו להתכנסות הכי מהירה. ניתן לראות **בגרף של הממוצע (השמאלי)**, כי האלגוריתם הרגיל התחיל הכי טוב והתכנס במהירות אך במשך הדורות הלאמארקי בממוצע בעל הציון ההכי גבוהה, לאחריו הדרוואני והרגיל הכי פחות טוב.

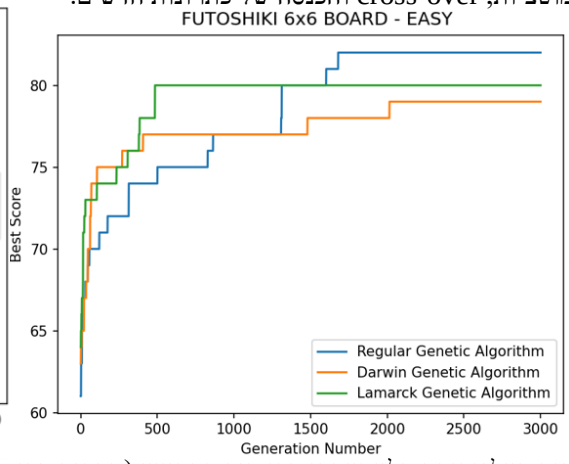




רמת קושי tricky-לוח 5*5 :

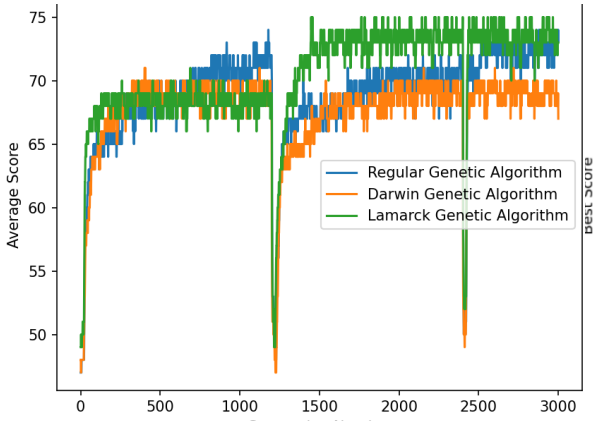
נשים לב כי כאן, **בגרף ה best (הימני)** האלגוריתם הדרוואני והלאמארקי התכנסו ביחד(הדרוואני 20 דורות קודם)ופתרו את הלוח **לפני** האלגוריתם הגנטי הרגיל (שאחרי כ 750 דורות הגיע לפתרון). כלומר בלוחות קשים האלגוריתם הגנטי הרגיל חלש כיוון שאין בו לוגיקה מסובכת שתעזור לו (כיוון שהוא רנדומי לחלוטין ברובו). בסה"כ כלל האלגוריתמים הגיעו לתוצאה יפה והצליחו לפתור את הלוח תוך מספר קטן של דורות. ניתן לראות **לפי הגרף של הממוצע (השמאלי)** כי לאורך כל הדרך האלגוריתם הלאמארקי מתכנס הכי מהר ומגיע לציון הגבוה ביותר. בנוסף, תחילה האלגוריתם הגנטי הרגיל מחזיק בציון גבוה יותר (בממוצע) מאשר הדרוואני, אך בהמשך הדרוואני משתפר ועוקף את האלגוריתם הגנטי הרגיל (בממוצע). ניתן לראות בגרף של הממוצע שכל 1200 דורות יש קפיצה וזאת בעקבות הלוגיקה שלנו שמנסה לעזור ולאפשר פתח לפתרונות חדשים (בכך נמנעים ממקסימום לוקאלי ומאפשרים יותר רנדומיות) וזאת ע"י הגברת אחוז המוטציות, cross-over והכנסה של פתרונות חדשים.

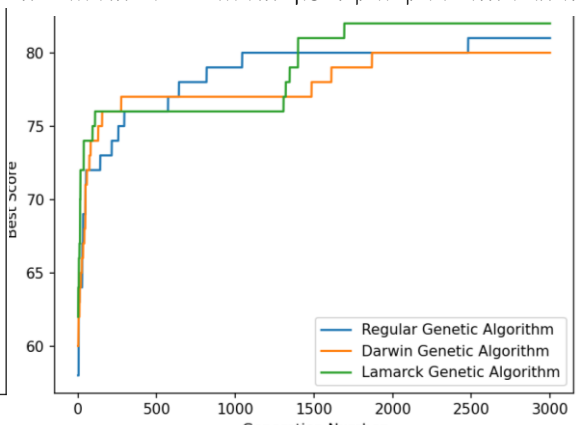




רמת קושי easy - לוח 6*6 :

אפשר לראות **שבגרף ה best (הימני)** רק האלגוריתם הגנטי הרגיל הצליח לפתור את הלוח (תוך כ 1720 דורות). למרות זאת האלגוריתם הלאמארקי הגיע לציון גבוה יותר מהאלגוריתמים האחרים ב1300 הדורות הראשונים (הוא השתפר בצורה מהירה יותר מהיתר). נשים לב כי האלגוריתם הדרוואני היה חלש בלוח זה והגיע לציון הכי נמוך בסוף 3000 דורות. **בגרף של הממוצע (השמאלי)** ב1200 הדורות הראשונים האלגוריתם הלאמארקי הכי טוב בממוצע (מתכנס הכי מהר ובעל הציון הכי גבוה), ואחרי 1200 דורות האלגוריתם הגנטי הרגיל הכי טוב, לאחריו הלאמארקי ורק לבסוף הדרוואני. שוב הדרוואני הכי פחות טוב בממוצע.



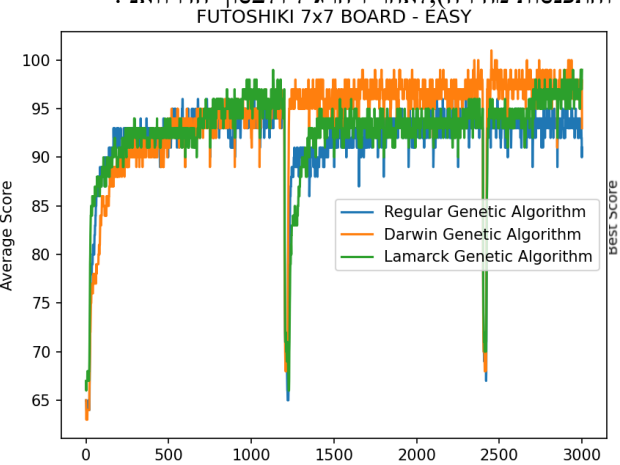


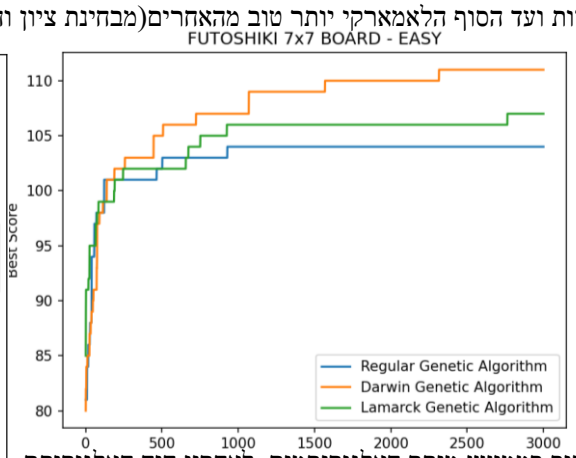
רמת קושי tricky-לוח 6*6:

בגרף ה best (הימני) רק האלגוריתם הלאמארקי הצליח לפתור את הלוח (במסגרת הריצה) בנוסף לאלגוריתם הלאמארקי לקח יותר זמן להשתפר מלאחרים. בסה"כ האלגוריתם הרגיל והדרוואני הגיעו לתוצאות

בס"ד

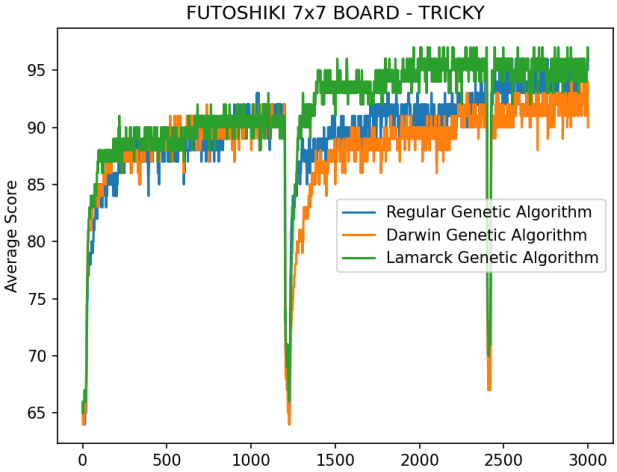
דומות מבחינת ציון.**לפי הגרף של הממוצע (השמאלי)** ב 1200 הדורות הראשונים האלגוריתם הגנטי הרגיל היה טוב יותר בממוצע מהאחרים (מבחינת ציון), ואחרי 1200 דורות ועד הסוף הלאמארקי יותר טוב מהאחרים(מבחינת ציון והתכנסות מהירה),לאחריו הרגיל ולבסוף הדרוואני.

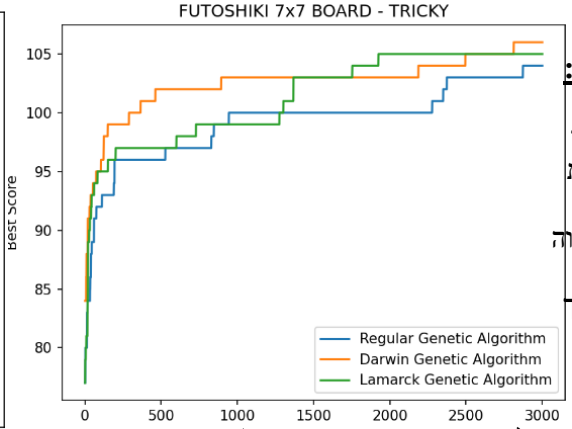




רמת קושי easy - לוח 7*7:

בגרף ה best (הימני) רק האלגוריתם הדרוואני הצליח לפתור את הלוח (במסגרת ההרצה) כמו כן שוב האלגוריתם הלאמארקי היה טוב יותר מהאלגוריתם הגנטי הרגיל. **לפי הגרף של הממוצע (השמאלי)** ב 1200 הדורות הראשונים כלל האלגוריתם היו דומים. אחרי 1200 דורות האלגוריתם הדרוואני היה יותר טוב בממוצע מיתר האלגוריתמים, לאחריו היה האלגוריתם הלאמארקי ולבסוף האלגוריתם הגנטי הרגיל ובכך הממוצע מתיישב עם הגרף של ה best.





רמת קושי tricky-לוח 7*7:

בגרף ה best (הימני) כל האלגוריתמים לא הצליחו לפתור את הלוח במסגרת ה3000 דורות שהקצנו). כמו כן האלגוריתם הדרוואני הגיע לתוצאה ההכי גבוה בתום ההרצה, וקצת מתחתיו נמצא האלגוריתם הלאמארקי, כך שהאלגוריתם הגנטי הרגיל הכי פחות טוב (כלומר חלש ביחס לשאר) שכן אין בו לוגיקה מסובכת שתעזור לו (כאן הוא לא מצליח כיוון שהוא רנדומי ברובו).

לפי הגרף של הממוצע (השמאלי) ב 1200 הדורות הראשונים האלגוריתם היו דומים ולאחר מכן האלגוריתם הלאמארקי היה יותר טוב בממוצע מהשאר, לאחריו האלגוריתם הגנטי הרגיל ולבסוף האלגוריתם הדרוואני. בניגוד לגרף של ה best , לפי ממוצע הציונים הלאמארקי מתכנס יותר מהר ומגיע לציון גבוה יותר. כמו כן, האלגוריתם הגנטי בממוצע נראה יותר טוב מאשר הדרוואני.

מסקנות:

לאור הניתוח שערכנו למעלה ניתן לראות בברור שהיחס בין ביצועי האלגורתמים השונים (כלומר הרגיל, הדארוויני והלאמארקי) משתנה בהתאם לגדלי הלוח ורמות הקושי השונות. **כך שעבור לוחות גדולים יותר ומסוג קשה יותר (tricky) האלגוריתמים אשר ביצעו את האופטימיזציות (הדרוואני והלאמארקי) הגיעו לתוצאות טובות יותר והתכנסו מהר יותר, ואילו האלגוריתם הגנטי הרגיל הצליח בעיקר בלוחות הקטנים והקלים יותר** (שכן הוא מתבסס על רנדומיות ברובו ולכן מצליח להתכנס לפתרון באופן זהה/ טוב יותר ומהיר יותר).

באופן כללי, ניתן לראות כי **בממוצע האלגוריתם הלאמארקי מוביל בפער מבחינת ציון וקצב ההתכנסות על פני יתר האלגוריתמים,** ואילו האלגוריתם הדרוואני לעיתים היה החלש ביותר בממוצע אך התכנס לעיתים מהר יותר מהאלגוריתם הגנטי הרגיל.

נשים לב שכיוון שמדובר באלגוריתמים גנטיים, מבוססי הסתברות ורנדומיות, לא ניתן לחתוך מסקנות בצורה גורפת ועקבית שכן היו גם הרצות לעיתים שסתרו הרצות אחרות אך כן השתדלנו באופן כללי לחקור את המקרה הכללי שחוזר לרוב. כמו כן, לא בכל ההרצות האלגוריתמים הנ"ל מצאו פתרון ולעיתים נדרשו מספר רב של הרצות/ הרצה למספר רב יותר של דורות עד לכדי התכנסות לפתרון הנכון ולכן יש לזכור זאת כאשר נריץ את התוכנית ולשנות בהתאם את כמות הדורות שרוצים שתהיה בהרצה.

באופן כללי שמנו לב לתופעה מעניינת ומופלאה – עבור לוחות קלים ממש עדיף כמה שפחות להתערב כלומר מימוש שמתבסס בעיקרו על רנדומיות הביא לתוצאות הטובות ביותר, וככל שננסה להתערב עם לוגיקות מסובכות כך האלגוריתם ייחלש ויביא לתוצאות פחות טובות.