ביולוגיה חישובית – תרגיל 2 אלגוריתם גנטי

הרצת הסימולציה:

ראשית, התוכנה נכתבה בPYTHON, יש להתקין את הספריות שאני עושה להם IMPORT הספריות הם: matplotlib, copy, random, numpy.

חשוב: יש שלושה אלגוריתמים שיכולים לרוץ בסימולציה זו, להלן הסבר כיצד להריץ כל אחד:

<u>אלגוריתם גנטי רגיל:</u> על מנת להריץ את האלגוריתם הגנטי הרגיל (חלק א' של המטלה), יש לגשת לפונקציית ()mutate ולוודא שאנו מחזירים את: return change_random_number(board) כאשר שאר הפונקציות צריכות להיות בהערה. שימו לב, בפונקציית ()main יש לוודא ש: is_darwinian = **False**

<u>האלגוריתם הגנטי הלאמארקי:</u> על מנת להריץ אל האלגוריתם לפי לאמרקית (חלק ב' של המטלה), יש לגשת לפונקציית ()mutate ולוודא שאנו מחזירים את: return lamarki_mutate(board, restrictions) כאשר שאר הפונקציות צריכות להיות בהערה. שימו לב, בפונקציית ()main יש לוודא ש: is_darwinian = **False**

<u>האלגוריתם הגנטי הדארוויני:</u> על מנת להריץ את האלגוריתם הדארוויני, יש לגשת לפונקציית ()mutate ולוודא שאנו מחזירים את: return lamarki_mutate(board, restrictions) כאשר שאר הפונקציות צריכות להיות בהערה. שימו לב, בפונקציית ()main יש לוודא ש: is_darwinian = **True**

לאחר שווידאתם שכל התנאים הנ"ל מתקיימים עבור כל אלגוריתם רצוי, יש להריץ את הmain והתוכנה תפעל. כעת, בתחילת התוכנית יש בהערה "PARAMETERS" אלו הפרמטרים שמגדירים את התוכנית, אפרט עליהם בקצרה:

שם הפרמטר	תפקיד	ערך התחלתי
GENERATION_SIZE	גודל כל מחזור של פתרונות	100
DUPLICATION_SCORE	רלוונטי לפונקציה calculate_fitness, מספר הנקודות שאני מוסיפים לציון של לוח כאשר נספרת בו טעות	10
GREATER_THAN_RESTRICTION_SCORE	רלוונטי לפונקציה calculate_fitness, מספר הנקודות שאני מוסיפים לציון של לוח כאשר נספרת בו טעות של אי שוויון שלא מתקיים	10
SIZE	משתנה גלובלי שמגדיר את מספר העמודות והשורות של הלוח, הערך ההתחלתי חסר משמעות מכיוון שאנו נקבל זאת מקובץ הקלט לכן אין מה לשנות פרמטר זה.	0
MAX_GENERATIONS	מספר האיטרציות של האלגוריתם ירוץ	100
PRECENTAGE_OF_REPLICATION	אחוז הלוחות שיעברו העתקה למחזור הבא של הלוחות	0.35

חלק א' - אלגוריתם גנטי רגיל

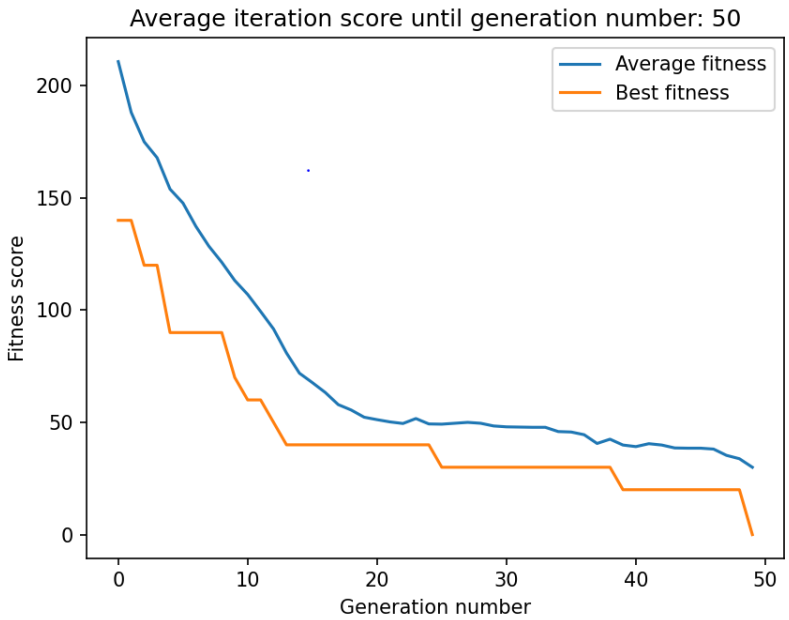
התחלתי עם מחזור ראשון של פתרונות של 100 לוחות, כאשר את כולם מילאתי באופן רנדומלי לחלוטין (על גבי תנאי ההתחלה של הקלט הנתון). כעת, בניתי פונקציית הערכה calculate_fitness שמטרתה הינה להיות מציין איכות עבור לוח. כלומר, עבור כל אחד מהלוחות אני בודק את הציון שלו לפי calculate_fitness. זוהי בעצם פונקציה שמקבלת לוח ומחזירה לי את הציון שלו. ככל שיש יותר טעויות בלוח (למשל שלא מכבדים אי שוויון או שספרה מופיע יותר מפעם אחת בעמודה \ שורה) אזי ציון הלוח עולה ב 10 נק'. כמובן שככל שציון הלוח נמוך יותר, כך הפתרון טוב יותר כאשר לוח עם ציון 0 הינו לוח עם פתרון תקין לחלוטין שמקיים את כלל האילוצים. אני עוצר את האלגוריתם כאשר אני מקבל לוח שציונו 0 או כאשר הגעתי למספר האיטרציות המקסימאלי שהגדרתי לתוכנה מראש (MAX_GENERATIONS).

מימוש האלגוריתם הגנטי הרגיל (ביצד נבחרים הדורות): כאמור, ראשית הגרלתי מחזור ראשון של לוחות מלאים במספרים רנדומליים לחלוטין. זוהי תהיה האוכלוסייה הראשונית שלי. כעת, בדקתי מה הציון של כל הלוחות במחזור הראשון שלי, ומיינתי אותם מהציון הנמוך ביותר לגבוהה ביותר (יש לזכור שבכל שהציון יותר נמוך כך הפתרון יותר טוב, כי יש בו פחות תקלות). כעת, העברתי למחזור הבא את 35% הלוחות עם הציון הטוב ביותר (תהליך REPLICATE). כעת נותר לי למלא את שאר ה65% לוחות של המחזור הבא. 65% אלו יגיעו מתהליך CROSSOVER, זהו תהליך שבו אני לוקח אבא אחד רנדומלי ואמא אחת רנדומלית (לא יכולים להיות אותו לוח) מ35% הלוחות הכי טובים ויוצר להם לוח ילד. הילד נוצר עי" פונקציית create_child_board זוהו פונקציה שמקבל שני לוחות (אבא ואמא), מגרילה מספר שורה, ואז מעתיקה את כל האבא עד השורה שהוגרלה ואת כל האמא מהשורה שהוגרלה והלאה. כך נוצר לנו לוח ילד שהוא חלק מהאבא וחלק מהאמא באופן רנדומלי. כעת יצרנו מחזור חדש של לוחות אבל עוד לא סיימנו, נותר לנו לעשות מוטציה ללוחות. הלוחות ילדים שנוצרו מתהליך CROSSOVER יעברו מוטציה. ישנם שני סוגים של מוטציות שאיתם ניסיתי להגיע לפתרון. המוטציה הראשונה הינה מוטציה יחסית פשוטה, נבחר איבר אחד רנדומלי בכל לוח של כל ילד שנוצר, ונשנה את הערך של האיבר שנבחר לערך רנדומלי ע"י פונקציית change_random_number(board). המוטציה השנייה קצת יותר מורכבת, זוהי מוטציה שבוחרת שני מספרי שורה רנדומליים שונים בתוך לוח, ופשוט מחליפה ביניהם ע"H פונקציית shuffle_rows(board). בהמשך אשווה בין ההבדלים של שני המוטציות כי הם אכן מעניינים ומשפיעים מאוד על האלגוריתם, אבל לבסוף החלטתי שבאלגוריתם הרגיל אשאר עם המוטציה הראשונה שהסברתי מכיוון שהיא הגיעה לפתרון יותר מהר ויותר טוב בכל אלפי ריצות שהרצתי. כעת סיימנו בעצם ליצור את המחזור השני שלנו. נשתמש בפונקציית calculate_fitness על מנת לבדוק את הציונים של כל המחזור, נמיין את המחזור ונחזור על כלל האלגוריתם שוב ושוב עד שנגיע לאחד משני תנאי עצירה: נגיע למחזור שיש בו לוח עם ציון 0 (בלומר אין בו טעויות והוא תקין לגמרי) או שנגיע לכמות מקסימלית של מחזורים שהגדרנו מלכתחילה.

ייצוג הפתרונות: מכיוון שהאלגוריתם הגנטי מסתמך המון על רנדומליות, נשים לב שהאלגוריתם לא תמיד יגיעו לפתרון מושלם (עם ציון fitness של 0). אבל, אכן בחלק רק מהפעמים האלגוריתם אכן הגיע לציון 0 והחזיר לוח תקין. להלן דוגמאות של פתרונות:

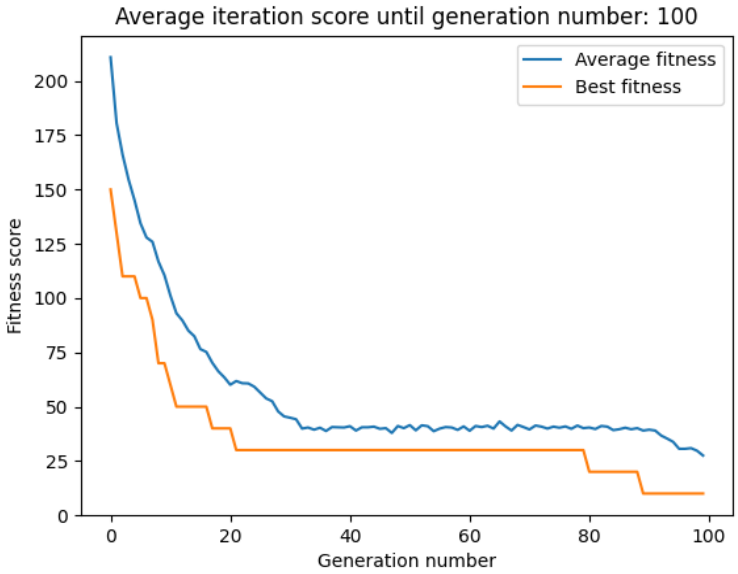
<u>דוגמא א':</u>

```
Best board:
2>1 5 3 4
      >
3 5>4 1 2

4 3 2 5 1
      > <
5 2 1 4 3
      > <
1<4 3 2 5

Best score: 0, average score: 30.0
```



דוגמא ב':

```
Best board:
5>4 1 3 2
      >
3 2>4 1 5

4 3 2 5 1
      > <
2 1 5 4 3
      > <
1<5 3 2 4

Best score: 10, average score: 27.9
```



הסבר: ניתן לראות הבדל בין שני הפתרונות הנראים למעלה. בדוגמא א' הגרף מייצג ריצה שהגיעה לפתרון מושלם תוך בדיוק 50 ריצות והחזירה לנו לוח ללא טעויות כלל (הלוח מודפס משמאל לגרף). ניתן לראות שבמחזור מספר 50 ממוצע הפתרונות היו בדיוק 30 (ניתן לראות בתמונה השמאלית למטה את הממוצע של המחזור האחרון) כלומר בממוצע לכל לוח במחזור 50 היה בערך 3 טעויות, אך מכיוון שהיה פתרון אחד בודד שהיה מושלם יכולנו לעצור את האלגוריתם כי הגענו לפתרון. בדוגמא ב' קיבלנו דוגמא לריצה שבה הגענו לפתרון ממש טוב (עם טעות אחת בלבד) תוך 100 מחזורים ואז עצרנו (כי זה מספר המחזורים המקסימאלי שהגדרתי). ניתן לראות שלאחר 100 מחזורים ממוצע הציונים הגיע ל27.9 כלומר במחזור האחרון לכל לוח היה בממוצע בין 2-3 טעויות, כאשר בפתרון הכי טוב היית בדיוק טעות אחת. הממצאים האלה מרתקים, מכיוון שניתן לראות באופן ברור שהאלגוריתם הגנטי עובד נהדר, אני מסמך על שינויים רנדומליים בכל מחזור, וככל שעוברים יותר מחזורים, ממוצע הציונים יורד (כלומר בכל לוח יש פחות טעויות) ואכן בסוף האלגוריתם אני מגיע לפתרון. כמובן שניתן להריץ את האלגוריתם כמה פעמים שנרצה, לפעמים נקבל פתרונות מושלמים ולפעמים נקבל פתרונות כמעט מושלמים, זה היופי באלגוריתם גנטי שמסתמך על המון תהליכים רנדומליים בשביל לפתור את הבעיה, אבל בכל ריצה ללא יוצא מן הכלל, תמיד נראה שציון הממוצע יירד באופן משמעותי לאורך הדורות, וזה סימן שהאלגוריתם עובד ושיש שיפור ממחזור למחזור.

פונקציית הערכה: יצרתי פונקציית הערכה שמטרתה לקבל לוח ולהחזיר ציון לאותו לוח, הפונקציה נקראת calculate_fitness והיא עובדת באופן פשוט יחסית. הפונקציה מקבלת לוח וסופרת את מספר הטעויות שבתוך הלוח, כל טעות מהווה 10 נקודות. כלומר, אם אותו מספר מופיע יותר מפעם אחת בשורה \ עמודה או שלא מכבדים אי שוויון בתוך הלוח, הציון של הלוח עולה ב10 נק'.

ביצד מבצעים הכלאה (CORSSOVER): ראשית בחרתי אבא ואמא מתאימים מתוך הורים פוטנציאלים במחזור (ההורים חייבים להיות חלק מה35% עם הציון הכי טוב, ווידאתי שאותו הורה לא יכול להיבחר פעמיים כלומר להיות גם האבא וגם האמא בו זמנית). כעת, בעזרת פונקציה create_child_board(mother_board, father_board) יצרתי להורים ילד. זוהי פונקציה שמקבלת שני לוחות (אבא ואמא) ומחזירה את הילד שלהם באופן הבא: אני מגריל מספר שורה רנדומלי, ואז מעתיק את הלוח של האבא עד מספר השורה שהגרלתי, ואת שארית הלוח מהלוח של האמא. כך קיבלתי לוח חדש שהוא חלק מהאבא וחלק מהאמא באופן רנדומלי וזה בעצם יהיה הילד שנוצר שיתווסף למחזור הבא.

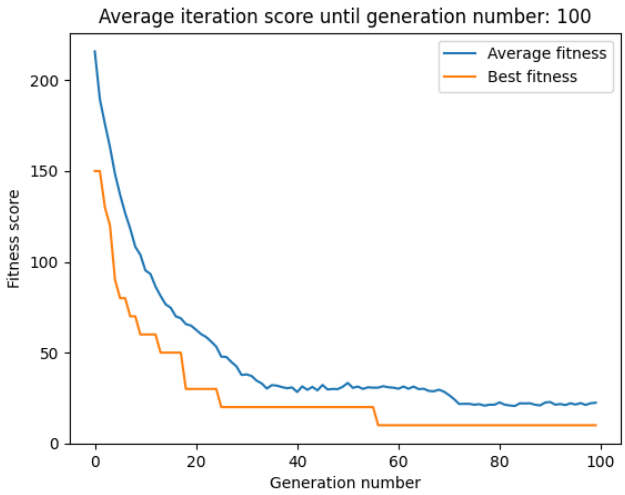
ביצד מבצעים מוטציה: ראשית רציתי לבחון כיצד מוטציות שונות משפיעות על האלגוריתם, ולאחר שאשווה בין הפתרונות אוכל לבחור המוטציה שהכי תתאים לאלגוריתם שלי מבחינת יעילות ואיכות פתרון. לכן, יצרתי שני סוגי מוטציות:

מוטציה א': מוטציה זו תקבל ילד שנוצר מCROSSOVER, תבחר איבר רנדומלי בתוך הלוח של הילד ותשנה את הערך שנבחר לערך רנדומלי אחר. מוטציה זו מופעלת ע"י פונקציית mutate כאשר מחזירים change_random_number(board)

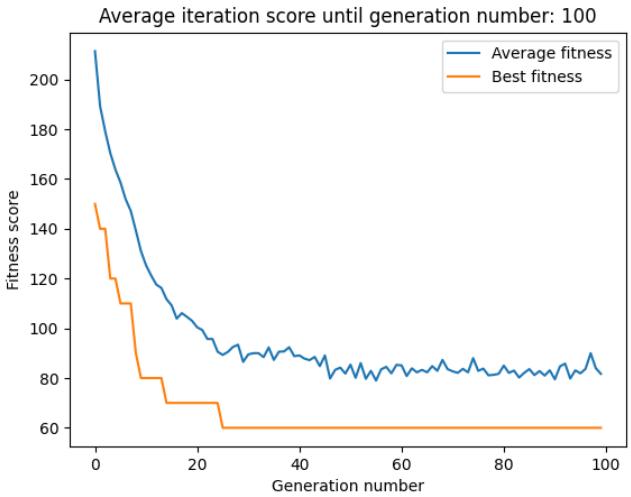
מוטציה ב': מוטציה זו תקבל ילד שנוצר מCROSSOVER, תבחר שני שורות רנדומליות בתוך הלוח של הילד ותחליף ביניהם. . מוטציה זו מופעלת ע"י פונקציית mutate כאשר מחזירים shuffle_rows(board)

להלן הממצאים של שני סוגי המוטציות:

תוצאת מוטציה א':



תוצאת מוטציה ב':



הסבר: ניתן לראות באופן ברור, שמוטציה א' (הגרף הימני) מצליחה להגיע לפתרונות יותר טובים של הלוחות מאשר מוטציה ב' (הגרף השמאלי). נשים לב, שלאחר 100 דורות, מוטציה א' הגיע לממוצע של 29, ולפתרון הכי טוב בציון 10 (בעל טעות אחת), לעומת מוטציה ב' שהגיעה לממוצע של 83 ולפתרון הכי טוב בציון 60 (בעל 6 טעויות). זה אכן נתון מרתק שאומר לנו המון על המשמעות של מוטציות ועל אופי ההשפעה שלהם על האלגוריתם. כלומר, מוטציות שונות יכולות להשפיע באופן שונה על שיפור הדורות באלגוריתם. כאן ניתן לראות בבירור, שהחלפת שורות שלמות בלוחות הרבה פחות יעיל משינוי איבר בודד רנדומלי בלוח. אני מאמין שזה קרה כי כאשר אני מחליף בין שני שורות שלמות, אני בעצם משנה 2n איברים במטריצה, כאשר חלקם יכולים להיות טובים וחלקם לא, ויש סבירות גבוהה שכאשר אני משנה יותר מדי איברים בפעם אחת אני בעצם מתערב יותר מדי באלגוריתם, ומעט את תהליך ההשתפרות שלו. ולכן, יכול להיות שאמנם תיקנתי כמה טעויות, אבל יש גם סבירות גבוהה שהוספתי כמה טעויות במקביל וזה בעצם פוגע ביעילות האלגוריתם שלי. לכן, לטובת תרגיל זה, החלטתי שבאלגוריתם שלי אעבוד רק עם מוטציה א' עקב היעילות שלה על גבי מוטציה ב'.

התמודדות עם בעיית ההתכנסות המוקדמת: על מנת למנוע התכנסות מוקדמת ווידאתי שהמוטציות שהגדרתי באלגוריתם עובדות באופן רנדומלי לחלוטין. לכן, אם נשנה את מספר הריצות המקסימליות ליותר גבוהה, תמיד נגיע לפתרון מתישהו, אבל מכיוון שהמוטציה מתבצעת באופן רנדומלי לגמרי זה יכול לקחת לפעמים קצת זמן ולפעמים יותר זמן. לכן האלגוריתם שיצרתי לפעמים יגיע לפתרון מושלם, אך לפעמים יחזיר לנו פתרון כמעט מושלם עקב התכנסות מוקדמת. לשם כך, נוכל לתת לאלגוריתם להמשיך לרוץ עד שיחזיר לנו פתרון מושלם (זה יקרה מתישהו כי בשלב מסוים האיבר שעושה את הבעיה ייבחר באופן רנדומלי) אך זה יכול לקחת זמן.

לכמה דורות להריץ את האלגוריתם: לאחר טסטים רבים של האלגוריתם, ולאחר בדיקה של מספר דורות מקסימאלי של 50, 100, 200, 500, 1000, 10000 שמתי לב שלאחר 100 דורות, האלגוריתם לרוב מביא פתרון מושלם או כמעט מושלם (עם טעות אחת). אם עד 100 דורות האלגוריתם לא החזיר פתרון מושלם, כנראה שהוא בבעיית ההתכנסות המוקדמת, אשר תקבל מענה עקב המוטציה הרנדומלית שיצרנו, אך יכול להיות שזה ייקח מספר רב של דורות עד שנגיע לפתרון מושלם. לכן בשביל למנוע זמני ריצה ארוכים בשביל לפתור טעות קטנה אחת באיבר אחד בלוח שלם, הגדרתי שמספר הריצות המקסימאלי הינו 100. זה כמובן בר שינוי בפרמטרים שפורטו בעמוד הראשון של הדו"ח.

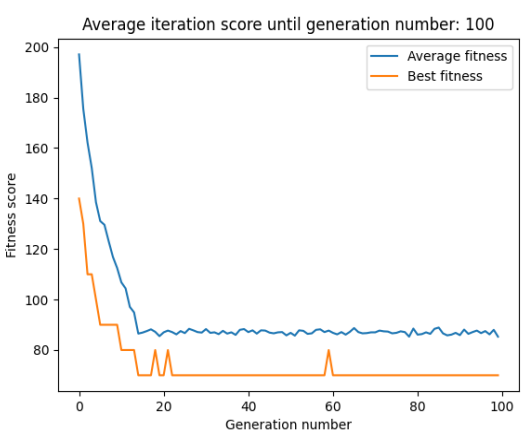
חלק ב' - אלגוריתם גנטי לאמארקי ודארוויני לעומת אלגוריתם גנטי רגיל

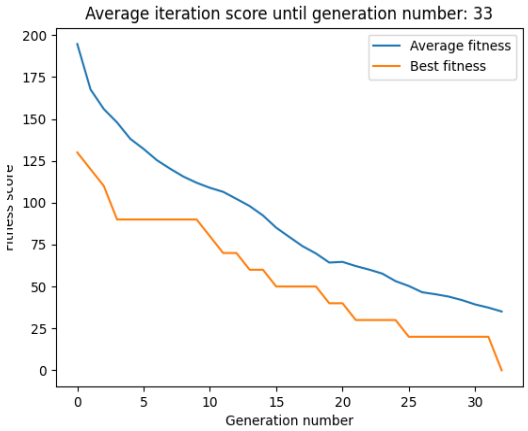
מימוש האלגוריתם הגנטי הלאמארקי: אלגוריתם זה דומה מאוד לאלגוריתם הגנטי הרגיל, רק שכאן אבצע אופטימיזציה לכל דור ורק לאחר מכן אחשב את הציון הfitness של כל לוח בדור. בעצם זה יאפשר לי לבצע שיפור לכל פתרון וכך אעביר פתרון משופר לדור הבא. באלגוריתם לאמארקי החלטתי שהמספר הקבוע של צעדי האופטימיזציה שאני מבצע על כל לוח הוא כמספר האי שוויוניים שמוזנים לי ללוח (מספר שנשאר קבוע לאורך כל הסימולציה לכלל הדורות). ראשית, בדור הראשון שוב אמלא 100 לוחות באופן רנדומלי לחלוטין, אז אבחר את ה35% הטובים מבניהם לדור הבא. עכשיו, שוב אבצע CROSSOVER ואצור ילדים מה35% הלוחות הכי טובים, רק שהפעם כל ילד יקבל מוטציה משופרת. המוטציה עובדת באופן הבא: עבור כל ילד שאצור, אעבור על כלל האי שוויוניים שהוזנו לי ללוחות ואבדוק האם הם מתקיימים באופן תקין או לא אצל הילד, אם הם לא מתקיימים אבצע חילוף בין שני האיברים כך שהאי שוויון יתקיים. אם אין בעיה של אי

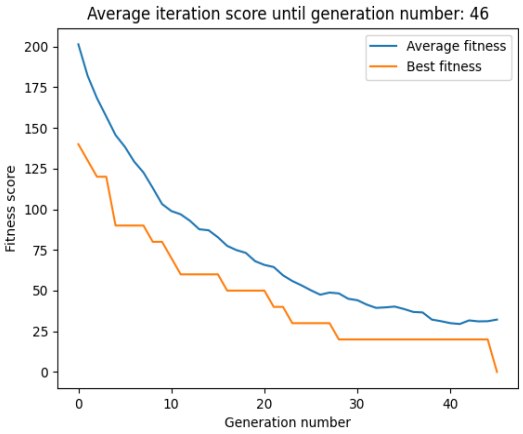
שוויון אז כנראה שיש מספר שמופיע יותר מפעם אחת בשורה \ עמודה כלשהי, ולכן אבצע מוטציה רגילה כמו באלגוריתם הגנטי הרגיל אם זה המצב (אשנה איבר רנדומלי בלוח). לסיכום, האלגוריתם הלאמארקי הוא בעצם אלגוריתם גנטי משופר, המוטציה אצלו עוברת על כלל האי שוויונים ומוודא שהלוח בנוי כך שכל האי שוויונים מתקיימים באופן תקין.

מימוש האלגוריתם הגנטי הדארוויני: באלגוריתם זה כל לוח עובר אופטימיזציה, הfitness שלו נקבע אחרי האופטימיזציה, אבל הדור הבא נוצר על סמך הפתרון המקורי לפני האופטימיזציה.

כעת נבחן את שלושת האלגוריתמים על גדלים שונים של לוחות(5*5, 6*6, 7*7) עבור 2 דרגות קושי שונות (קל \ קשה).נתחיל 5*5 קל. מימין תוצאות אלגוריתם גנטי, באמצע אלגוריתם לאמארקי, משמאל אלגוריתם דארוויני:





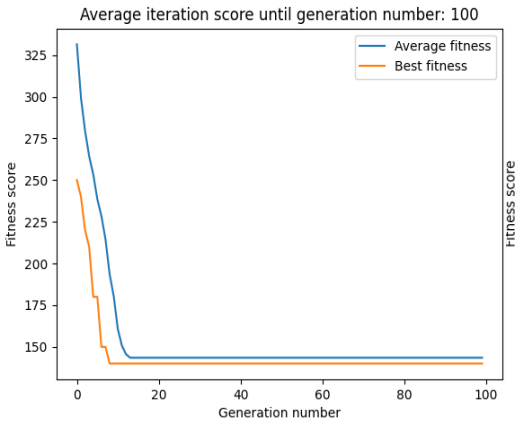


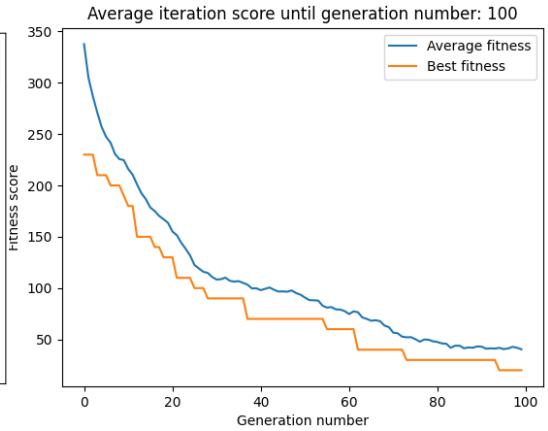

```
Best board:
5 3 1 4 2
      >
4 1<2<3 5

2 4 5>1 3
>       <
1 2 3 5 4

3 5 4 2 1

Best score: 0, average score: 32.2
```

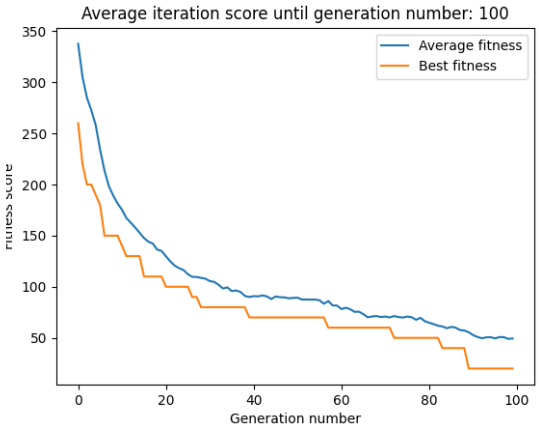
ממצאים: נשים לב, שהאלגוריתם הגנטי שלנו אכן מצא פתרון מושלם לאחר 46 דורות עם ממוצע ציונים של 32. להלן הפתרון:

כמו כן, נשים לב שהאלגוריתם הלאמארקי מצא פתרון מושלם אפילו בזמן יותר יעיל, לאחר 33 דורות בלבד עם ממוצע ציונים של 30. זה תוצאה הגיונית מכיוון שהאלגוריתם הלאמארקי מבצע אופטימיזציה לכל פתרון ובעצם שיפר את המוטציה שהאלגוריתם הגנטי הרגיל עשה. מנגד, אפשר לראות שגם לאחר 100 דורות, האלגוריתם הגנטי הדארוויני לא הגיע לפתרון. כאשר הממוצע שלו עומד על 88 והציון הכי טוב שלו הוא כמעט 70. ניתן להסיק מכאן שעצם העובדה שהדור הבא באלגוריתם הדארוויני נוצר על סמך הפתרון המקורי לפני האופטימיזציה, זה משפיע לרעה על יעילות האלגוריתם, ומכאן שהאלגוריתם הדארוויני מתברר להיות האלגוריתם הפחות יעיל עבור לוחות 5*5 ברמת קושי קלה.

כעת נעבור ללוח 6*6 עם רמת קושי קשה. מימין מוצג אלגוריתם גנטי רגיל, באמצע לאמארקי ומצד שמאל דארוויני:

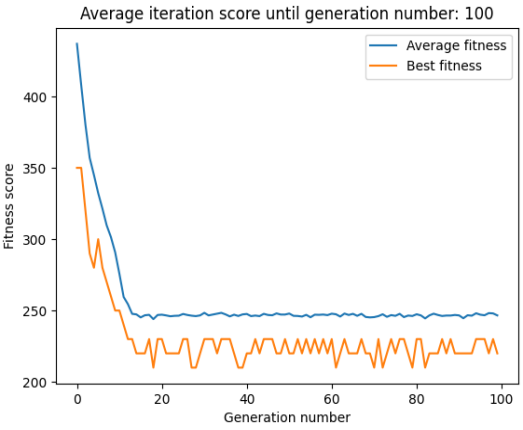


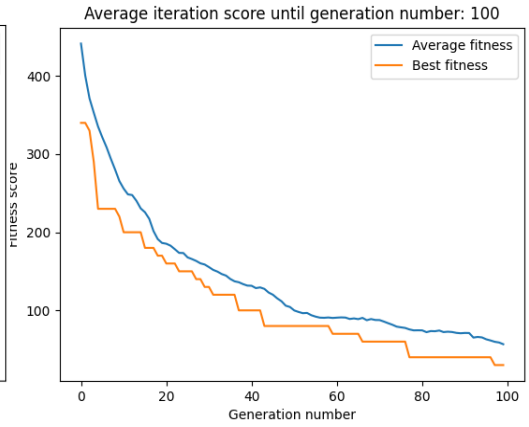


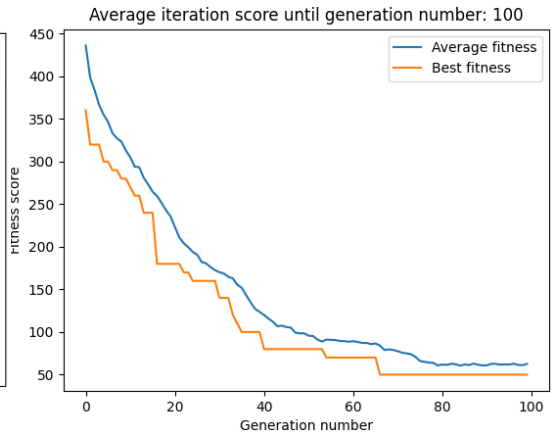


ממצאים: נשים לב, אף אחד מהאלגוריתמים לא מצא פתרון מושלם. האלגוריתם הגנטי הרגיל והאלגוריתם הלאמארקי מצאו פתרון עם ציון 20 (כלומר בעל 2 טעויות) לאחר 100 דורות. הממוצע באלגוריתם הגנטי הרגיל היה 53 והממוצע באלגוריתם הגנטי הלאמארקי היה 46. האלגוריתם הדארוויני גם כן לא הגיע לפתרון מושלם, כאשר הפתרון הכי טוב שהחזיר היה בעל ציון של 140. סה"כ שוב קיבלנו שהאלגוריתם הלאמארקי אמנם לא נתן פתרון מושלם, אך החזיר בממוצע את הפתרונות הכי טובים בזמן שהאלגוריתם הדארוויני החזיר את הדורות עם הפתרונות הכי גרועים.

כעת נעבור ללוח 7*7 עם רמת קושי קשה. שוב, מימין מוצג אלגוריתם גנטי רגיל, באמצע לאמארקי ומצד שמאל דארוויני:







ממצאים: נשים לב, גם כאן אף אחד מהאלגוריתמים לא מצא פתרון מושלם. האלגוריתם הגנטי הרגיל מצא פתרון עם ציון 50 (5 טעויות) והאלגוריתם הלאמארקי מצאו פתרון עם ציון 30 (3 טעויות) לאחר 100 דורות. הממוצע באלגוריתם הגנטי הרגיל היה 68 והממוצע באלגוריתם הגנטי הלאמארקי היה 55. האלגוריתם הדארוויני גם כן לא הגיע לפתרון מושלם, כאשר הפתרון הכי טוב שהחזיר היה בעל ציון של 230 עם ממוצע פתרונות של 247. נשים לב, שבגרף של האלגוריתם הדארוויני (השמאלי) ציון הfitness הכי טוב עולה ויורד לאורך הדורות. זה בעצם אומר לנו שבאלגוריתם זה, כשאר הלוח ברמת קושי גבוהה ובגודל 7*7, לא בהכרח נגיע לפתרון יותר טוב אם נרוץ לאורך יותר זמן. אני מעריך שזה נובע בגלל שבאלגוריתם זה אני יוצר דור חדש לפני האופטימיזציה, ולכן כאשר המורכבות בלוח גדלה, התהליך הזה לא מספיק חזק ויעיל בשביל להגיע לפתרון מושלם. סה"כ שוב קיבלנו שהאלגוריתם הלאמארקי אמנם לא נתן פתרון מושלם, אך החזיר בממוצע את הפתרונות הכי טובים בזמן שהאלגוריתם הדארוויני החזיר את הדורות עם הפתרונות הכי גרועים.

סיכום

יש לציין שהרצתי את הסימולציה גם בלוח 5*5 עם רמת קושי קשה, 6*6 עם רמת קושי קל ובלוח 7*7 עם רמת קושי קל וגם בריצות אלו האלגוריתם הגנטי הלאמארקי תמיד היה במקום הראשון, אחריו האלגוריתם הגנטי הרגיל ולבסוף האלגוריתם הגנטי הדארוויני. ניתן להסיק מכך שהאופטימיזציה שנעשית בלאמארק משפרת מאוד את היעילות של האלגוריתם הגנטי, ובעצם מגיע להישגים טובים יותר משאר האלגוריתמים. בנוסף, עצם העובדה שבאלגוריתם הדארוויני הדור הבא נוצר על פי הפתרון המקורי לפני האופטימיזציה מאטה את כל תהליך ההשתפרות. בעצם, אם הייתי מגדיל את מספר הדורות המקסימאלי מ-100 ל-1000 אכן הייתי מקבל פתרונות יותר טובים לאורך זמן של יותר דורות, אך זה היה עולה לי בזמני ריצה יותר ארוכים, ובנוסף השיפור לא היה משמעותי עקב בעיית ההתכנסות המוקדמת (שאכן טיפלנו בה, אך בגלל שהמוטציות רנדומליות לוקח לפעמים מספר רב של דורות עד שרואים שיפור משמעותי בממוצע הציונים).