בס"ד

דו"ח תרגיל 2 – ביולוגיה חישובית

גיא עדני – 208642884
חן ברסנו – 319004339

איך להריץ

מצורף קובץ בשם ex2.exe, על מנת להריץ אותו יש לפתוח cmd בתיקייה בו הוא נמצא ולכתוב ex2.exe.
לתכנית ישנם 3 מצבים:

- Regular Mode – אלגוריתם גנטי רגיל. ניתן להריץ ללא ארגומנטים או בתוספת אחד מהדגלים:
  -r, -R, -regular, -Regular
- Darwin Mode – האלגוריתם הגנטי בתוספת האופטימיזציה של דרווין. ניתן להריץ בתוספת אחד מהדגלים:
  -d, -D, -darwin, -Darwin
- Lamarck Mode – אלגוריתם גנטי בתוספת האופטימיזציה של למארק. ניתן להריץ בתוספת אחד מהדגלים:
  -l, -L, -lamarch, -Lamarck

למשל הרצה של האלגוריתם במצב Lamarck תיראה כך: ex2.exe -L

מיד לאחר מכן מתחיל האלגוריתם לעבוד. לאורך ריצת התוכנית מודפס עבור כל דור ה-fitness הכי טוב עד שבשלב מסוים ה-fitness לא משתנה והתוכנית עוצרת. הפלט הוא 3 קבצים: perm.txt, plain.txt ותמונה של גרף שמראה את ציון ה-fitness כפונקציה של מספר הדורות. (fitness graph.png)

רקע

בתרגיל זה התבקשנו לפענח צופן מונו אלפביתי בעזרת אלגוריתם גנטי. צופן מונו אלפביתי הוא צופן שבנוי מהחלפה של אות באות אחרת. לצורך המטלה קיבלנו 4 קבצים, קובץ מוצפן ו-3 קבצי עזר:

1. enc.txt – קובץ טקסט של מילים שעברו הצפנה, אותו יש לפענח.
2. dict.txt – "מילון" של מילים שכיחות באנגלית.
3. Letter_Freq.txt – קובץ שמכיל שכיחות של אותיות באלף-בית באנגלית.
4. Letter2_Freq.txt – קובץ שמכיל את השכיחות של צמדי אותיות.

אתחול

אתחלנו אוכלוסייה בגודל 100 כאשר כל פרט באוכלוסיה הוא בעצם candidate אפשרי לפתרון – האוכלוסיה מורכבת מ-dictionaries, כאשר כל dictionary מייצג מיפוי בין אותיות לפענוח שלהם.

האלגוריתם

הגבלנו את המעבר על האוכלוסיה בלכל היותר 150 דורות.

בדור הראשון (דור 0), נחשב fitness לכל איבר באוכלוסיה. בדורות הבאים – נמיין את הפתרונות לפי ציון ה-fitness שלהם וניקח את ה-20% הטובים ביותר כדי ליצור את next generation.

לאחר מכן, אנחנו יוצרים offspring בעזרת crossover של 2 הורים (פתרונות), ועושים עליו mutation בהסתברות מסוימת (גבוהה). את הצאצא אנחנו מוסיפים לדור הבא, וכך נמשיך בלולאה עד שנגיע שוב לאוכלוסייה בגודל 100 שתחליף את האוכלוסיה הקודמת.

נמשיך עד שלא יהיה שינוי בציון ה-fitness הטוב ביותר ושם בעצם האלגוריתם יעצור.

פונקציית ה-fitness

לצורך חישוב ציון ה-fitness השתמשנו ב-3 הקבצים שצורפו במודל – dict.txt, Letter_Freq.txt, Letter2_Freq.txt. אנחנו בודקים עבור כל מילה את השכיחות שלה בקובץ לאחר הפענוח ומשווים עם המידע בקובץ Letter_Freq.txt. בנוסף, אנחנו בודקים עבור כל צמד אותיות את השכיחות שלהן בקובץ לאחר הפענוח ומשווים עם המידע בקובץ Letter2_Freq.txt. לבסוף, אנחנו מבצעים חיתוך בין המילים בקובץ לאחר הפענוח (כאשר הפכנו את הקובץ ל-set כדי למנוע כפילויות של מילים) לבין המילים שבקובץ dict.txt. סה"כ אלה 3 הפרמטרים שיקבעו את ציון ה-fitness.

crossover בין הפתרונות

לקחנו 2 הורים והגרלנו אינדקס בין 0 ל-25 שישמש כנקודת ה-crossover ביניהם. לאחר מכן בדקנו שאין כפילויות של אותיות בצאצא שנוצר, ע"י החלפה רנדומלית של אות כפולה באות שלא נמצאת בפתרון. את 2 ההורים בחרנו בעזרת הפונקציה get_cross_parents שפועלת בשיטה של טורניר – נגריל 5 פתרונות מתוך האוכלוסיה, נמיין אותם לפי ציון ה-fitness (מהגבוה לנמוך) וניקח את ההורה הראשון להיות הפתרון הראשון מבין ה-5 (לאחר המיון). לאחר מכן נחזור שוב על התהליך כדי לבחור את ההורה השני. כך נחזיר 2 פתרונות שישמשו כהורים בתהליך ה-crossover.

מימוש המוטציות

לצורך ביצוע שלב המוטציה, הגרלנו 2 אינדקסים בפתרון והחלפנו את האותיות במיקומים האלה. את המוטציה עצמה הפעלנו בהסתברות MUTATION_RATE – פרמטר איתו שיחקנו והגענו למסקנה שמוטציה בהסתברות גבוהה מביאה לתוצאות טובות יותר כפי שיפורט בהמשך.

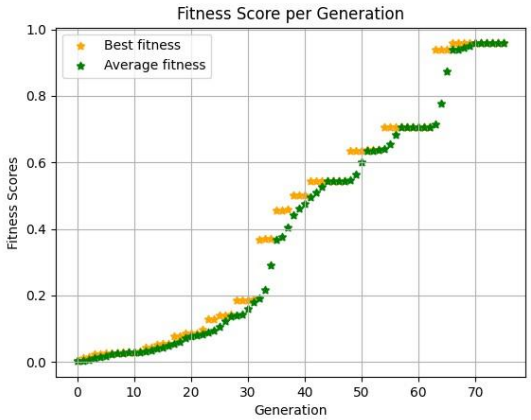
התכנסות מוקדמת

טיפלנו בהתכנסות מוקדמת ע"י כך שהגדרנו 2 פרמטרים: ערך threshold וחסם על מספר ריצות האלגוריתם. אנחנו ממשיכים להריץ את האלגוריתם הגנטי שוב ושוב עד שציון ה-fitness הטוב ביותר עובר את ערך ה-threshold שהגדרנו או עד שהגענו למספר מסוים של ריצות (5). בכל ריצה כזאת האלגוריתם מייצר decryption key ובודק את ציון ה-fitness שלו – אם קיבלנו ציון fitness טוב יותר נעדכן אותו ואת ה-decryption key המתאים. לבסוף נשתמש ב-decryption key הטוב ביותר בשביל לייצר את הקובץ המפוענח שירשם לקובץ plain.txt.

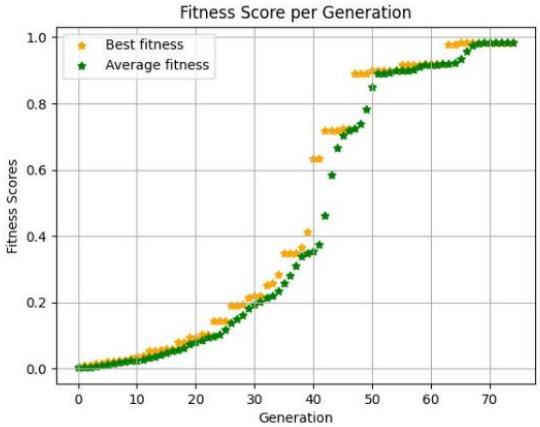
השוואת פרמטרים

גודל אוכלוסייה:

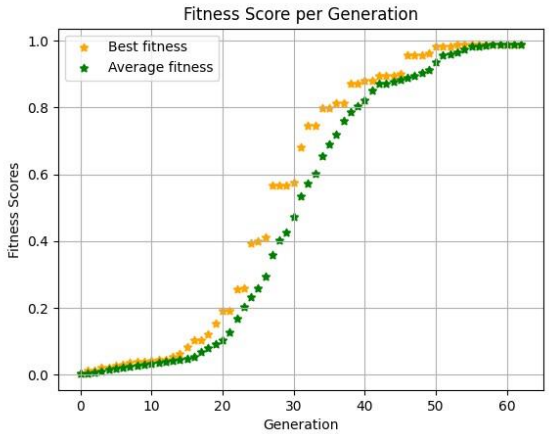
גודל אוכלוסייה של 50 – 9350 קריאות לפונקציית ה-fitness, התכנסות סביב דור 70.



גודל אוכלוסייה של 100 – 7650 קריאות לפונקציית ה-fitness, התכנסות סביב דור 70.

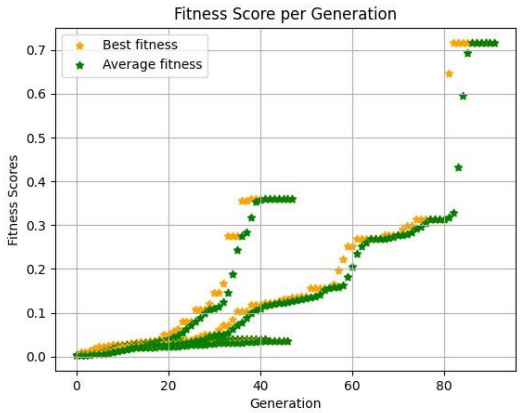


גודל אוכלוסייה של 300 – 38,100 קריאות לפונקציית ה-fitness (איטי יחסית), התכנסות סביב דור 60.

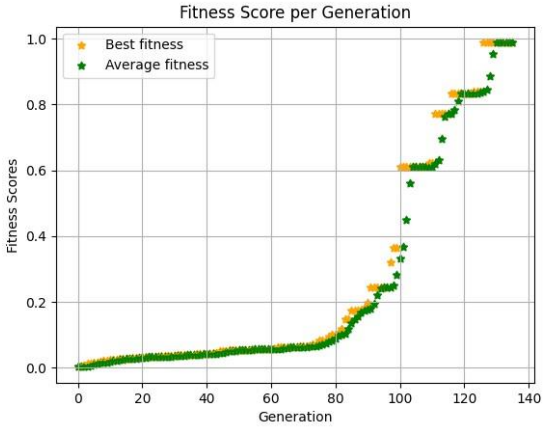


הסתברות למוטציה :

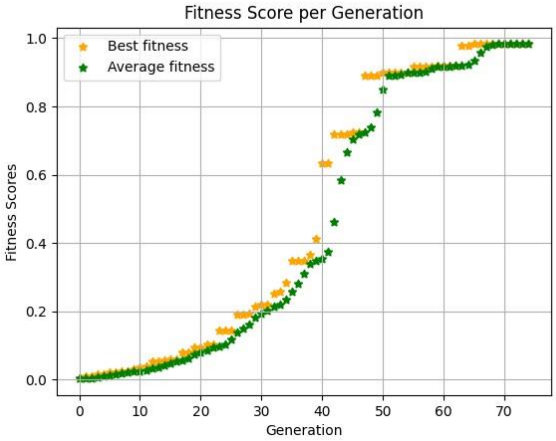
הסתברות של 0.2 לבצע מוטציה – כפי שניתן לראות בגרף למטה, הגענו כמה וכמה פעמים להתכנסות מוקדמת עד שלבסוף ריצה אחת כן הצליחה לאחר 80+ דורות וסה"כ 52,900 קריאות ל-fitness.



הסתברות של 0.5 לבצע מוטציה – לא הייתה התכנסות מוקדמת אך לאחר 80 דורות היה ניתן לראות שיפור משמעותי בציון ה-fitness ורק בסביבות הדור ה-140 האלגוריתם התכנס עם סה"כ 27,300 קריאות.

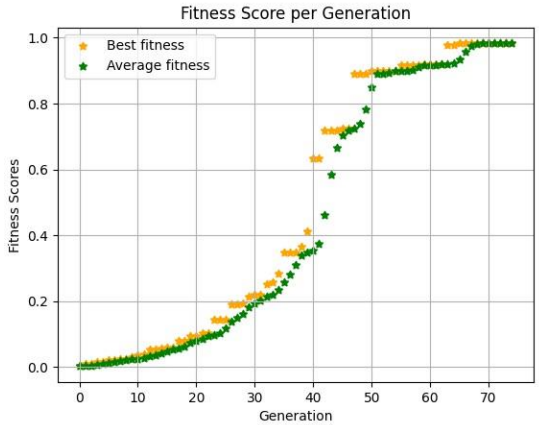


מוטציה בהסתברות של 1 – ניתן לראות שההתכנסות היא סביבות ה-70 דורות עם 15,100 קריאות לפונקציית ה-fitness. הסקנו מנתונים אלו שכנראה תמיד כדאי לבצע מוטציה.

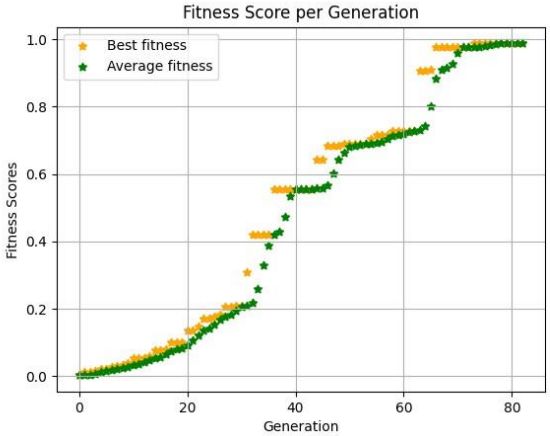


גודל קבוצת ה-elite :

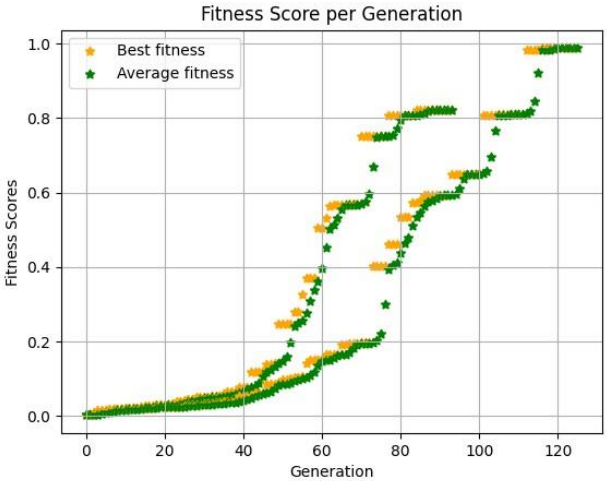
כאשר לקחנו 20% מהפתרונות הטובים ביותר כדי ליצור את הדור הבא (15,100 קריאות, התכנסות לאחר 70 דורות) -



כאשר לקחנו 50% מהפתרונות הטובים ביותר כדי ליצור את הדור הבא (16,700 קריאות, התכנסות לאחר 80 דורות) -



כאשר לקחנו 70% מהפתרונות הטובים ביותר כדי ליצור את הדור הבא (44,200 קריאות, התכנסות לאחר 120 דורות, עם כמה התכנסויות מוקדמות בדרך) –



סה״כ הסקנו שהערכים האידיאלים לפרמטרים הנבחנים הם אוכלוסייה בגודל 100, עם הסתברות 1 למוטציה וקבוצת elite של 20% מהפתרונות הטובים.

גרסה דארווינית וגרסה למארקית

באלגוריתם הדארוויני – חישוב ה-fitness של כל פתרון מתבצע לאחר אופטימיזציה לוקלית, אך מה שמועבר לדור הבא הוא "הגנום" של הפתרון **לפני** האופטימיזציה.

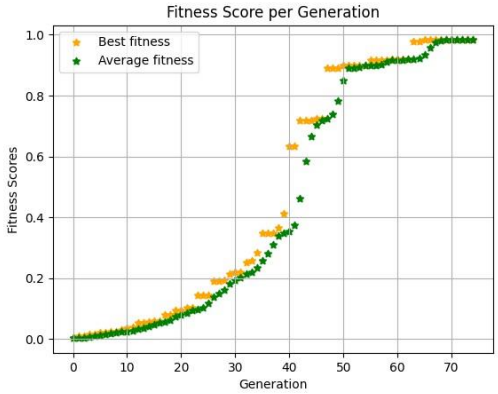
ממשנו את האלגוריתם כך – על כל פרט באוכלוסיה הפעלנו אופטימיזציה לוקלית וחישבנו את ה-fitness החדש שלו. את האופטימיזציה עשינו בעזרת הפונקציה local_optimization שלוקחת אות רנדומלית ואת המיפוי שלה מהמפתח עם ה-fitness הכי טוב עד לדור הנוכחי, מכניסה אותם למפתח שעליו מתבצעת המוטציה, ולבסוף מבצעת תיקון על מנת שהמפתח יהיה עדיין חוקי.

באלגוריתם הלמארקי – חישוב ה-fitness של כל פתרון מתבצע לאחר אופטימיזציה לוקלית כמו בדארווין, אך מה שמועבר לדור הבא זה "הגנום" **לאחר** האופטימיזציה.

ממשנו את האלגוריתם בדומה לגישה הדרוויניסטית, אך הפעם אם לפתרון לאחר המוטציה היה fitness גבוה יותר מהפתרון המקורי – העברנו אותו ואת ציון ה- fitness שלו לדור הבא.

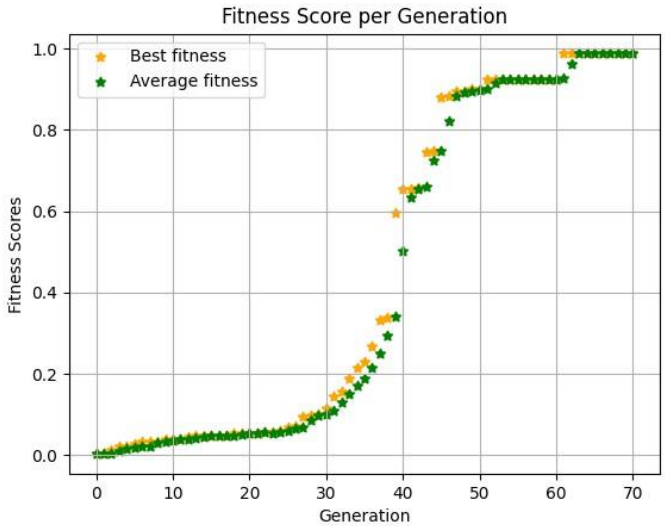
השוואה בין הגישות השונות

האלגוריתם הגנטי **הרגיל** –



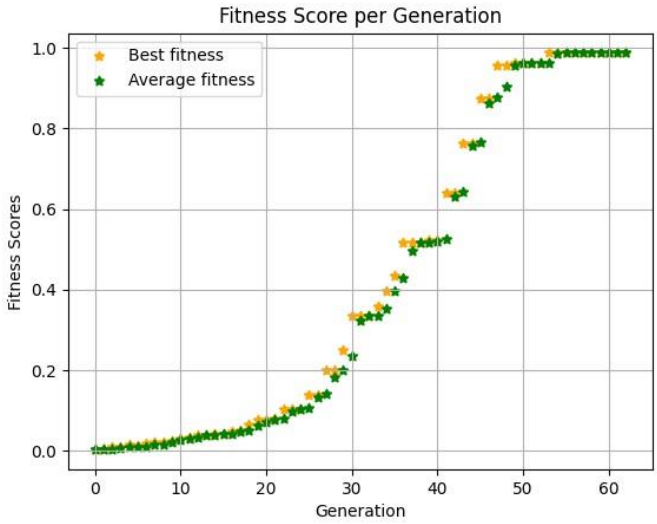
התכנסות סביבות דור 75, 15,100 קריאות.

לפי הגישה של **דארווין** –



התכנסות סביבות דור 71, 21,400 קריאות.

לפי הגישה של **למארק** –



התכנסות סביבות דור 63, 19,000 קריאות.

מבחינת **מספר הקריאות** לפונקציית ה-fitness : ניתן לראות כי באלגוריתם הדארוויניסטי מספר הקריאות הוא הגבוה ביותר, לאחר מכן באלגוריתם הלמארקי ולבסוף באלגוריתם הגנטי הרגיל. ניתן להסביר זאת ע״י כך שבאלגוריתמים של דארווין ולמארק אנחנו מבצעים אופטימיזציות שכרוכות בחישובי fitness נוספים (מעבר למה שקורה באלגוריתם הרגיל). ואם נשווה בין האלגוריתם של דארווין ללמארק – הגיוני שמספר הקריאות לפונקציית ה-fitness יהיה גבוה יותר אצל דארווין כיוון שמה שעובר לדור הבא זה הגנום של הפתרון לפני האופטימיזציה ולכן יש סיכוי גדול יותר שנעשה יותר חישובים.

מבחינת **מספר הדורות** שעברו עד להתכנסות : ניתן לראות שהאלגוריתם שמגיע להתכנסות הכי מהר הוא האלגוריתם עבור גישת למארק, מה שמתיישב עם העובדה שאצל למארק אנחנו מתייחסים ל-fitness החדש כלומר מה שמועבר לדור הבא זה הגנום של הפתרון לאחר האופטימיזציה – והסיכוי שהפתרונות האלו יהיו טובים גדול יותר.