מציאת פרמטרים מיטביים ל־GA על בעית הפונקציה

גלעד הרצפלד וטל ואניש

10 ביולי 2016

פרויקט סיכום – אלגוריתמים אבולוציוניים 202.2.5651

המחלקה למדעי המחשב, אוניברסיטת בן גוריון

גלעד הרצפלד - giladh11@gmail.com

טל ואניש - tal.vanish@gmail.com

תוכן עניינים

1 מבוא

בפרק זה נציג את הסיבות המרכזיות בגינן בחרנו לעסוק במציאת פרמטרים מיטביים ל־GA, וכן נתאר באופן כללי את הבעיה ומרכיביה.

1.1 הקדמה

במהלך הקורס "אלגוריתמים אבולוציוניים" הוצגו בפנינו בעיות שונות שנפתרו ע"י GA. בעיות אלו השתמשו בערכים שונים עבור הפרמטרים של ה־GA: גודל אוכלוסיה, הסתברות ל־$Crossover$ ול־$Mutation$ ועוד. נדמה שערכים אלו נקבעו לאחר ניסוי וטעיה ע"י החוקרים, ללא הנמקה סדורה לפני ביצוע הניסוי. רצינו לבחון אם יש הגיון מאחורי ערכים אלו, ואם ניתן לאפיין את הקשרים בין ערכים אלו לטיב הפתרון של הבעיה.

בחרנו לענות על שאלה זו ע"י החלת תהליך האבולוציה על פרמטרי ה־GA עצמם באמצעות בחינה של בעיית הפונקציה, וריאציה קלה על בעיה ידועה בשם Symbolic Regression. נרצה להשתמש באלגוריתם אבולוציוני כדי למצוא מופע מיטבי של GA לפתרון בעיית הפונקציה. מופעים שונים של GA נבדלים אחד מן השני בערכים של הפרמטרים האבולוציוניים (גדלים שונים של אוכלוסיה, הסתברויות שונות לאופרטורי שינוי וכיו"ב). לכן, נגדיר מופע מיטבי של GA ככזה המבטיח ביצועים מהירים וחל על משפחות שונות של פונקציות.

בעבודה זו ננסה לנתח את הקשר בין הפרמטרים האבולוציוניים של ה־GA לבין טיבו כפותר בעיית הפונקציה. נרצה לבדוק אם ייתכן וכדאי להשקיע בחיפוש פרמטרים מיטביים ל־GA לפני הרצתו, וכיצד פרמטרים מסוימים משפיעים על יכולתו של ה־GA לפתור פונקציות ממשפחות שונות. את התהליך האבולוציוני המוצא פתרון לבעיית הפונקציה נכנה "אבולוציה מסדר ראשון". באופן דומה, את התהליך האבולוציוני המוצא מופע מיטבי של GA לבעיית הפונקציה נכנה "אבולוציה מסדר שני".

1.2 הגדרת בעיית הפונקציה – אבולוציה מסדר ראשון

מופע של בעיית הפונקציה הינו "קופסה שחורה" אשר מגלמת פונקציה כלשהי $f : \mathbb{R} \to \mathbb{R}$. ניתן לתשאל את הקופסה על הערכה שלה לנקודה כלשהי $x \in \mathbb{R}$ ולקבל ערך $f(x)$. המטרה היא למצוא פונקציה כלשהי $g : \mathbb{R} \to \mathbb{R}$ כך ש־$|f(x) - g(x)|$ יהיה קטן ככל האפשר לכל $(x, f(x)) \in D$, כאשר $D \subset \mathbb{R}$ קבוצת הנקודות הנבחנת.

את קבוצת המופעים של הבעיה ניתן לחלק באופן טבעי למשפחות שונות של פונקציות: פולינומים ממעלה k, פונקציות טריגונומטריות, פונקציות מעריכיות וכיו"ב.

1.3 חיפוש GA מיטבי לבעיית הפונקציה – אבולוציה מסדר שני

בהנתן מופע של בעיית הפונקציה, ניתן למצוא פתרון ע"י GA פשוט: פרטים באוכלוסיה הם פונקציות המיוצגות ע"י עצים מעל קבוצת טרמינלים ופעולות אריתמטיות, ואופרטורי $Crossover$ ו־$Mutation$ הפועלים בהסתברות כלשהי

בהתאמה. בנוסף, יש לקבוע מראש את גודל האוכלוסיה P; גודל קבוצת הנקודות הנבחנת s; וכן את העומק המירבי של העצים בדור הראשון d.

לפיכך, ניתן לתאר GA כקונפיגורציה המכילה ערכים אפשריים עבור סדרת הפרמטרים האבולוציוניים $\mathrm{ParamGA} = (P, p_c, p_m, s, d)$. בהנתן קבוצה O של מופעים לבעיית הפונקציה, נרצה למצוא קונפיגורציה של GA המוצאת את הפתרונות הטובים ביותר עבור O ביעילות הגבוהה ביותר. כדי למצוא קונפיגורציה כזו, נשתמש ב־GA כאשר הפרטים באוכלוסייה הם קונפיגורציות ה־GA לבעיית הפונקציה. פירוט נוסף על מהלך האבולוציה בחיפוש ה־GA המיטבי מובא בהמשך (פרק 5).

2 שאלות מחקר

בעבודה ננסה לענות על שתי שאלות מחקר עיקריות:

א. האם קיימים פרמטרים למופע של GA המניבים שיפור משמעותי ביעילות הריצה על בעיית הפונקציה?

הרצה של אלגוריתמים אבולוציוניים מתאפיינת בזמני ריצה ארוכים במיוחד בשל הסימולציות הרבות על הפרטים השונים בכל דור. נרצה לבדוק כיצד פרמטרים שונים ל־GA משפיעים על יעילות הריצה ולברר אם ניתן למצוא פרמטרים מיטביים לשם כך. ננסה למצוא פרמטרים שמגיעים לסף הדיוק הנדרש באופן יעיל יותר מפרמטרים אחרים, או לחלופין מגיעים לחסם העליון על מספר הדורות אך משיגים פתרון מדויק יותר. אם נמצא פרמטרים כאלו, נוכל לשער שייתכן ששיפור כזה קיים גם בשיטות אבולוציוניות אחרות, כגון GP. השערה זו יכולה להוות עילה להמשך מחקר עבור בעיות מורכבות יותר מהבעיה שבחנו.

ב. כיצד פרמטרים מיטביים עבור משפחת פונקציות אחת יתפקדו עבור משפחת פונקציות אחרת?

אחד המדדים שמשפיעים על טיבו של GA לבעיית הפונקציה הוא יכולתו להפיק פתרון טוב ביעילות גבוהה למשפחות שונות של פונקציות. מדד זה משקף את רצוננו למצוא קונפיגורציה ל־GA אשר תתאים לכמה שיותר סוגים שונים של פונקציות. באופן טבעי ניתן לשאול כיצד קונפיגורציה ל־GA שתפקדה כיאות מעל משפחת הפולינומים, לדוגמה, תתפקד מעל משפחת הפונקציות המעריכיות. נרצה לגלות אם אכן יש קשר בין משפחות שונות של פונקציות עבור קונפיגורציה כלשהי. אם נמצא פרמטרים ל־GA המתאימים למשפחות שונות של פונקציות ב"בעיית הפונקציה" נוכל לשער שייתכן שפרמטרים אלו יהיו תקפים גם לבעיות אחרות. כמובן שמחקר נוסף ידרש בנושא.

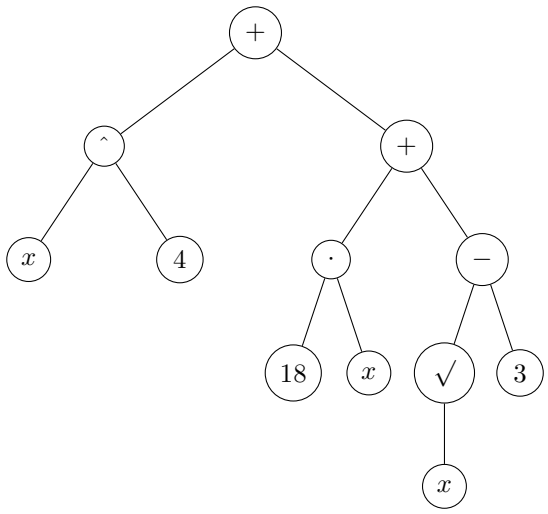
*נעיר כי שאלה (ב) תקפה במידה שאכן נמצאו פרמטרים המבטיחים דיוק יעילות גבוהים יותר בהשוואה לפרמטרים אחרים.

3 מהלך האבולוציה – בעיית הפונקציה

בפרק זה נסקור את מהלך האבולוציה מסדר ראשון עבור בעיית הפונקציה. כזכור, מופע לבעיית הפונקציה הינו "קופסה שחורה" המגלמת פונקציה כלשהי $f:\mathbb{R}\to\mathbb{R}$, ופתרון לבעיה הינו מודל כך $g:\mathbb{R}\to\mathbb{R}$ כך ש־$|f(x)-g(x)|$ קטן ככל האפשר לכל $(x,f(x))\in D$, כאשר D זו קבוצת הנקודות הנבחנת.

3.1 ייצוג הבעיה

פרטים באוכלוסיה הם פונקציות המיוצגות ע"י עצים מעל קבוצה מסוימת של טרמינלים ופעולות. לדוגמה, הפונקציה $f(x)=x^4+18x+\sqrt{x}-3$ מתאימה לעץ הבא:



3.2 אופרטורי השינוי

- אופרטור הזיווג (Crossover) - בהנתן שני הורים T_1,T_2, מפעילים עליהם בהסתברות p_c את אופרטור הזיווג הבוחר שני תת־עצים ST_1,ST_2 באופן אקראי משני ההורים בהתאמה, ויוצר שני ילדים חדשים: הילד הראשון הינו T_1, אך במקום תת־העץ ST_1 מופיע תת־העץ ST_2. הילד השני נבנה באופן סימטרי.
- אופרטור המוטציה (Mutation) - בהנתן עץ T, אופרטור המוטציה מבצע אחד מן השינויים הבאים על העץ בהסתברות p_m:
  - בחירת צומת כלשהי המייצגת פונקציה k־מקומית, והחלפתה בפונקציה k־מקומית אחרת.
  - בחירת תת־עץ שמאלי או ימני של השורש, והחלפתו בתת־עץ אקראי.
  - בחירת צומת כלשהי והחלפתה בתת־עץ כלשהו של אותו העץ.

- בחירת צומת כלשהי המייצגת פונקציה לא קומוטטיבית, והיפוך סדר הארגומנטים שלה.
- הפיכת העץ לתת־עץ מושרש עבור שורש כלשהו המייצג פונקציה שנבחרה באקראי.
- החלפת העץ כולו בעץ אקראי מעומק 2.
- החלפת העץ כולו בתת־עץ אקראי של אותו העץ.

3.3 פונקצית הניקוד (lFitness[3])

התאמתו של פרט באוכלוסיה נקבעת לפי המרחק שלו מן הפתרון הנכון. אם f_T היא הפונקציה המתוארת ע"י העץ T, אזי התאמתו של T מוגדרת ע"י:

$$lFitness\left(T\right)=\frac{\sum_{(x,y)\in D}\left|f_T\left(x\right)-y\right|}{\left|D\right|}$$

לפיכך, ככל ש־$lFitness\left(T\right)$ נמוך יותר, כך מידת התאמתו של T גבוהה יותר.

3.4 מהלך האבולוציה

האבולוציה מתבססת על הפרמטרים הבאים:

- $GENERATIONS$ - מספר הדורות המרבי עבור האבולוציה.
- ε - גודל חיובי קטן כלשהו.

חמשת הפרמטרים האחרונים יועמדו לאבולוציה בשלב הבא:

- p_c - ההסתברות לביצוע Crossover בין שני פרטים.
- p_m - ההסתברות לביצוע Mutation על פרט מסוים.
- P - גודל האוכלוסיה.
- s - גודל קבוצת הנקודות D אל מולה מודדים את התאמתם של פרטים באוכלוסיה.
- d - עומק העצים המרבי של הפרטים בדור הראשון.

כעת, נתאר את מהלך האבולוציה:

(א) בנה קבוצת נקודות D באמצעות הקופסה השחורה.

(ב) הגרל אוכלוסיה ראשונית בגודל P של עצים לכל היותר בעומק d, המייצגים פתרונות אפשריים לבעיה.

[3] קיצור של Lower Level Fitness.

(ג) $generation \leftarrow 1$.

(ד) כל עוד $generation < GENERATIONS$ וגם $(\exists T \text{ s.t } lFitntess\,(T) < \varepsilon)$!

() $newPopulation \leftarrow oldPopulation$.

() בחר באקראי $\frac{P}{2}$ זוגות באוכלוסיה ועבור כל אחד מהם בצע Crossover בהסתברות p_c. הוסף את הצאצאים ל־$newPopulation$.

() לכל פרט ב־$oldPopulation$, בצע Mutation בהסתברות p_m. הוסף את הפרטים החדשים ל־$newPopulation$.

() מיין את הפרטים לפי ה־$lFitntess$ וצמצם את $newPopulation$ ל־P הפרטים הטובים ביותר.

() $generation \leftarrow generation + 1$.

4 סקירת תכנה – אבולוציה מסדר ראשון לפתרון בעיית הפונקציה

בפרק זה נסקור בקצרה את התכנה למימוש האבולוציה על בעיית הפונקציה. נדגיש כי הפירוט המובא כאן חלקי בלבד, ומכיל את מרכיבי המערכת המהותיים להבנת התהליך. ניתן לבחון את המימוש המלא ב־Git: `https://github.com/giladh11/BestParamsProjectGA.git` תחת ה־lowerLevelGA packages ו־evolutionGaTools.

- BlackBoxTree - מחלקה זו מייצגת מופע לבעיית הפונקציה.
  - שדות:
    - $function$ - פונקצית המטרה הטמונה בתוך הקופסה השחורה, מיוצגת ע"י עץ.
  - מתודות:
    - $eval\,(x)$ - המתודה מקבלת ערך ל־x ומחזירה את ההערכה של $function$ על x.
- FunctionTreeChromosome - מחלקה זו מייצגת פרט באוכלוסיה עבור האבולוציה מסדר ראשון.
  - שדות:
    - $syntaxTree$ - עץ המייצג את המודל הנוכחי לקופסה השחורה.
  - מתודות:
    - $crossover\,(other)$ - המתודה מקבלת פרט אחר באוכלוסיה, ומבצעת זיווג בין הפרטים.
    - $mutate\,()$ - המתודה מבצעת מוטציה על הפרט.
- SymRegSolverChromosome - מחלקה זו מייצגת את המנוע המחולל את האבולוציה מסדר ראשון על מופעים של FunctionTreeChromosome. מאחר ומחלקה זו מייצגת פרט באוכלוסיה באבולצה מסדר שני, היא מכילה מתודות נוספות עבור האופרטורים האבולוציוניים עליהן נפרט בפרק הבא (פרק 5.2).
  - שדות:
    - MAX_NUM_OF_ITERATIONS_LOWER_LEVEL - המספר המירבי של דורות עבור האבולוציה מסדר ראשון לפתרון בעיית הפונקציה. משמש כתנאי עצירה עבור האבולוציה מסדר ראשון.
    - EPSILON_DISTANCE_FOR_LOWER_EVOLUTION_TO_STOP - ערך מספרי הקובע את הסף הרצוי עבור פתרון אפשרי לבעיית הפונקציה. משמש כתנאי עצירה עבור האבולוציה מסדר ראשון.
    - $ParamGA$ - אובייקט המגדיר פרמטרים הכרחיים לביצוע אבולוציה מסדר ראשון, וכן מייצג את הפרמטרים שעומדים לאבולוציה מסדר שני. מכיל את הפרמטרים שהוזכרו קודם לכן: גודל

האוכלוסייה, הסתברות ל־Crossover, הסתברות ל־Mutation, גודל קבוצת הנקודות D וכן את עומק העצים המירבי עבור הדור הראשון d.

- $baseFunctions$ - קבוצה של פונקציות מעליה נבנים העצים באוכלוסיה באבולוציה מסדר ראשון לפתרון בעיית הפונקציה.

○ מתודות:

- $trySolving\,(\text{BlackBoxTree}\ blackbox)$ - המתודה מקבלת קופסה שחורה המייצגת מופע לבעיית הפונקציה, ומחוללת את התהליך האבולוציוני למציאת המודל הטוב ביותר לקופסה השחורה. תוך כדי פעולתה היא מעדכנת את שדה ה־$effort$.
- $crossover\,(other)$ - המתודה מקבלת פרט אחר באוכלוסיה של האבולוציה מסדר שני, ומבצעת זיווג בין הפרטים. פירוט מובא בהמשך (פרק 5.2)
- $mutate\,()$ - המתודה מבצעת מוטציה על הפרט באוכלוסיה של האבולוציה מסדר שני. פירוט מובא בהמשך (פרק 5.2).

• BestModelCandidate - מחלקה זו מייצגת את המודל הטוב ביותר לקופסה השחורה כפי שנמצא ע"י SymRegSolverChromosome. מאחר ומחלקת SymRegSolverChromosome משמשת אותנו לאבולוציה מסדר שני, נרחיב על השדות והמתודות שלה בסקירת התכנה הבאה (פרק 6).

○ שדות:

- $bestSyntaxTree$ - עץ המייצג את המודל המקורב לקופסה שחורה.
- $effortElement$ - שדה זה מייצג את המאמץ של SymRegSolverChromosome כדי למצוא את $bestSyntaxTree$. המאמץ הוא שקלול של מספר הדורות, מספר פעולות הזיווג והמוטציה ומספר הנקודות ששוערכו במהלך האבולוציה. נפרט על כך בהמשך (פרק 5.4).
- $distanceFromBlackBox$ - המרחק הממוצע עבור OBJECTIVE_NUM_OF_POINTS נקודות של המודל מהקופסה השחורה.[4] נעיר כי בעוד שהמרחק של המודל מן הקופסה השחורה עבור הנקודות אל מולן נבחן המודל צפוי להיות קטן, המרחק של המודל מן הקופסה השחורה עבור מספר גדול יותר של נקודות צפוי להיות גדול יותר.
- $hFitnessElement$ - שדה זה הוא שקלול של $distanceFromBlackBox$ ו־$effortElement$. פירוט מובא בהמשך (פרק 5.4).

○ מתודות:

[4] כלומר, אם $P = \overline{\left\{(x,y)\,\middle|\,y = blackbox.function\,(x)\right\}}$ כך ש־$|P| = \text{OBJECTIVE\_NUM\_OF\_POINTS}$ ו־f_{best} היא הפונקציה המתאימה ל־$bestSyntaxTree$, אזי המרחק נתון ע"י $\frac{\sum_{(x,y)\in P}|f_{best}(x)-y|}{|P|}$.

- $hFitnessCalculator$ - המתודה משקללת את $distanceFromBlackBox$ ואת $effortElement$ ונותנת ציון למודל על בסיסם.

5 מהלך האבולוציה – מציאת GA מיטבי לבעיית הפונקציה

בפרק זה נסקור את מהלך האבולוציה מסדר שני עבור מציאת GA מיטבי לבעיית הפונקציה. מופע לבעיה זו הינו קבוצה O של מופעים לבעיית הפונקציה, ופתרון לבעיה הינו GA המספק את המודל הטוב ביותר לכל $blackbox \in O$ ביעילות הגבוהה ביותר. בפרק זה נגדיר במדויק כיצד בחרנו למדוד טיב של פרמטרים ויעילותם.

5.1 ייצוג הבעיה

פרטים באוכלוסיה הם קונפיגורציות של GA. קונפיגורציה של GA היא סדרת ערכים עבור הפרמטרים האבולוציוניים של ה־GA: גודל האוכלוסיה P, ההסתברות לזיווג p_c, ההסתברות למוטציה p_m, גודל קבוצה הנקודות הנבחנת s והעומק המרבי של העצים באוכלוסיה הראשונית d. לכן, ניתן לתאר GA באופן הבא:

P	p_c	p_m	s	d

כאשר: $5 \leq P \leq 10, 0 \leq p_m, p_c \leq 1, 5 \leq s \leq 20, 1 \leq d \leq 8$.

5.2 אופרטורי השינוי

- אופרטור הזיווג (Crossover) - בהנתן שני הורים GA_1 ו־GA_2, אופרטור הזיווג מופעל בהסתברות p_c ובוחר באקראי נקודת זיווג $1 \leq crossoverPoint \leq 5$ באופן אחיד. האופרטור יוצר שני צאצאים: האחד מכיל את הערכים מן ההורה הראשון עד לנקודת הזיווג והחל ממנה את הערכים מן ההורה השני, והשני ההפך.
- אופרטור המוטציה (Mutation) - בהנתן פרט GA, אופרטור המוטציה יוצר פרט חדש בהסתברות $\hat{p_m}$ באופן הבא: אחד הפרמטרים האבולוציוניים נבחר באופן אקראי ובמקום הערך הנוכחי מוגרל ערך כלשהו בטווח בערכים החוקיים של אותו פרמטר.

5.3 חידוד על פונקצית הניקוד בשני הסדרים

לפני שנגדיר במדויק את hFitness, נחדד את הגדרת ה־Fitness עבור האבולוציה מסדר ראשון ועבור האבולוציה מסדר שני באופן כללי.

- lFitness - זהו רכיב ה־Fitness באבולוציה מסדר ראשון לבעיית הפונקציה. ככזה, הוא מייצג את טיב התאמתו של מודל לקופסה שחורה נתונה. נסמן:
  - $f_{blackBox}$ - הפונקציה הטמונה בתוך הקופסה השחורה.
  - g_{model} - המודל שנמצא לקופסה השחורה בתהליך האבולוציוני.
  - P - קבוצת נקודות של הקופסה השחורה, כלומר $P = \left\{(x, y) \middle| y = f_{blackBox}(x)\right\}$.

לפיכך, lFitness של מודל כלשהו מוגדר ע"י:

$$\frac{\sum_{(x,y)\in P} |g_{model}(x) - y|}{|P|}$$

- hFitness - זהו רכיב ה־Fitness באבולוציה מסדר שני למציאת GA מיטבי. תפקידו לייצג את טיב ביצועו של תהליך אבולוציוני עם קבוצת פרמטרים ParamGA על קבוצה O של קופסאות שחורות. כדי לחשב את hFitness מעל הקבוצה O, עלינו להתחשב בביצועים של התהליך האבולוציוני על כל קופסה שחורה בנפרד[5]. נכנה את הביצועים של התהליך על קופסה שחורה **אחת** ב־hFitnessElement. ערך זה הינו שקלול של טיב המודל שנמצא לקופסה(lFitness) יחד עם המאמץ שהושקע בה (תיאור מפורט של השקלול בסעיף הבא).

בבואנו לחשב את ה־hFitness, נרצה להתחשב ב־hFitnessElement של כל קופסה שחורה. לכן, עבור קבוצה M של מודלים שנמצאו לקבוצה O, נגדיר את הניקוד של פרט ב־SymRegSolverChromosome באבולוציה מסדר שני ע"י:

$$\frac{\sum_{m\in M} m.hFitnessElement}{|M|}$$

5.4 פונקצית הניקוד (hFitness[6])

נזכיר כי לכל $bestModel$, שנמצא ע"י אבולוציה מסדר ראשון על קופסה שחורה, קיים שדה $hFitnessElement$ המייצג את שקלול טיבו כמודל של הקופסה השחורה וכן של המאמץ שהושקע כדי למצוא אותו. לכן, לפני שנגדיר את פונקצית הניקוד של האבולוציה מסדר שני, נגדיר את המרכיבים של $hFitnessElement$ עבור מודל כלשהו m:

- הרכיב הראשון של $hFitnessElement$ עבור המודל הינו $effort$, המוגדר ע"י:

$$effort(m) = \alpha_1 \cdot generationNum + \alpha_2 \cdot numOfCrossover + \alpha_3 \cdot numOfMutations$$
$$+ \alpha_4 \cdot numOfPointsEvaluated + \alpha_5 \cdot sumOfTreesSizesCreated$$

כאשר $\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5 \in \mathbb{N}$ קבועי נרמול. קבענו את ערכי α_i כך ש־$effort(m)$ יהיה בסדר גודל של 10^2, וישקפו את יחס המאמצים בין חמשת הפרמטרים.

- הרכיב השני של $hFitnessElement$ עבור המודל הינו $distanceFromBlackBox$, כפי שהוגדר קודם לכן.

כעת, ניתן להגדיר הניקוד של מודל m ע"י:

$$hFitnessElement(m) = C_1 \cdot distanceFromBlackBox + C_2 \cdot effort(m)$$

[5] כשם ש־lFitness מתחשב במרחק של מודל מהקופסה השחורה לכל נקודה בנפרד.
[6] קיצור של Higher Level Fitness.

כאשר $C_1 = \frac{1}{\varepsilon} \cdot 1000, C_2 = 1$. הקבוע C_1 מנרמל[7] את $distanceFromBlackBox$ לסדר גודל של 10^3 ומהווה את הרכיב המרכזי בחישוב ה־$hFitness$. מודל שהגיע לדיוק הנדרש יקבל $\text{hFitness} = 1000$. עבור מודלים שהגיעו לסף דיוק דומה ברכיב ה־$C_1 \cdot distanceFromBlackBox$, הרכיב $C_2 \cdot effort(m)$ יקבע אילו מן המודלים בעל ניקוד גבוה יותר.

נזכיר כי מופע לבעיה הינו קבוצה O של קופסאות שחורות אשר יש למצוא להן מודלים טובים ככל הניתן. לכן, אם S היא קבוצת המודלים הטובים ביותר עבור קבוצה O שנמצאו ע"י פרט GA מסוג SymRegSolverChromosome, ניתן להגדיר באופן טבעי את פונקצית הניקוד של האבולוציה מסדר שני עבור אותו הפרט ע"י:

$$\text{hFitness}\left(\text{SymRegSolverChromosome}\right) = \frac{\sum_{bestModel \in S} hFitnessElement\left(bestModel\right)}{|S|}$$

5.5 מהלך האבולוציה

האבולוציה מתבססת על הפרמטרים הבאים:

- $\hat{p_c}$ - ההסתברות לביצוע Crossover בין שני פרטים. בחרנו $\hat{p_c} = 0.8$.
- $\hat{p_m}$ - ההסתברות לביצוע Mutation על פרט מסוים. בחרנו $\hat{p_m} = 0.25$.
- $\hat{P}$ - גודל האוכלוסיה. בחנו כמה גדלים אפשריים של אוכלוסייה: $10 \leq \hat{P} \leq 20$.
- $GENERATIONS$ - מספר הדורות המרבי עבור האבולוציה. בחרנו $GENERATIONS = 50$.

כעת, נתאר את מהלך האבולוציה:

(א) הגרל אוכלוסיה ראשונית בגודל $\hat{P}$.

(ב) $generation \leftarrow 1$.

(ג) כל עוד $generation < GENERATIONS$

() $newPopulation \leftarrow oldPopulation$.

() בחר $\frac{\hat{P}}{2}$ זוגות באוכלוסיה ועבור כל אחד מהם בצע Crossover בהסתברות $\hat{p_c}$. הוסף את הצאצאים ל־$newPopulation$.

() לכל פרט ב־$oldPopulation$, בצע Mutation בהסתברות $\hat{p_m}$. הוסף את הפרטים החדשים ל־$newPopulation$.

() מיין את הפרטים לפי ה־hFitness וצמצם את $newPopulation$ ל־$\hat{P}$ הפרטים הטובים ביותר.

() $generation \leftarrow generation + 1$.

[7] מודל שהגיע בדיוק ל־ε יקבל ניקוד של 1000 לאחר נרמול.

6 סקירת תכנה – אבולוציה מסדר שני למציאת GA מיטבי לבעיית הפונקציה

בפרק זה נסקור בקצרה את התכנה למימוש האבולוציה מסדר שני עבור מציאת GA מיטבי לבעיית הפונקציה. פעם נוספת, הפירוט המובא כאן איננו כולל את כל מרכיבי המערכת ומתמקד אך ורק באלו החיוניים להבנת התהליך האבולוציוני. ניתן לבחון את המימוש המלא ב־Git: `https://github.com/giladh11/BestParamsProjectGA.git` תחת ה־higherLevelGA packages ו־evolutionGaTools.

- TestFunctions - תפקידה של מחלקה זו היא לספק פונקציות אקראיות ממשפחות שונות של פונקציות. השתמשנו במחלקה זו בכדי ליצור קבוצה O של קופסאות שחורות המשמשת מופע לבעיית מציאת GA מיטבי באבולוציה מסדר שני.
  - מתודות:
    - $polynomials\,(degree, count)$ - מתודה היוצרת רשימה של $count$ פולינומים אקראיים ממעלה $degree \geq$.
    - $exponents\,(length, count)$ - מתודה היוצרת רשימה של $count$ פונקציות מעריכיות אקראיות באורך $length \geq$.
    - $trigonometricFunctions\,(length, count)$ - מתודה היוצרת רשימה של $count$ פונקציות טריגונומטריות אקראיות באורך $length \geq$.
    - $getTestFunctions\,()$ - להשלים.
- ParamGA - מחלקה זו מייצגת את הפרמטרים שעומדים לאבולוציה מסדר שני. כפי שראינו בסקירה הקודמת, מופע של המחלקה מוחזק כשדה ב־SymRegSolverChromosome. ה־SymRegSolverChromosome משתמש בפרמטרים אלו כדי למצוא פתרון עבור האבולוציה מסדר ראשון לבעיית הפונקציה.
  - שדות:
    - $populationSize$ - גודל האוכלוסיה.
    - $pCrossover$ - ההסתברות לביצוע זיווג.
    - $pMutation$ - ההסתברות לביצוע מוטציה.
    - $dataSetSize$ - גודל קבוצת הנקודות אל מולה נבחנים המודלים.
    - $maxInitialDepth$ - העומק המירבי של פרטים בדור הראשון של האוכלוסיה.
  - מתודות:
    - $getRandomParamGA\,()$ - מחוללת אובייקט $\text{ParamGA} = (P, p_c p_m, |D|, d)$ עם ערכים אקראיים חוקיים. משמשת ליצירת האוכלוסיה הראשונית של פרטי SymRegSolverChromosome עבור האבולוציה מסדר שני.

- SymRegSolverChromosome - מחלקה זו מייצגת פרט באוכלוסיה עבור האבולוציה מסדר שני. ככזו, היא מכילה את האופרטורים האבולוציוניים $crossover, mutate$ שתוארו בפרק הקודם. לפירוט נוסף ראו את תיאור המחלקה בסקירת התכנה הקודמת (פרק 5.2).

- Effort - מחלקה זו מייצגת את המאמץ של SymRegSolverChromosome שהושקע כדי למצוא את כל אחד מן המודלים לכל אחת מן הקופסאות השחורות שניתנו לו. נדגיש כי בחרנו להביא בחשבון גורמים נוספים בחישוב המאמץ מעבר למספר הדורות. בחישוב ה־hFitness של SymRegSolverChromosome, פרמטר זה משוקלל כפי שתואר בפרק הקודם.

  - שדות:

    - $generationNum$ - מספר הדורות עד שנמצא המודל.
    - $numOfCrossover$ - מספר פעולות הזיווג שנעשו בתהליך מציאת המודל. שדה זה משקף הרעיון שככל שמתבצעות יותר פעולות זיווג על עצים כך התהליך האבולוציוני יקר יותר.
    - $numOfMutations$ - מספר פעולות המוטציה שנעשו בתהליך מציאת המודל. הרעיון מאחורי שדה זה דומה לרעיון מאחורי השדה הקודם.
    - $numOfPointsEvaluated$ - מספר הנקודות ששוערכו בתהליך מציאת המודל. כזכור, כל פרט באוכלוסיה עבור האבולוציה מסדר ראשון נמדד בהתאמתו ע"י מרחקו מקבוצת הנקודות D. שדה זה משקף את הרעיון שתהליך אבולוציוני בו משערכים מספר גדול של נקודות כבד יותר.
    - $sumOfTreesSizesCreated$ - סכום גדלי העצים שנוצרו במהלך האבולוציה. משקף את הרעיון שככל שעצים גדולים יותר, כך יקר יותר לשערך אותם.

  - מתודות:

    - $calculateEffort()$ - מתודה המשקללת את המאמץ ע"י השדות, כפי שהוגדר בפרק הקודם (פרק 5.4).

- HigherLevelEngine - מחלקה זו היא המנוע המחולל את האבולוציה מסדר שני על פרטים מסוג SymRegSolverChromosome.

  - שדות:

    - $listOfBlackboxes$ - רשימה של אובייקטים מסוג BlackBoxTree המייצגת מופע לאבולוציה מסדר שני. כל פרט באוכלוסיה נמדד בהתאם למודלים שמצא עבור הקופסאות השחורות ובהתחשב במאמץ שהשקיע בכך.
    - $populationSize$ - גודל האוכלוסיה עבור האבולוציה מסדר שני.

  - מתודות:

- $createPopulation\,(popluationSize)$ - מתודה זו יוצרת אוכלוסיה ראשונית ע"י הגרלה של מופעי SymRegSolverChromosome הנבדלים זה מזה בפרמטרים האבולוציוניים שלהם כפי שהתקבלו מ- $getRandomParamGA\,()$.
- $evolve\,(GENERATIONS)$ - המתודה אשר מחוללת את התהליך האבולוציוני כפי שתואר בפרק הקודם (פרק 5.5).
- $getBestSymRegSolver\,()$ - המתודה מחזירה את המופע המיטבי של SymRegSolverChromosome שנמצא בסוף התהליך האבולוציוני.

7 ממצאים ומסקנות(לא גמור)

בחלק זה נתאר את תהליך העבודה שביצענו בנסיון לענות על שאלות המחקר. לפני הרצת האבולוציה מסדר שני, רצינו לבחון כיצד מופעים שונים של GA מתמודדים עם פונקציות ממשפחות שונות. לשם כך, כתבנו סביבת הרצה עבור כל אחת מרמות האבולוציה וביצענו עליהן ניסויים. נתאר את הסביבות ואת הניסויים שביצענו.

7.1 RunLowerLevel - סביבת ההרצה עבור האבולוציה מסדר ראשון

סביבה זו תומכת בפעולות הבאות[8]:

(א) יצירת Setups שונים: Setup הוא צירוף של ParamGA וקופסה שחורה, עליה ניתן להריץ אבולוציה מסדר ראשון.

(ב) setParamGA - מאפשרת לבחור ערכים עבור currentParamGA, על בסיסם ייבנו Setups.

(ג) setNewFunc - מאפשרת לבנות קופסה שחורה על בסיס פונקציה המיוצגת ע"י מחרוזת. לאחר פעולה זו, ייבנה Setup הכולל את הפונקציה החדשה ואת currentParamGA.

(ד) setTestFunctions - בונה עשרה Setups על בסיס currentParamGA ועשר פונקציות המבחן שהגדרנו בשיטה $getTestFunctions$ שבמחלקה TestFunctions (ראו תחילת פרק 6).

(ה) setFamily chosenName poly/exp/trigo 'x' 'y' - יוצרת y Setup-ים על בסיס currentParamGA ו-y פונקציות אקראיות השייכות למשפחה שנבחרה ממעלה או באורך x.

(ו) runAll x: מריצה את כל ה-Setups הנוכחיים בזכרון x פעמים.

(ז) printSetups: עבור כל Setup, יודפס מידע המשקלל את כל ההרצות שבוצעו על ה-Setup. לדוגמה: מרחק ממוצע/מירבי/מזערי, מאמץ ממוצע/מירבי/מזערי וכו'.

7.2 ניסויים - אבולוציה מסדר ראשון

כדי לקבל מושג ראשוני על ההשערות שהעלנו בשאלות המחקר, ערכנו מספר הרצות של מקרי מבחן. עבור כל מקרה מבחן, הגדרנו מספר עותקים זהים של ParamGA הנבדלים זה מזה אך ורק בערך של פרמטר אבולוציוני יחיד. רצינו לבדוק כיצד שינוי בערך של פרמטר אחד משפיע על תוצאות המהלך האבולוציוני. להלן הפירוט:

(א) שינוי בגודל D: דיוק של מודל באבולוציה מסדר ראשון נקבע ע"י מרחקו הממוצע מקבוצת הנקודות D, שגודלה מתקבל כפרמטר מ-ParamGA. לאחר האבולוציה מסדר ראשון מתקבל bestModel שדיוקו מחושב מחדש אל מול מספר גדול של נקודות (ראו תיאור המחלקה BestModelCandidate בפרק 4), ומהווה את הדיוק

[8] ראו תמונה 8.1 בנספח.

האובייקטיבי של המודל. שערנו שככל שגודלה של D במהלך האבולוציה יהיה גדול יותר כך דיוקו האובייקטיבי של המודל יגדל. עם זאת, השערה זו התבדתה כפי שניתן לראות מגרף 8.2 בנספח.

(ב) שינוי ב־p_m: לא זיהינו מגמה ברורה. כפי שניתן לראות מהדוגמה בגרף 8.3 בנספח, ערכים שונים של שיעור ה־p_m עבור שלוש פולינומים ממעלה ≥ 2 לא הצביעו על שינוי מגמתי בדיוק שהושג.

(ג) שינוי בגודל האוכלוסייה: כפי שניתן להתרשם מגרף 8.4 בנספח, עבור שלושה פולינומים ממעלה ≥ 2 לא זוהה שינוי בדיוקם בעקבות שינוי תואם בגודל האוכלוסייה.

7.3 RunHigherLevel ־ סביבת ההרצה עבור האבולוציה מסדר שני

סביבה זו תומכת בפעולות הבאות[9]:

(א) setTestFunctions ־ יוצרת HigherLevelSetup לאבולוציה מסדר שני הכולל קבוצת פונקציות המבחן שהוגדרו על ידנו ב־getTestFunction() שבמחלקה TestFunctions.

(ב) setFamily chosenName poly/exp/trigo 'x' 'y' ־ יוצרת HigherLevelSetup לאבולוציה מסדר שני הכולל את משפחת הפונקציות שנבחרה.

(ג) runAll ־ מריצה את כל ה־HigherLevelSetup־ים בזכרון.

(ד) printSetupsInfo ־ מדפיסה את המידע על כל HigherLevelSetup בזכרון. עבור כל HigherLevelSetup מודפסים הפרמטרים הטובים ביותר שנמצאו וערך ה־hFitness שלהם. כמו כן, ניתן גם לראות את המודל הטוב ביותר שפרמטרים אלו מניבים לכל קופסה שחורה, המאמץ שהושקע עבור אותו מודל, הדיוק של המודל וכן ערך ה־hFitnessElement שלו.

(ה) setParamGaSetup ־ מאפשרת לקבוע ערכים עבור ParamGA ולבדוק את ערך ה־hFitness שלהם על קבוצות שונות של קופסאות שחורות. בעזרת שיטה זו, בדקנו כיצד פרמטרים מתפקדים על משפחות שונות של פונקציות.

7.4 ניסויים ־ אבולוציה מסדר שני

בפרק זה נתאר את הניסויים שערכנו כדי לענות על שאלות המחקר.

- הרצת אבולוציה מסדר שני על 10 פולינומים מדרגה ≥ 2:
  - הגדרנו HigherLevelSetup עם 10 פולינומים מדרגה ≥ 2 והרצנו את האבולוציה מסדר שני עליו.

[9] ראו תמונה 8.5 בנספח.

○ כפי שניתן לראות, בכל דור של האבולוציה כל פרט SymRegSolverChromosome מורץ על כל אחד מהפולינומים. עבור כל פולינום, מתקבל ערך hFitnessElement והשקלול של כולם מגדיר את ה־hFitness של הפרט SymRegSolverChromosome:

```
$setFamily 10poly poly 2 10
runAll
 created 10 new blackBoxes in a list
$  Running Setup 10poly
       calculated hFitnessElement for 38*x^1 + 6| distanceWeight: 736.249710118198, effortWeight: 239.0
       calculated hFitnessElement for  + 47| distanceWeight: 701.3529630810922, effortWeight: 2.0
       calculated hFitnessElement for 45*x^1| distanceWeight: 784.9529588968807, effortWeight: 69.0
       calculated hFitnessElement for 18*x^1 - 25| distanceWeight: 520.9192680477414, effortWeight: 102.0
       calculated hFitnessElement for 19*x^1 - 36| distanceWeight: 570.1691865131551, effortWeight: 53.0
       calculated hFitnessElement for 38*x^2| distanceWeight: 4664.749820103212, effortWeight: 475.0
       calculated hFitnessElement for 19*x^2 + 23*x^1| distanceWeight: 3089.692958690449, effortWeight: 477.0
       calculated hFitnessElement for 11*x^2 + 43*x^1 + 14| distanceWeight: 132763.0263458314, effortWeight: 502.0
       calculated hFitnessElement for 27*x^2| distanceWeight: 4182.798801054979, effortWeight: 477.0
       calculated hFitnessElement for 15*x^1| distanceWeight: 110.72405664775431, effortWeight: 121.0
 calculated hFitness for
        ParamGA - populationSize: 20, pCrossover: 0.401311000375586, pMutation: 0.5718873318945531,
        dateSetSize: 30, maxInitialTreeDepth: 1, bloatPenaltyRate: 0.9641926348816532
   hFitness = 15064.163606898484
       calculated hFitnessElement for 38*x^1 + 6| distanceWeight: 1991.0262865500765, effortWeight: 96.0
       calculated hFitnessElement for  + 47| distanceWeight: 844.3429598209207, effortWeight: 0.0
       calculated hFitnessElement for 45*x^1| distanceWeight: 716.186591239051, effortWeight: 4.0
       calculated hFitnessElement for 18*x^1 - 25| distanceWeight: 957.4373576863877, effortWeight: 66.0
       calculated hFitnessElement for 19*x^1 - 36| distanceWeight: 361.7899618591842, effortWeight: 55.0
       calculated hFitnessElement for 38*x^2| distanceWeight: 34670.58954075871, effortWeight: 87.0
       calculated hFitnessElement for 19*x^2 + 23*x^1| distanceWeight: 8064126.94830606, effortWeight: 91.0
       calculated hFitnessElement for 11*x^2 + 43*x^1 + 14| distanceWeight: 1057080.3303377295, effortWeight: 87.0
       calculated hFitnessElement for 27*x^2| distanceWeight: 5.708089495265906E9, effortWeight: 80.0
       calculated hFitnessElement for 15*x^1| distanceWeight: 111.70962882830793, effortWeight: 18.0
 calculated hFitness for
        ParamGA - populationSize: 5, pCrossover: 0.2430024403059540 8, pMutation: 0.46465411052310845,
        dateSetSize: 37, maxInitialTreeDepth: 3, bloatPenaltyRate: 1.7654147634951178
   hFitness = 5.717250939626877E8

....................
```

○ כבר בדור הראשון ישנו פרט עם hFitness=4687~, כלומר פי $4\frac{1}{2}$ פחות מהדיוק הנדרש:

```
 calculated hFitness for ParamGA - populationSize: 13, pCrossover: 0.07354830652783084, pMutation: 0.7872
1.082164005417698
   hFitness = 17085.236262436345
     ***
iter = 0          hFitness = 4687.444666607192
param = ParamGA - populationSize: 29, pCrossover: 0.38085353384803333, pMutation: 0.7931209540808409,
dateSetSize: 34, maxInitialTreeDepth: 4, bloatPenaltyRate: 0.7927024235210136
      ***

       calculated hFitnessElement for 38*x^1 + 6| distanceWeight: 776.6861635383862, effortWeight: 148.0
       calculated hFitnessElement for  + 47| distanceWeight: 257.604789255339, effortWeight: 1.0
       calculated hFitnessElement for 45*x^1| distanceWeight: 559.9359119027127, effortWeight: 9.0
```

○ בדור השמיני אנחנו מגיעים לדיוק הנדרש, עם hFitness=1000:

```
 calculated hFitness for ParamGA - populationSize: 20, pCrossover: 0.9042705250914993, pMutation: 0.8702677
1.2808869139209949
   hFitness = 7541.002424934282

iter = 8          hFitness = 1017.1895403878932
param = ParamGA - populationSize: 29, pCrossover: 0.38085353384803333, pMutation: 0.7931209540808409,
dateSetSize: 32, maxInitialTreeDepth: 4, bloatPenaltyRate: 0.7484788604890102
      ***
       calculated hFitnessElement for 38*x^1 + 6| distanceWeight: 140.89258101411684, effortWeight: 134.0
       calculated hFitnessElement for  + 47| distanceWeight: 269.9224576630854, effortWeight: 6.0
       calculated hFitnessElement for 45*x^1| distanceWeight: 709.6384998292535, effortWeight: 22.0
```

○ לאחר הרצה זו, הגרלנו 10 פולינומים ממעלה ≥ 2 ובדקנו מהו ערך ה־hFitness עבור הפרמטרים שמצאנו ביחס לקבוצת הפולינומים החדשה:

```
SetupHigherLevel: name: 10_poly^2.setParams
    ParamGA: ParamGA - populationSize: 29, pCrossover: 0.0380853, pMutation: 0.79312,
   dateSetSize: 32, maxInitialTreeDepth: 4, bloatPenaltyRate: 0.0
     with hFitness: 10634.0372203382
```

- כפי שניתן לראות, דוגמה זו ממחישה כי על אף שפרמטרים אלו הניבו ערך hFitness טוב עבור קבוצת פולינומים אחת, הם תפקדו באופן רע למדי עבור קבוצת פולינומים אחרת. על אף ששערנו שקבוצת פרמטרים המתאימה למשפחת פונקציות אחת יכולה להתאים למשפחת פונקציות אחרת, דוגמה זו מראה כי אפילו עבור קבוצות פונקציות מאותה משפחה השערה זו נסדקת.
- זאת ועוד, בחינה של פרמטרים מיטביים עבור פולינומים עבור פונקציות ממשפחת הפונקציות הטריגונומטריות והמעריכיות גלתה פערי hFitness בסדרי גודל משמעותיים בהרבה.

7.5 סיכום

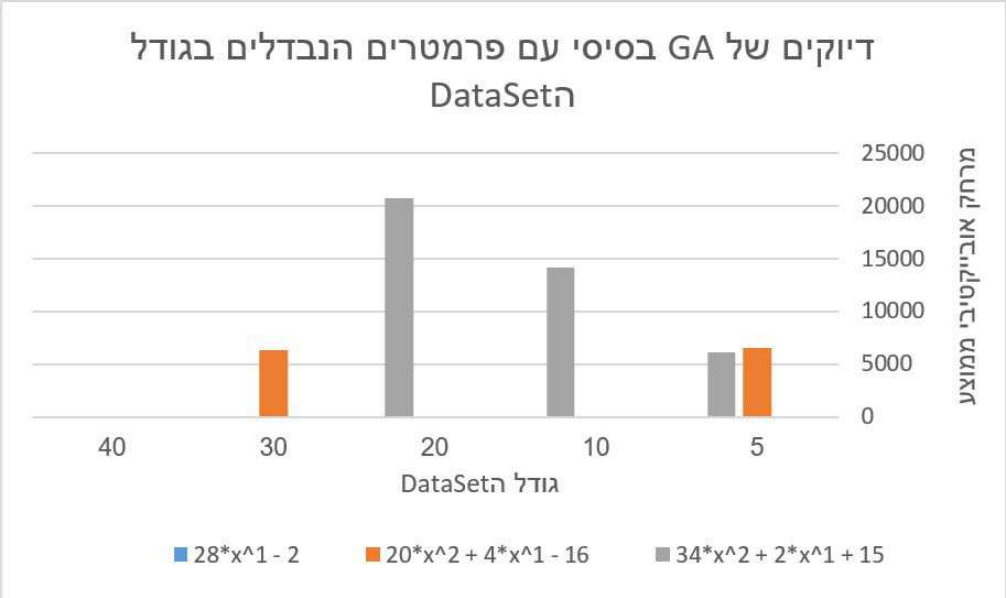
- להוסיף את השיקולים שלנו בבחירת ה־ $hFitness$. להסביר למה לא לקחנו פשוט את ה־$hFitness$ להיות ה־$lFitness$ ולנמק למה לקחנו בחשבון את כל ה"פעולות האבולוציוניות" שנעשו כדי להעיד על המורכבות של התהליך האבולוציוני.

8 נספח

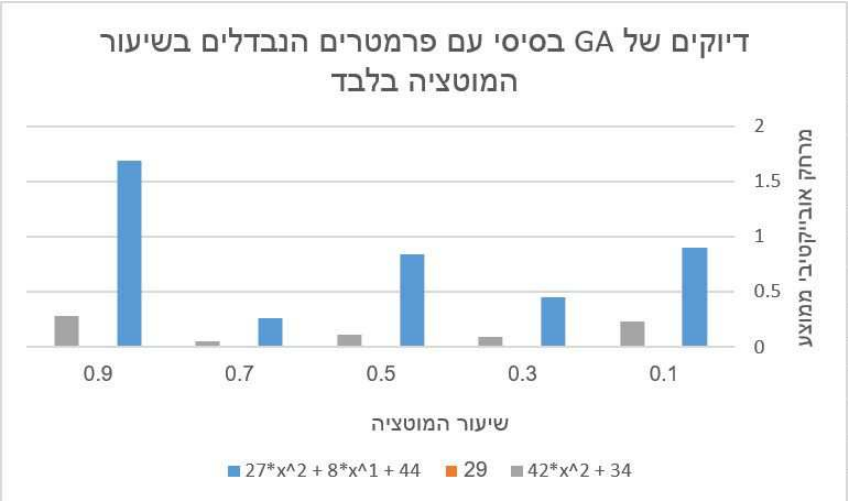
8.1 תמונה - תפריט סביבת ההרצה של האבולוציה מסדר ראשון

```
------- Test SymRegSolverChromosomes program  -------

Options:
random - create a random black box and run on it
setNewFunc 'FUNCTION_STRING' - will create a new BlackBox according to the requested string
setTestFunctions - will add the 10 diverse functions to the SetupList with the current ParamGA
setFamily poly/exp/trigo 'x' 'y' - will add 'y' functions from the family with degree/length x
copySetups x y - will copy all the setups with indexes between x and y (including) as new ones with the current p
run x - will rerun the last blackBox x times. when finished will print the results of the pervious runs and the a
runAll x - will make each memory setup run the same amount in total
printModels - will print all the models found by the runs on the currentBlackBox
printSetups - will print all the setups currently on memory
chooseSetup x - will change to the requested setup index
currentParamGA - will print the current ParamGA used
setParamGA  - to choose 6 new params for currentParamGA
currentSetup - will printthe current setup index, the blackbox and its averages
removeCurrentSetup - will remove the current setup from the memory(and reset the box
resetBlackBox - will reset the currentBox
help - will print this option menu again
quit - will exit the program

§
```

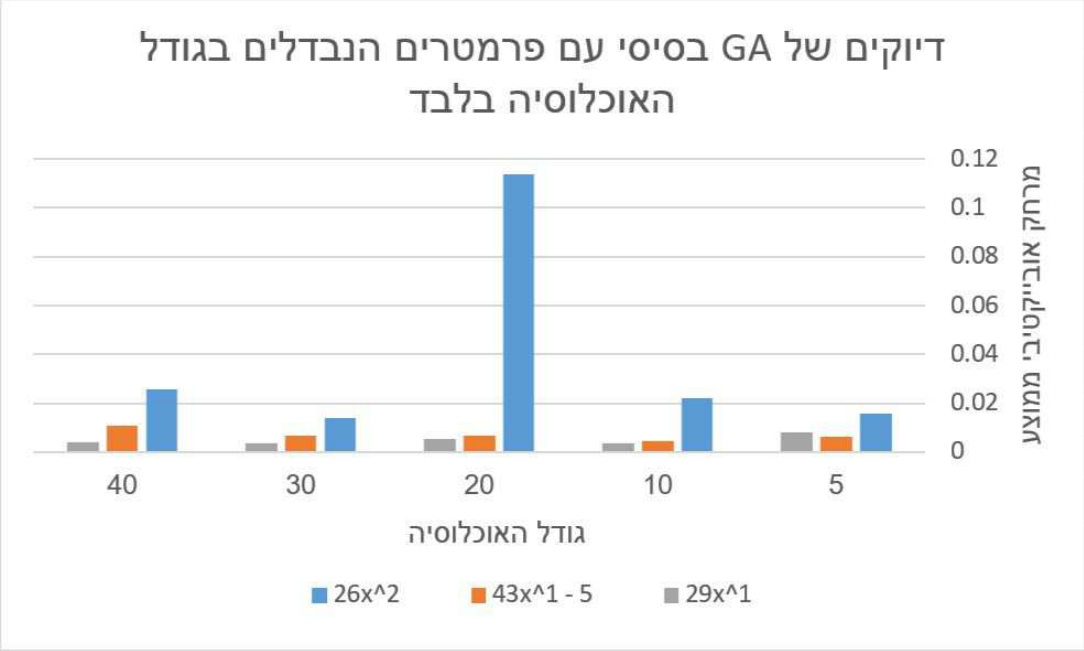
8.2 גרף - גודל D אל מול דיוק



8.3 גרף - שיעור p_m אל מול דיוק



8.4 גרף - גודל P אל מול דיוק



8.5 תמונה - תפריט סביבת ההרצה של האבולוציה מסדר שני

```
------- RunHigherLevel program  -------

Options:
default - will set all the default setups and run them
setTestFunctions chosenName  -  will add a list of 10 functions from the testfunctions, to a new setup chosenName in the list
setFamily chosenName poly/exp/trigo 'x' 'y' - will add a list of 'y' functions from the family with degree/length x, to a new setup chosenName in the list
runAll - will make sure all the current setups were run
printSetups - will print all the setups currently on memory
printSetupsInfo - will print all the setups currently on memory, and the black boxes and the bestModels each one of them found
printSetupsModels - will print all the setups currently on memory, and the bestModels each one of them found
printSetupsBoxes - will print all the setups currently on memory, and the black boxes
chooseSetup x - will change to the requested setup index
currentParamGA - will print the current ParamGA used
setParamGA  - to choose 6 new params for currentParamGA
setParamGaSetups x y - will recreate the setups with indexes between x and y (including) as new ones with the current paramGA
          //the idea is that it allows you to run different families on params we got in different runs
currentSetup - will printthe current setup index, and all its models
removeCurrentSetup - will remove the current setup from the memory(and reset the box
help - will print this option menu again
quit - will exit the program

$
```

8.6 קישור - תוצאות לאחר הרצת Setup באבולוציה מסדר שני

הפלט זמין בקישור הבא:

https://www.dropbox.com/s/d6mggthe2i2iit8/SetupHigherLevel_3poly3.rtf?dl=0

עבור ערכי ParamGA הנתונים, ניתן לראות לכל קופסה שחורה את המודל הכי טוב שהושג ואת המאמץ.

8.7 קישור - תוצאות לאחר הרצת Setup באבולוציה מסדר שני

הפלט זמין בקישור הבא:

https://www.dropbox.com/s/6mfiiayduv5eyel/SetupHigherLevel_5poly3.rtf?dl=0

עבור ערכי ParamGA הנתונים, ניתן לראות לכל קופסה שחורה את המודל הכי טוב שהושג ואת המאמץ.

8.8 קישור - תוצאות לאחר הרצת Setup באבולוציה מסדר שני

הפלט זמין בקישור הבא:

https://www.dropbox.com/s/lao9rmju63ar5a0/SetupHigherLevel_5trigo3.rtf?dl=0

עבור ערכי ParamGA הנתונים, ניתן לראות לכל קופסה שחורה את המודל הכי טוב שהושג ואת המאמץ.

9 מקורות מידע וסימוכין

- כל הקוד שנכתב במהלך וכן הדו"ח מצוי ב־Git:

  https://github.com/giladh11/BestParamsProjectGA.git

- מאמר המתאר פתרון של בעיית ה־Symbolic Regression באמצעות Genetic Programming ואת השיקולים השונים:

  http://web.eecs.utk.edu/~czhang24/projects/cs528_Project2_Zhang.pdf

- מאמר המתאר תהליך של Meta Genetic Programming על אופרטורי השינוי:

  http://cfpm.org/pub/papers/mgp.pdf

- עבור האבולוציה מסדר ראשון לפתירת בעיית הפונקציה, נעזרנו בקוד פתוח שפותר את בעיית ה־Symbolic Regression באמצעות GA:

  https://github.com/lagodiuk/genetic-programming