**פתרון בעיית ארבעת הצבעים באמצעות אלגוריתם גנטי**

בתרגיל זה השתמשנו באלגוריתם גנטי על מנת לפתור את בעיית ארבעת הצבעים.

את הבעיה הצגנו באמצעות מפת המקטעים השונים (1 עד 12), הגובלים אחד עם השני, כאשר הפתרון הסופי של הבעיה מופיע כמפה הצבועה בארבעת הצבעים.

בדומה לטבע, שבו בכל דור רק היצורים המתאימים שורדים, גם באלגוריתם גנטי רק הפתרונות הכי "מתאימים" יעברו תהליך של זיווג (Crossover).

בכתיבת האלגוריתם הגנטי מימשנו את העקרונות הבאים:

**זיווג (Crossover)**

מתבצע בתדירות קבועה התלויה בערך הפרמטר שיוכנס: הפרמטר Frequency of Crossover משמעותו שתהליך הזיווג יתבצע אחת ל- X דורות. אחד ההורים יוחלף (ימות) במפה החדשה שנוצרה (הצאצא) בזיווג, כך שגודל האוכלוסייה הכולל לא ישתנה.

**מוטציה (Mutation)**

מתבצע בכל דור, כאשר מקטע אקראי ישנה את צבעו בהתחשב בהסתברות שתוכנס: הפרמטר probability of Mutation (0 עד 1) יקבע האם המקטע יחליף את צבעו.

בכל מפה באוכלוסייה הכוללת ייבחר מקטע אקראי יחיד.

**שיערוך (Evaluation)**

זהו השלב שבו בוחרים את הפרטים שישתתפו בתהליך הזיווג לפי רמת ההתאמה.

בחרנו שתהליך הזיווג יתבצע בין 2 המפות הטובות ביותר ביחס לפונקציית ההערכה שמימשנו, כלומר עם כמה שפחות שכנים הצבועים בצבע "שאינו חוקי".

פונקציית ההערכה (Fitness function) רצה על כל המפות באוכלוסייה ובודקת את כמות המקטעים הלא חוקיים, כלומר מקטעים אשר אחד משכניהם בעל צבע זהה.

**מניעת התכנסות מוקדמת:**

על מנת למנוע התכנסות מוקדמת, הגרנו את המפות עם צבעים רנדומליים, ובנוסף כל מוטציה נעשתה ברנדומליות מוחלטת בלי התחשבות באליטיזם בקידום הפתרון. יתירה מזו את הקרוסאובר עשינו אך ורק למספר מוגבל ומועט של מפות מתוך אוכלוסיית המפות. כל הגורמים האלה נועדו למנוע התכנסות מוקדמת של פתרון הבעיה.

**הוראות להרצת התוכנית:**

**התוכנית כתובה בשפת Python 3, ומשתמשת בספריות tkinter ליצירת ה- GUI ובספריית matplotlib ליצירת גרפים.**

יש להריץ את מחלקת Main.py, בחלונית שתיפתח להזין את הפרמטרים הנדרשים, וללחוץ על Start. בסיום הריצה תופיע מפה אשר צבועה באופן חוקי, וכן גרפים המראים את השינויים **בציון הממוצע** ו**בציון הטוב ביותר** ל**אורך הדורות**.

מצורפת דוגמת הרצה בנספח א'.

**הבחנות:**

בהתבסס על התוכנית שכתבתנו, להלן ההבחנות שעל פיהן בחרנו את תמהיל האופרטורים שעבורו צירפנו את דוגמת ההרצה (נספח א').

ככל **שהסתברות המוטציה** **(probability of Mutation)** גבוהה יותר (מקס' 0.1), כך נגיע לפתרון חוקי בפחות דורות. לעומת זאת, בהסתברות מינימלית של 0.01 הפתרון משמעותית ארוך יותר. לכן בחרנו להתמקד בהסתברות של 0.05 כערך ממוצע בין השניים.

**תכיפות הזיווג (Frequency of Crossover)** לא משפיע על מס' הדורות למציאת פתרון חוקי, אך ערך קטן מדי מגדיל משמעותית את זמן ריצת התוכנית, והדבר גורם "לתקיעות" וחוויית משתמש פחות טובה.

בדומה ל**תכיפות הזיווג**, גם **גודל הדור (size of population)** משפיע על זמן הריצה של התוכנית אך לא משפיע על מס' הדורות הסופי הנדרש להגיע לפתרון.

נראה כי **הסתברות המוטציה** **(probability of Mutation)** הוא הפרמטר היחיד אשר משפיע על זמן מציאת הפתרון.

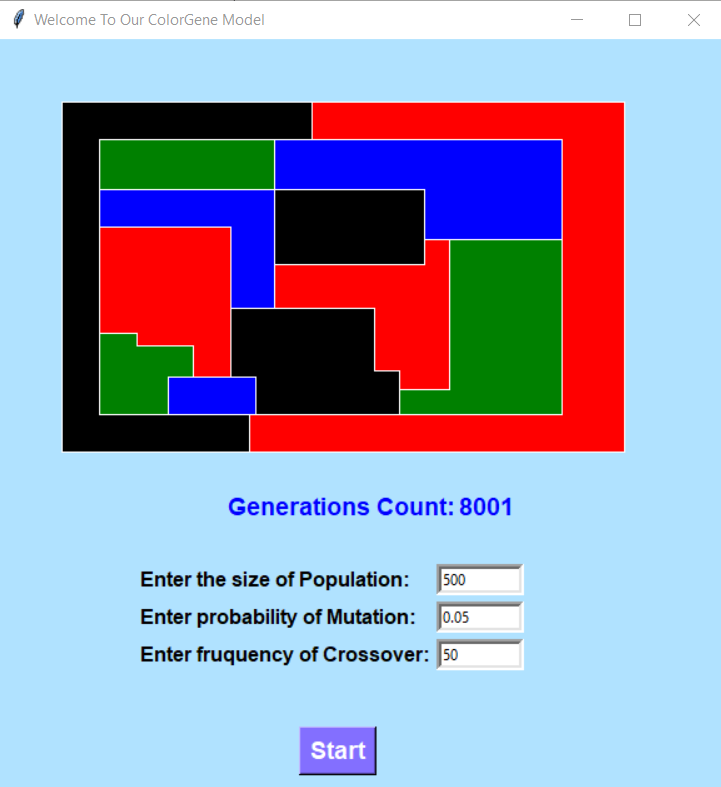
**נספח א':**

דוגמא להרצת התוכנית עבור ערכי הפרמטרים הבאים:

**גודל הדור = 500**

**הסתברות המוטציה = 0.05**

**תכיפות הזיווג = 50**

****

מגישים:

**בוריס אילצ'נקוב, ת.ז 323546606**

**יאיר כהן, ת.ז 318571718**