

By: Eden Dayan and Maya Byle

תוכן עניינים:

3 מבוא
4 הגדרת הבעיה
6 תיאור הפתרון
8 התוכנה
10 תוצאות ריצה
14 מסקנות

מבוא:

אלגוריתם אבולוציוני:

אלגוריתמים אבולוציוניים מתבססים על הרעיון של התפתחות אבולוציה. העיקרון של אלגוריתם זה עובד לפי העיקרון האבולוציוני - "החזק שורד". כל דור שנוצר טוב יותר מהקודם שלו: חלק מהפתרונות ייבחרו לרבייה (בדרך כלל המוצלחים יותר). לפעמים, בהסתברות מסוימת, יוכנס שינוי קל (מוטציה) בפתרון החדש. נקבל דור חדש של פתרונות הקרוב צעד נוסף לפתרון הבעיה הנתונה. האלגוריתם מקבל כקלט מופעים כלשהם, שהם קלט לבעיה המקורית. הפלט נבדק ע"י פונקצית התאמה (fitness) ואז מתבצעות מוטציות שונות בפתרון - מתודת הבחירה המבוססת על הערכת ההתאמה נבחרת, והאבולוציה מתחילה. נחזור על התהליך מספר רב של פעמים (דורות) ולבסוף נגיע לפתרון הקרוב ביותר.

פאזל צינורות:

פאזל צינורות הוא משחק בו יש לוח ועל כל משבצת יש צינור בצורה מסוימת. המטרה: לסובב את הצינורות ולפתור את הפאזל כך שכולם יהיו מחוברים ויכולו להעביר ביניהם מים ללא דליפות.

תיאור המשחק:

המצב ההתחלתי של המשחק הוא לוח ריבועי המחולק למשבצות. בכל משבצת יש צינור באחת מהצורות הבאות: L, T, I, i שנמצא ברוטציה מסוימת. המשתמש צריך לסובב את הצינורות (כל צינור סביב עצמו במשבצת בה הוא נמצא) על מנת למצוא את הפתרון האופטימלי ולחבר את כל הצינורות באופן מושלם. המטרה של המשתמש היא להצליח לעשות זאת בכמה שפחות צעדים \ זמן.

הגדרת הבעיה:

מימוש אלגוריתם אבולוציוני שיודע לפתור פאזל צינורות.

המטרה -

למצוא פתרון כמה שיותר טוב לפאזל בכמה שפחות זמן / איטרציות.
מתחילים באוכלוסייה של פתרונות אקראיים ובכל שלב של האלגוריתם יוצרים דור חדש.
האלגוריתם דואג שהדור החדש יהיה טוב יותר מקודמו תוך העדפה של פתרונות העונים על דרישות הפאזל.

מאפייני המשחק -

- בכל ריצה של האלגוריתם המשתמש יכול לבחור את ההגדרות בהתאם לרצונו:
- גודל לוח (4,5,7,10)
- גודל האוכלוסייה
- אליטיזם ריז: אחוז הפרטים שעוברים לדור הבא כמו שהם
- כמות דורות מקסימלית
- האם להתייחס לדרך (למספר הצעדים) או רק לפתרונות סופיים, אם כן, ניתן לבחור מקסימום לחיצות לכל צינור (טווח צעדים).
- כן ניתן לראות איך הפרמטרים השונים משפיעים על זמן הריצה, מספר הדורות הנדרשים לפתרון, וציון fitness הסופי.

מצבים -


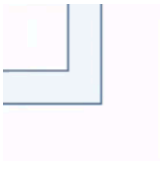
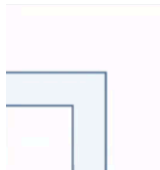

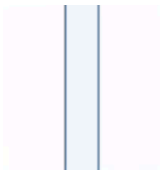

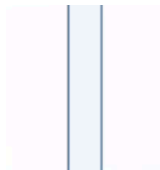

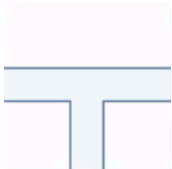

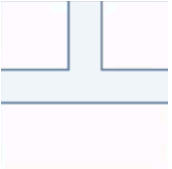





למשחק שלנו שני מצבים:

- פתרון סופי בלבד: האוכלוסייה מורכבת מפרטים המהווים וקטורים המייצגים פתרון סופי (וקטורים של מספרים המבטאים את הרוטציה הסופית של כל איבר על הלוח).
- כולל דרך: האוכלוסייה מורכבת מפרטים המהווים וקטורים המייצגים מספר צעדים שבוצעו על כל איבר בלוח (וקטורים של מספרים כמות הסיבובים שבוצע על איבר מסוים עד להגעתו לפתרון המיוצג בוקטור)

האובייקטים - הצינורות:

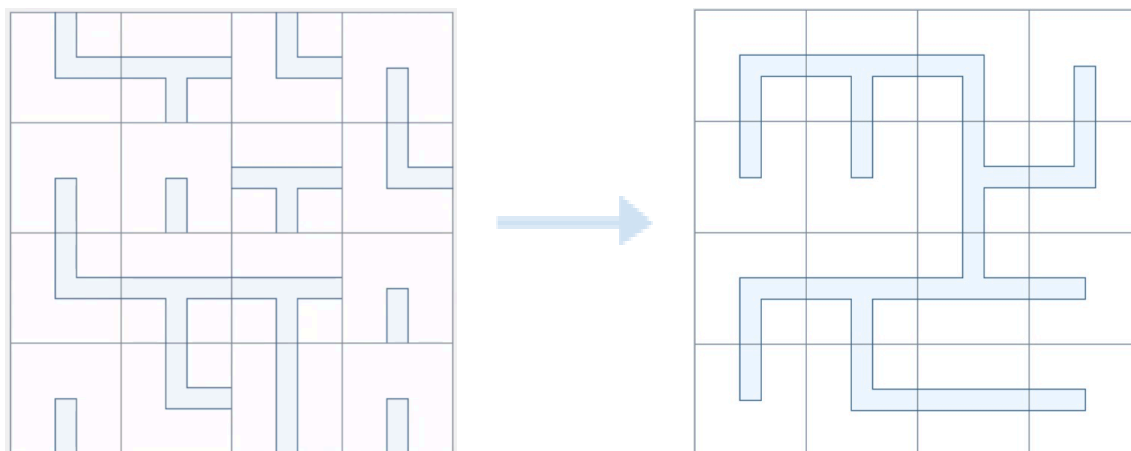


ייצוג הצינורות:

	0	1	2	3
L				
I				
T				
i				

תיאור הפתרון - אלגוריתם אבולוציוני

אלגוריתם אבולוציוני מורכב מכמה מחזורים של יצירת דורות שאנו רוצים שילכו שיתפרו.



מבנה הפרט

- כל פרט באוכלוסייה מורכב מוקטור באורך n^2 (מייצג את גודל הלוח)
- פתרון סופי בלבד: הוקטור מורכב ממספרים בין 0 ל-3 המתארים את מצב הצינור הסופי במשבצת מסוימת.
- כולל דרך: הוקטור מורכב ממספרים המתארים את כמות הסיבובים שבוצעו על המשבצת הנוכחית עד למצבה הסופי.
- את הוקטור קוראים משמאל לימין.
- האיבר ה- i שלו מייצג את המשבצת $[i \% n]$ $[i / n]$ באשר x אחז זה גודל הלוח.

אתחול האוכלוסייה

נאתחל m (גודל האוכלוסייה שנבחר) וקטורים באורך n^2 (גודל הלוח) באופן אקראי.

חישוב הפיטנס

פונקציית הפיטנס של פרט באוכלוסייה תלוי באיכות הפתרון שאותו הוא מייצג; האם הצליח להגיע לפתרון סופי או לא, כמה קבוצות מקושרות יש לו וגודל של כל קבוצה, מספר הצינורות שלו היוצאות מגבולות הלוח. ובמידה והאלגוריתם מופעל במצב הכולל דרך ($isMoves=True$), נספור כמה צעדים נעשו בדרך. כל אלו יחד נותנים לאלג' את ההערכה של כמה טוב הוקטור הנבחר. ההשפעות על הפיטנס:

גורמים שמעלים את הצינור הסופי:	גורמים שמורידים את הצינור הסופי:
צינור תואם לפתרון הסופי $\leftarrow +$ נקודה	צינור שלא תואם לפתרון הסופי $\leftarrow -$ נקודה
גודל הקבוצה המקושרת הכי גדולה $\leftarrow +$ גודל הקבוצה	צינור שיוצא מגבולות הלוח: $\leftarrow -$ חצי נקודה
	במצב כולל דרך, מספר צעדים גדול מהפתרון הסופי $\leftarrow -$ הפרש הצעדים/טווח הצעדים

* לאחר מכן נרמלנו את ציון הפיטנס לתחום $[0,10]$ ולכן הפיטנס האידיאלי $= 10$.

זהו הציון האופטימלי שפרט באוכלוסייה יכול להגיע אליו.

יצירת אוכלוסייה חדשה

- λ אחוז מהפרטים של תת האוכלוסייה יישמרו כאלטות לדור הבא (אליטיזם רייט).
- נבצע crossover על שני וקטורים באוכלוסייה בהסתברות 0.9.
- על כל אחד מהוקטורים באוכלוסייה, אנחנו מבצעים מוטציה בהסתברות של 0.2.

התוכנה:

ספרייה מרכזית: eckity
כדי להריץ את האלגוריתם האבולוציוני ולחשב את הריצה הכי טובה האפשרית של הבעיה.
המחלקות משתמשות \ יורשות וממשות את eckity.

פירוט של המחלקות שכתבנו:

[BoardCreator.py](#)

מטפל ביצירת האינדיבידואלים באוכלוסייה.

[BoardEvaluator.py](#)

משערך את איכות פרט באוכלוסייה על סמך התאמה\קרבה לפתרון אופטימלי, בדיקת גדלי קבוצות מקושרות ועוד.

[BoardEvolution.py](#)

האובייקט שמנהל את האלגוריתם האבולוציוני.

[BoardIndividual.py](#)

מתאר פרט באוכלוסייה, המייצג לוח במצב מסוים.

[Shape.py](#)

אובייקט של צינור בלוח.

[ShapesDFS.py](#)

מחלקה שנועדה לבצע חישובים ובדיקות על הפרטים באוכלוסייה במטרה לבדוק את איכותם.

[UI.py](#)

הממשק של בחירת הפרמטרים. בנינו ממשק לבחירת פרמטרי-ריצה שונים אותם המשתמש יכול להגדיר ולראות את התוצאות בהתאם לפרמטרים שבחר.

[Boards.py](#)

מידע סטטי על לוחות המשחק האפשריים, כולל את צורות הצינורות לכל גודל לוח, הפתרון האופטימלי אליו נשווה את פתרון האלגוריתם בחישוב פונקציית הפיטנס ואת מצב הלוח הנוכחי בשביל הנראות בUI.

[main.py](#)

מקבל קלט מהמשתמש, מאתחל ומריץ את האלגוריתם בהתאם.

בנוסף יצרנו הצגה ויזואלית של המשחק, המאפשרת לשחק/ לראות את הפתרון הטוב ביותר של ריצת האלגוריתם הנוכחית.

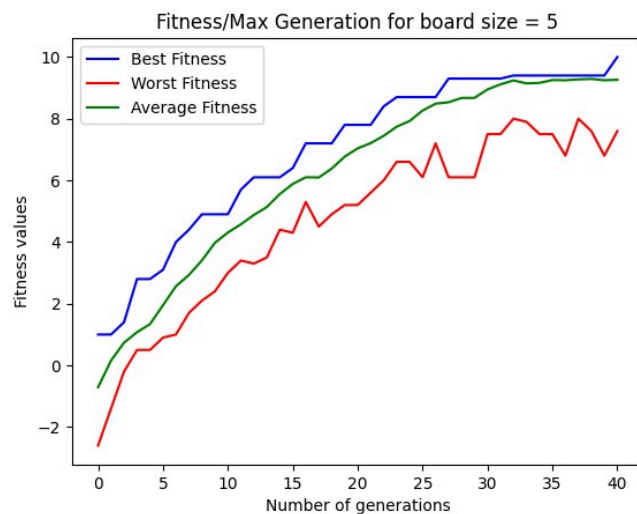
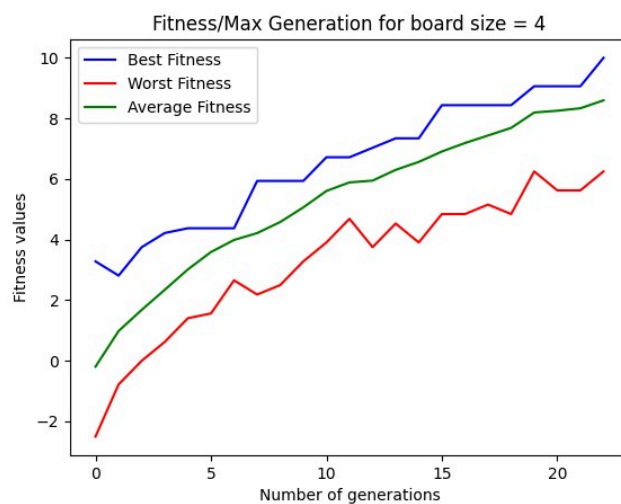
תוצאות ריצה וגרפים:

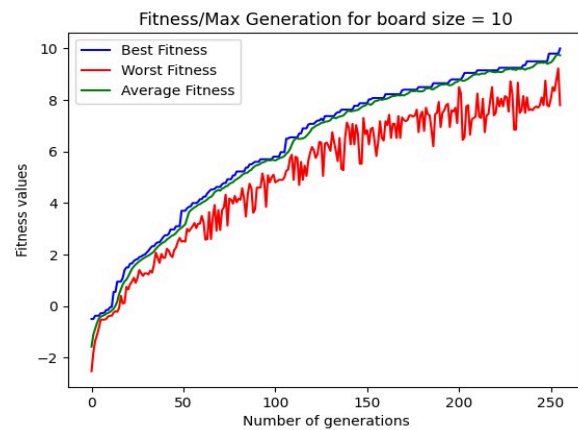
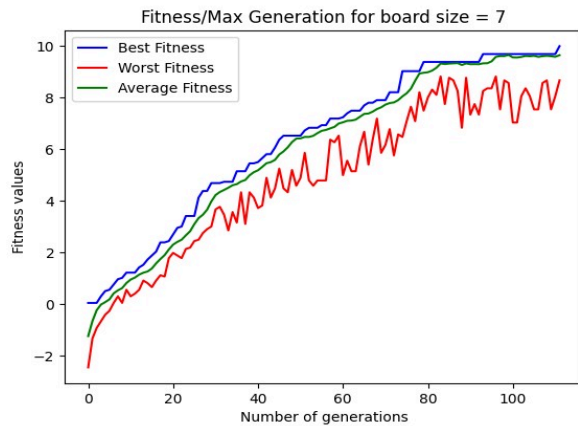
לאחר שסיימנו לכתוב את כל מחלקות האלגוריתם, ביצענו הרצות ובדיקות, עם פרמטרים שונים בכל פעם, בשביל להבין ולשפר את יכולות האלגוריתם. הערה: כדי לשפר את אמינות הניסויים - הרצנו כל בדיקה 30 פעמים והגרפים המוצגים מורכבים מהמוצעים של אותן ריצות.

Board size .1

בתור התחלה בחנו את השפעת גודל הלוח על מספר הדורות הנדרש עבור התכנסות הפתרון. רצינו להבין עד כמה השינוי בגודל משפיע על המהירות שבה האלגוריתם שלנו מוצא את הפתרון האופטימלי של הפאזל. הרצנו את האלגוריתם עם פרמטרים זהים עבור כל לוח כדי לראות את ההשפעה של הגודל בלבד ללא היסחים.

תוצאות:



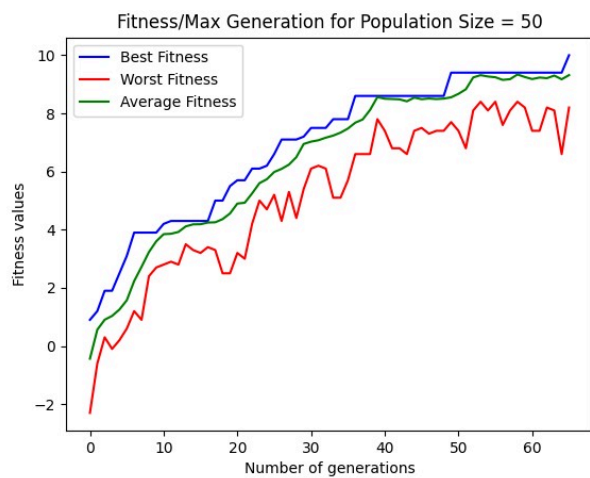
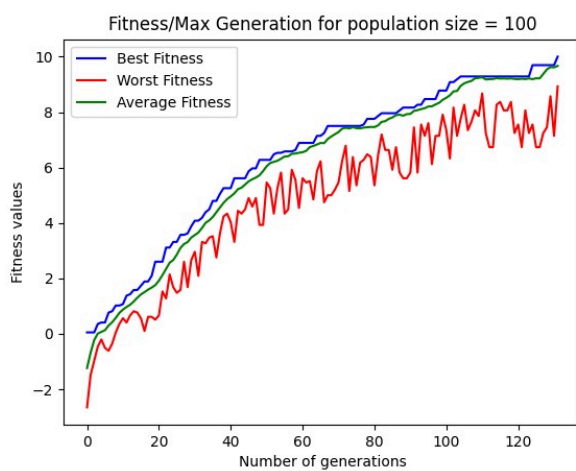


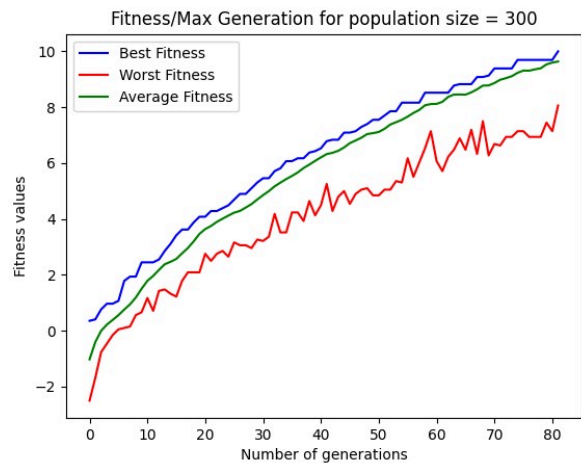
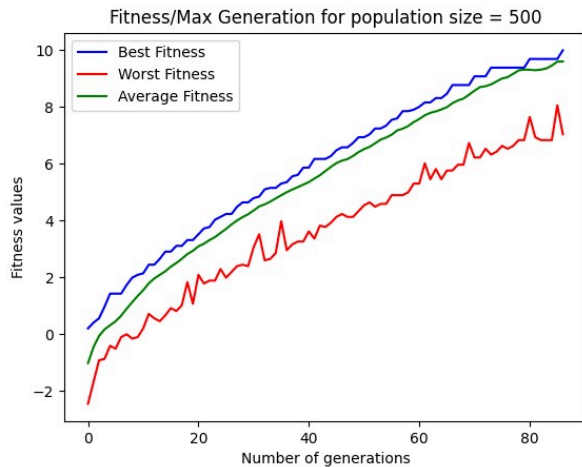
כפי שציפינו, מצאנו כי לאלגוריתם לוקח יותר זמן להתכנס ככל שגודל הפאזל גדל.

Population size .2

בנוסף, רצינו לבחון את השפעת גודל האוכלוסייה על התוצאות. הרצנו את האלגוריתם על פאזל בגודל קבוע של 7×7 עם גדלי אוכלוסייה משתנים: 100, 500, 300, 100.

תוצאות:



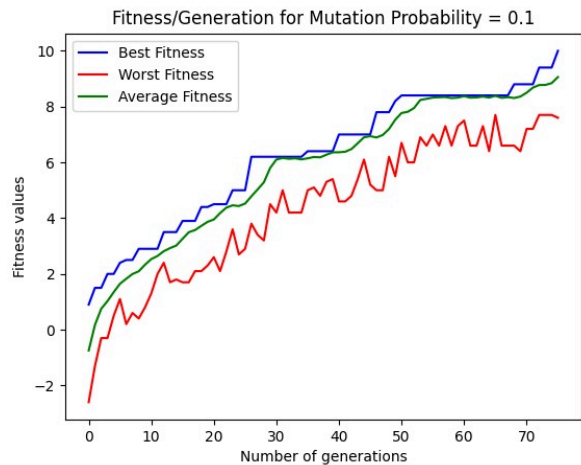
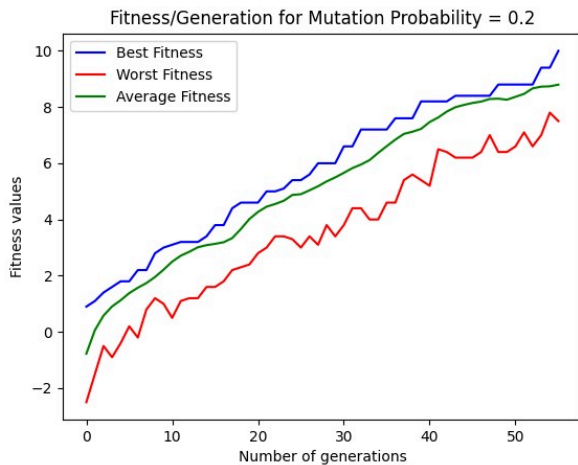


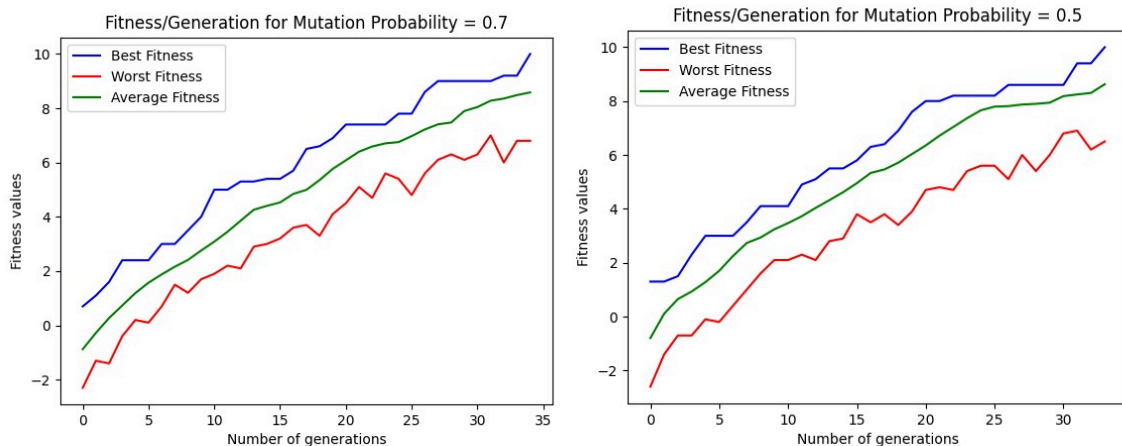
כלומר, ככל שגודל האוכלוסייה גדול יותר, כך האלגוריתם יכול ליצור מוטציות איכותיות יותר והוא יגיע לפתרון האופטימלי בפחות דורות. ככל שמסתכלים על מימדים גדולים יותר של אוכלוסיות - ההשפעה נהיית יותר מזערית.

Mutation Probability .3

בכל איטרציה של האלגוריתם אנו יוצרות דור חדש, כחלק מיצירת הדור אנחנו משתמשות בחישוב הסתברותי שקובע את הסיכוי לעשות מוטציה על וקטור יחיד. ביצענו ניסויים ובהם שינינו את ההסתברות לביצוע של מוטציה על פרט באוכלוסיה. בדקנו עבור אוכלוסיה בגודל 300 ופאזל בגודל 5×5 .

תוצאות:

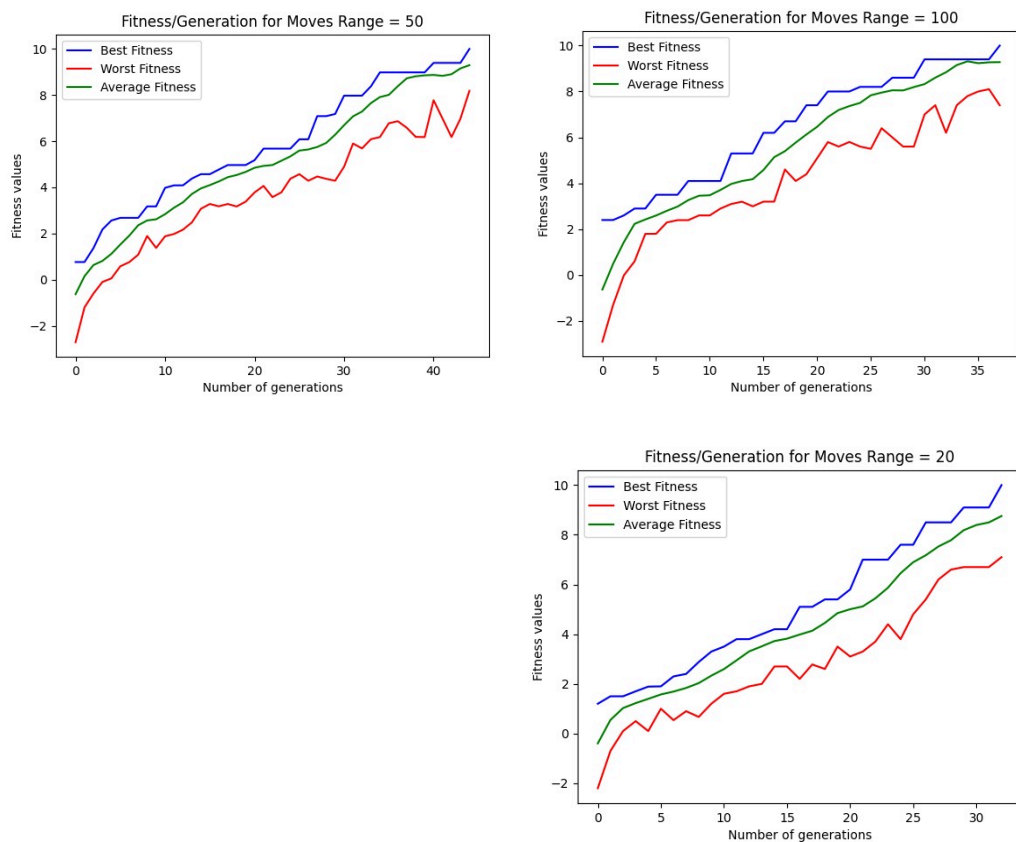




Moves range .4

לאלגוריתם שלנו יש שני מצבים, אוכלוסייה שמורכבת רק מפתרונות סופיים או אוכלוסייה של פתרונות עם התייחסות למספר הצעדים.
 בדקנו כיצד הטווח של מספר הצעדים ממנו האוכלוסייה מורכבת משפיע על התוצרים של האלגוריתם. בדקנו עבור אוכלוסייה בגודל 300 ופאזל בגודל 7×7 .

תוצאות:



מסקנות:

גודל הפאזל והאוכלוסייה:

מצאנו כי ככל שגודל הפאזל גדול יותר, כך צריך מספר דורות רב יותר על מנת להגיע לפתרון האופטימלי:

- פאזל בגודל 4: צריך בממוצע 25 דורות
- פאזל בגודל 5: צריך בממוצע 43 דורות
- פאזל בגודל 7: צריך בממוצע 115 דורות
- פאזל בגודל 10: צריך בממוצע 260 דורות

ביצענו עוד ניסויים וראינו כי יש קשר בין גודל הפאזל וגודל האוכלוסייה. בעצם הסקנו כי ככל שהפאזל גדול יותר - כך נרצה אוכלוסייה התחלתית גדולה יותר. זאת מכיוון שעבור פאזל גדול יותר יהיו יותר אפשרויות לפתרונות (לא בהכרח נכונים). האלגוריתם מתחיל מוקטורים רנדומליים כגודל האוכלוסייה, אם נסתכל על מספר מצומצם יחסית של וקטורים לעומת גודל הפאזל אנחנו בקלות עלולים לפספס פתרונות עם פוטנציאל גבוה וכך תהליך השדרוג של הפריטים ייקח יותר זמן.

Moves Range:

במהלך הניסיונות לבדיקת השפעת טווח הצעדים על תוצאות האלגוריתם ראינו תוצאה שהפתיעה אותנו. בניגוד לציפיות שהיו לנו, השינויים על טווח הצעדים ממנו נוצרת האוכלוסייה לא השפיעו באופן משמעותי על היכולת שלו למצוא פתרונות אופטימליים. ממצא זה הדגיש לנו את יכולת ההסתגלות של האלגוריתמים האבולוציוניים.

חישוב fitness:

בעת בניית האלגוריתם התלבטנו רבות כיצד נכון לחשב את הפיטנס. כחלק מהניסיונות שלנו התלבטנו כמה "עונש" וכמה "פרס" צריך לתת על מאפיינים של הפתרון. עבור אלגוריתמים שפועלים במצב שכולל מספר צעדים, רצינו לתת עונש גדול עבור כמות צעדים גדולה, שכן גם אם בסוף הפתרון הוא נכון הוא עדיין צריך להענש על כך שלקח לו הרבה מאוד צעדים להגיע. בהתחלה הורדנו את ההפרש בין כמה צעדים הפתרון עשה לכמה צעדים צריך לעשות בפועל אבל ראינו שההשפעה של שיטה זו על הציון גדולה מידי ואנחנו מפספסות וקטורים רבים שיכולים להוות פתרון טוב. לכן "נירמלנו" את ההשפעה ע"י כך שחילקנו את התוצאה ב $moves\ range$. בנוסף, רצינו לדמות את איכות הפתרונות לצורה בה האדם חושב ולכן החלטנו גם להעניש על צינורות שהחיבורים שלהם יוצאים משטח הפאזל מכיוון שזוהי פעולה ללא הגיון כאשר מנסים לפתור.

Mutation Probability:

באמצעות הניסויים, גילינו כי עבור $mutation\ probability = 0.5$ אנחנו מקבלות את התוצאות הטובות ביותר באלגוריתם האבולוציוני שלנו לפתרון הבעיה. זו הייתה בדיקה חשובה שלאחר הגילוי שלה השתפרו לנו התוצאות של כל הניסויים. זה מדגיש את חשיבות המוטציות בשמירה על המגוון הגנטי והשפעתן על איכות הפתרון.