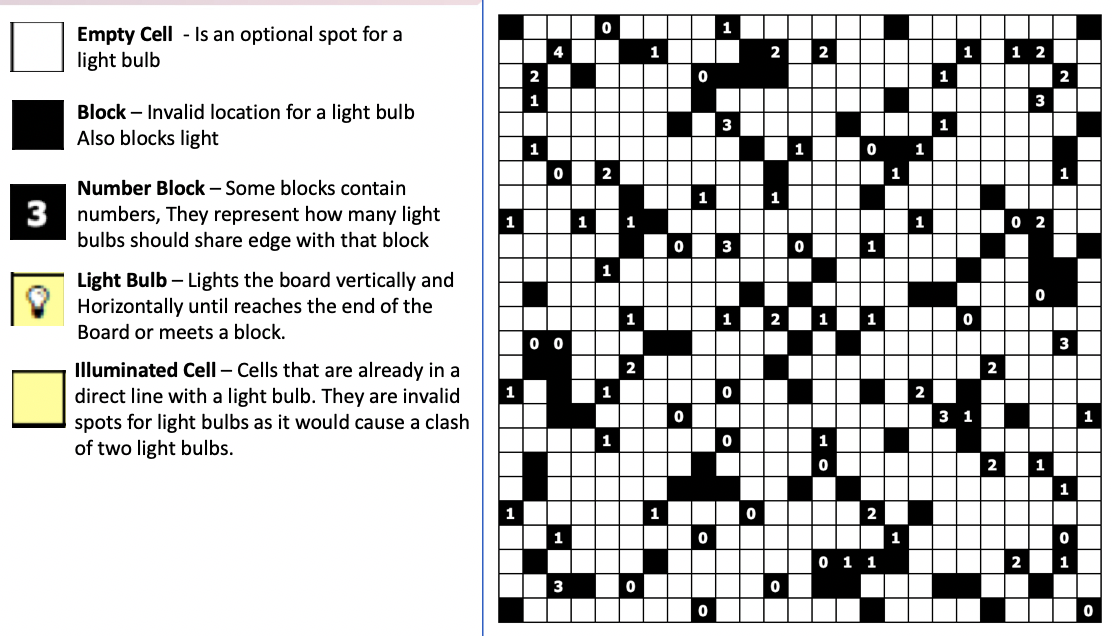
אלגוריתמים אבולוציונים – דו״ח מסכם

לירון אברהם 201553864 אדר עובדיה 305264202

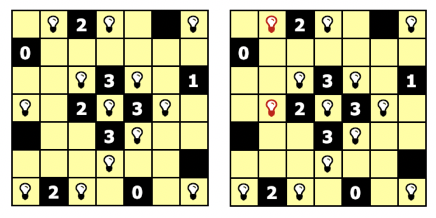
**הקדמה:**

בחרנו לבצע את הפרויקט בקורס על המשחק Lights Up Puzzle, למשחק זה מספר דרכי פתרון ביניהן אלגוריתמי חיפוש וחיפוש היוריסטי, אלגוריתמים אבולוציוניים ופתרונות היברידיים שמשלבים יתרונות משני הסוגים כשלמעשה מדובר בבעיית סיפוק אילוצים מורכבת. לוח המשחק בגודל NxN ומכיל משבצות ממספר סוגים, משבצות ריקות, בלוקים, בלוקים עם מספר, נורות ומשבצות מוארות. התנאים ההכרחיים לפתרון המשחק הם כדלהלן:

1. לכל בלוק עם מספר ישנן בדיוק אותו מספר של נורות צמודות אליו.
2. כל התאים מוארים.
3. אף נורה לא מאירה (מתנגשת) עם נורה אחרת.



**תרשים 1:** לוח המשחק וסוגי המשבצות השונים

****

**תרשים 2:** פתרון לא תקין (ימין) ופתרון תקין (שמאל) לאותו לוח משחק

**סקר ספרות:**

1. Np complete

ישנם מספר מחקרים קודמים שנערכו על המשחק והדרכים האפשריות לפתור אותו. מציאת פתרון ללוח המשחק נחשבת לבעיה np-שלמה כמו שציינו קנדל ושותפיו במחקר שערכו על משחקים np שלמים [1]. מרחב החיפוש של הבעיה הוא עצום ומכיל מספר רב מאוד של מצבים או פתרונות אפשריים (לא בהכרח תקינים), מרחב החיפוש של לוח כלשהו בגודל NxN תחום ב - . ניתן למעשה לבצע אופטימיזציות והקטנות למרחב החיפוש ולהגיע למרחב חיפוש בגודל -. *מדובר כמובן בהקטנה משמעותית של לוח החיפוש, אך בלוחות גדולים גם הצמצום הזה של מרחב החיפוש עשוי להשאיר מרחב חיפוש עצום בגודלו.*

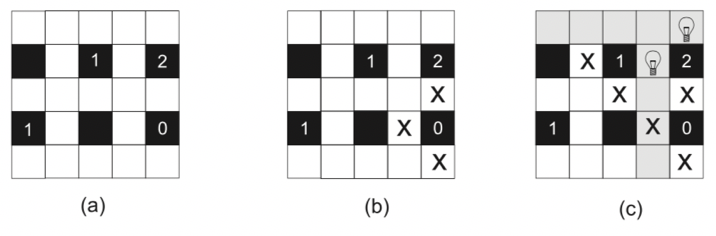
1. פתרונות באמצעות חיפוש ויוריסטיקות

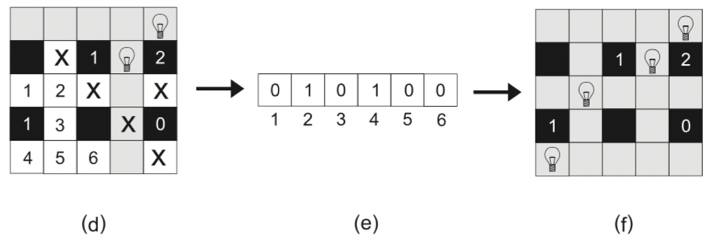
מחקרים רבים נעשו על מציאת פתרון למשחק באמצעות שיטות שונות של חיפוש, וחיפוש יוריסטי. צ׳יאו במחקרו, מציע אלגוריתם חיפוש מבוסס Elimination Search עם יוריסטיקות מבוססות תבניות, באמצעות שימוש ב – 4 תבניות נפוצות. תחילה השתמשו בתבניות המוכרות כדי למלא חלקים מהלוח, ולבסוף בחיפוש על מנת למלא את החורים.

1. פתרונות היברידיים

ישנם מספר מחקרים שעוסקים בפתרון המשחק בשיטות היברידיות, למשל שילוב של אלגוריתמים אבולוציוניים ורשתות נוירונים כמו שאורטיז-גרסיה הציע במחקרו [4].

סנצ׳ו ושותפיו מציעים דרך נוספת לפתרון המשחק במחקרם [2], תחילה נעשה עיבוד מקדמים ללוח המשחק, ומוצבות נורות אשר מיקומן ודאי כבר בשלב זה – למשל סביב בלוק עם המספר 4 (כלומר נורה בכל אחד מצלעותיו של הבלוק), או בלוקים עם המספר 3 שקיימים רק שלושה מיקומים לנורות סביבם (למשל צמודים לאחת מדפנות הלוח). בנוסף מסמנים ב -  X משבצות שאינם יכולות להכיל נורות (למשל כל המשבצות הצמודות לצלעותיו של בלוק 0). לאחר ביצוע העיבוד בצורה איטרטיבית ולאחר שלא נשארו נורות שמיקומן על הלוח ודאי, משתמשים במספר התאים החסרים לקביעת אורכו של הגנום בו ישתמש האלגוריתם האבולוציוני בשלב הבא. למשל, במקרה בו נשארו שישה תאים פנויים, יהיה הגנום באורך של 6 תאים, כך ש – 0 מסמן תא ריק ו – 1 מסמן נורה. לאחר ביצוע העיבוד המקדים נכנס לפעולה האלגוריתם הגנטי שמוסיף נורות רק ליד בלוקים שחסרים לידם נורות (למשל בלוק 3 שיש סביבו רק שתי נורות). לאחר השלב הגנטי נכנס שלב עיבוד מקדים נוסיף שבצורה דומה לעיבוד הראשוני מוריד את התאים שכעת לא יכולים להיות בהם נורות. לאחר של זה מתבצע שלב נוסף של אבולוציה שמחפש את הפתרון התקין של הלוח.





**הניסוי:**

1. מימוש – למעשה בחרנו לממש אלגוריתם עם עקרונות דומים לזה שהציעו סנצ׳ו ושותפיו [2], בחרנו לבצע עיבוד מקדים ללוח המשחק ובעזרת התאים שנשארו לקבוע את אורך הגנום. בנוסף שמרנו את מיפוי התאים החסרים על מנת להלביש את הגנומים חזרה על הלוח על מנת לבחון את הפנוטיפ. במאמר המוזכר לעיל מצאו פתרונות למספר לוחות 14x14 (בנוסף ללוחות קטנים יותר). אנחנו מעוניינים למצוא פתרונות גם ללוחות גדולים יותר (למשל 25x25(. תחילה בחרנו לממש את האלגוריתם באמצעות Heuristic Lab, פריימוורק מוכר על בסיס C# למימוש אלגוריתמים אבולוציוניים, אך לאחר מכן בין היתר עקב קשיים טכניים (שנינו עובדים על מערכות הפעלה שאינן וינדוס ו – Heuristic Lab לא עובד בצורה טובה בסביבות וירטואליות) בחרנו לממש את כל סביבת ההרצה בעצמנו באמצעות Node JS. הסיבה לבחירה ב - Node JS היא שידענו שאין חבילות שאנו צריכים ולא קיימות (לא השתמשנו בחבילות יעודיות כלשהן), ועל מנת לתרגל שפת תכנות ששנינו לא כתבנו בה זמן רב. מעבר לכך, אין סיבות טכניות לבחרה ב – Node JS לפרויקט זה.
2. לוחות – ידענו שנרצה להריץ את האלגוריתם על מספר מגוון ורחב של לוחות בגדלים שונים. לשם כך כתבנו סקריפט שלמעשה מעתיק לוחות מהאתר - <https://www.puzzle-light-up.com/> ומפרסר אותם לאובייקט איתו נוכל לעבוד, כאשר הלוח מיוצג על ידי מערך דו מימדי של מספרים שלמים ותאיו מיוצגים כלהלן:

**let** CellType = { "BLOCK":-1, //A Bock without a number  
 "ZERO\_LIGHTS":0, //Block with the number 0  
 "ONE\_LIGHT":1, //Block with the number 1  
 "TWO\_LIGHTS":2, //Block with the number 2  
 "THREE\_LIGHTS":3,//Block with the number 3  
 "FOUR\_LIGHTS":4, //Block with the number 4  
 "NO\_LIGHT":5, //Empty cell  
 "LIGHT\_BULB":6, //Light Bulb  
 "LIGHT":7, //Illuminated cell  
 "INVALID":8}; //Invalid bulb location (pre-processing only)

1. פיטנס

הערכת הפיטנס של האינדיבידואל מתבצעת על ידי ספירת התאים שאינם מוארים, בנוסף למספר המקומות בהם יש נורות עודפות (למשל 4 נורות סביב בלוק 3) ומספר המקומות שנורות חסרות בהן (למשל 2 נורות סביב בלוק 3). לאחר מכן הוספנו תכונה מחיפוש נובלטי למנגנון הפיטנס על ידי כך שביצענו ספריה של מספר האינדיבידואלים הזהים בדור ומתן ״קנס״ של פקטור החזרות של האינדיבידואל.

1. סלקציה

בהתחלה השתמשנו בסלקציה אליטיסטית, שמרנו אחוז מסוים מהאוכלוסייה וזרקנו את השאר. מהאינדיבידואלים ששרדו (ניסינו פרמטרים שונים מ – 10 אחוז ועד 80 אחוזים ביצענו קרוסאובר בצורה רנדומלית לחלוטין. בנוסף, ניסינו גם לשמור את האחוזון העליון ללא שינוי לדור הבא, או לבצע עליו מוטציות.

לאחר מכן, כשנתקענו בבעיית המינימום הלוקאלי, בחרנו לממש סלקציה שונה, למעשה כל אחד מה-90% האינדיבידואלים הטובים ביותר באוכלוסייה נבחר בצורה סדרתית כהורה א׳, לאחר מכן בוחרים את הורה ב׳ בצורה רנדומלית לחלוטין. לאחר ביצוע הקרוסאובר והמוטציות בודקים האם האינדיבידואל החדש טוב מההורה, אם הוא אכן טוב יותר שומרים אותו לדור הבא, ואם הורה א׳ יותר טוב אז מעבירים אותו לדור הבא. את ה-10% הנוספים שחסרים באוכלוסייה מרכיבים בצורה רנדומלית כמו האתחול הראשוני.

1. קרוסאובר

מימשנו מספר פונקציות קרוסאובר שונות, תחילה מימשנו קרוסאובר יוניפורמי שעבור בוחר בהסתברות של 50:50 מאיזה הורה יגיע כל תא בגנום של הצאצא. לאחר מכן הוספנו קרוסאובר 1 point שמחלק בצורה רנדומלית את האינדיבידואל ובוחר בצורה רנדומלית מאיזה הורה יגיע כל חלק. בנוסף, מימשנו קרוסאובר שעובד בצורה דומה לקודם אך למעשה מבצע את שתי האפשרויות ושומר את זו עם הפיטנס הנמוך ביניהן.

1. מוטציות

בחרנו לממש שתי מוטציות, מכיוון שמדובר למעשה במערך בינארי (0 או 1), בחרנו לממש מוטציה שבוחרת בצורה רנדומלית ביט אחד במערך והופכת אותו. בנוסף, בחרנו לממש מוטציה יותר אגרסיבית שעוברת על כל איברי המערך והופכת כל אחד מהם בהסתברות כלשהי.

1. פרמטרים

//The probability of a mutation to occur  
Configs.mutation\_probability = 0.3;  
  
//The probability for crossover  
Configs.crossover\_probability = 0.99;  
  
//The size of each generation  
Configs.generation\_size = 15000;  
  
//Percentage of individuals to survive to the next generation - elitism  
Configs.partGenerationToContinue = 0.20;  
  
//The number of generations to develop  
Configs.number\_of\_generations = 500;

1. תוצאות

בריצות הראשונות של האלגוריתם הפיטנס נעצר ב70-80 תלוי בגודל הלוח. לאחר בחינה של האינדיבידואלים ה״מוצלחים״ מצאנו שהחשש שלנו נכון ואנחנו באמת נתקעים בנקודה של מינימום לוקאלי. ניסינו לבצע שינויים רבים של שיטות הקרוסאובר והמוטציה, בנוסף לשינוי פרמטרים כמו הסיכוי למוטציה, גודל האוכלוסיה וכו׳. השינויים לא הראו שיפור כלשהי במהלך הריצות והפיטנס נשאר כשהיה. לאחר התייעצות עם צוות אחר שנתקע בבעיה דומה, בחרנו להוסיף מימוש של חיפוש נובלטי לגנטיקה כמו שהוצג במחקרים אחרים[5][6]. לשם כך בתחילה הוספנו ״ענישה״ לאינדיבידואלים זהים באותו דור, למעשה ספרנו כמה אינדיבידואלים זהים קיימים והכפלנו את הפיטנס של האינדיבידואלים האלה במספר הכפילויות. לאחר שינוי זה ראינו שיפור בפיטנס שירד עד 40, אך גם במקרה הזה נתקענו בנקודה של מינימום לוקאלי. לאחר מכן בחרנו לשמור מילון עם כניסה עבור כל גנום ייחודי, ביצענו ספירה של הופעה חוזרת של גנומים במהלך הריצות, ולאחר מכן בפונקציית הפיטנס, ״קנסנו״ את האינדיבידואלים בעלי הגנומים החוזרים בפקטור של מספר החזרות של הגנום.

1. גרפים

**מסקנות:**

**מקורות:**

[1] KENDALL, Graham; PARKES, Andrew; SPOERER, Kristian. A survey of NP-complete puzzles. *ICGA Journal*, 2008, 31.1: 13-34.

[2] SALCEDO-SANZ, Sancho, et al. A nested two-steps evolutionary algorithm for the Light-up puzzle. *Icga Journal*, 2009, 32.3: 131-139.‏

‏

[3] CHIU, Shih-Yuan, et al. A simple and rapid lights-up solver. In: *Technologies and Applications of Artificial Intelligence (TAAI), 2010 International Conference on*. IEEE, 2010. p. 440-443.‏

[4] ORTIZ-GARCIA, Emilio G., et al. A hybrid hopfield network-genetic algorithm approach for the lights-up puzzle. In: *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*. IEEE, 2007. p. 1403-1407.‏

[5] Moshe Sipper, Jason H. Moore, and Ryan J. Urbanowicz. Solution and Fitness Evolution (SAFE): Coevolving Solutions and Their Objective Functions. Institute for Biomedical Informatics, University of Pennsylvania, Philadelphia, PA 19104- 6021, USA and Department of Computer Science, Ben-Gurion University, Beer Sheva 84105, Israel.

[6] LEHMAN, Joel; STANLEY, Kenneth O. Exploiting open-endedness to solve problems through the search for novelty. In: *ALIFE*. 2008. p. 329-336.‏