מבוא לבינה מלאכותית
האוניברסיטה העברית

DIRTY PROJECTORS

פרויקט גמר בנושא רנדור תמונות לאובייקט תלת מימדי

מאת: הדס איזק, דורי רוזנברג, גיא קורנבליט, יעקב סנדרס

תקציר

הפרויקט עוסק בבניית אובייקט תלת מימדי, כך שבהינתן n-תמונות, כל תמונה נראית מזווית אחרת של האובייקט. האתגר במשימה הינו קירוב ההטלות של האובייקט לתמונות הקלט, כאשר הקושי לייצג את התמונות נאמנה הולך וגדל ככל שהתמונות שונות זו מזו.

ניגשנו לבעיה כבעיית חיפוש, ובחנו אלגוריתמים שונים כדי למצוא את המענה המיטבי. כמו כן, התמקדנו במקרה בסיס של ייצוג תלת מימדי של 3 תמונות בצבעי שחור לבן. ההשראה לפרויקט הינה אמנותית וטכנית, מאחר ועשויה לאפשר, בין היתר, מידול תלת מימדי אוטומטי של אובייקטים (כאשר תמונות הקלט זהות למשל).

ההשראה לפרויקט נבעה מהאמן האמריקאי מייקל מרפי[1], שמרכיב אובייקטים תלת מימדיים תלויים מחפצים ומצורות שונות כדי להציג תמונות שונות מזוויות שונות.

[1] https://www.perceptualart.com/

תוכן עניינים

מבוא - הגדרת הבעיה, תיאור מהלך המחקר ומתודולוגיה.

מתודולוגיה

מאפייני הבעיה - ביחס לפונקציית המטרה ושיטת האקספלורציה

ניתוח גישות אלגוריתמיות שונות לפתרון

א. חיפוש לוקאלי
ב. חיפוש אבולוציוני (אלגוריתם גנטי).
ג. בעיית אילוצים CSP.
ד. שיטות מתקדמות

השוואה בין הגישות השונות, ביצועים על גודל תמונה גדול יותר.

כיווני המשך

נספח - הסבר על כלים מתקדמים בעיבוד תמונה בהם נעשה שימוש.

מבוא

הגדרת הבעיה: בהינתן k תמונות, נבנה אובייקט תלת-מימדי כך שכל תמונת קלט מתגלה בזווית אחרת של האובייקט.

תיחום הפרויקט (scope) : 3 תמונות בגודל ריבועי $n \times n$, בצבעי שחור לבן. האובייקט מורכב מ-n^3 פיקסלים והאלגוריתמים מחליטים עבור כל פיקסל האם יהיה מלא או ריק. זוויות התצוגה של התמונות בתלת-מימד הן אורתוגונליות.

בחרנו בחיפוש כדי למדל ולפתור את הבעיה, מאחר ומרחב הפתרונות אקספוננציאלי (2^{n^3}) , ומפני ששיערנו כי נוכל לפתח עקרונות היוריסטיים שינחו חיפוש בצורה יעילה, בהתבסס על מאפייני הבעיה. בראייתנו, האתגר בבעיה נובע מהצורך לבצע ויתורים שונים על איכות הייצוג של כל תמונה במטרה לייצג ברמה טובה את כל תמונות קלט. הויתורים הנדרשים באו לידי ביטוי בהנמכת רזולוציה ואיבוד פרטים, תוך שימור הפרטים המרכזיים בייצוג של התמונה (קווי המתאר למשל, המעטפת וכו'). ככל שהשונות בין התמונות הלכה וגדלה, כך נראה שקשה יותר למצוא פשרה סבירה בין התמונות מבלי לאבד מידע משמעותי.

התמקדנו בשלוש גישות מרכזיות לפתרון -

א. חיפוש מקומי היוריסטי
ב. חיפוש אבולוציוני (גנטי)
ג. בעיית סיפוק אילוצים (CSP).

כל גישה דרשה בחינה מחודשת של אופן הייצוג של הבעיה (אילוצים, לעומת כרומוזומים ומרחב מצבים), ופעלה לנצל אלמנטים שונים של מרחב הבעיה כדי למצוא פתרון. במסגרת זאת, גישת CSP סיפקה מענה בינארי האם קיים פתרון אופטימלי לבעיה, בעוד שהחיפוש המקומי והגנטי פעלו לייצר פתרון מקרב מיטבי ככל האפשר.

במהלך העבודה, גילינו כי **מרחב הפתרונות עשיר במאפיינים שונים** של האובייקטים התלת מימדיים. למשל, גילינו כי עבור פונקציות מטרה שונות, והאופן בו התקדמנו במרחב החיפוש, הגענו לאובייקטים אשר ההטלות שלהם מייצגות נאמנה את תמונות הקלט, אך המאפיינים התלת מימדיים שלהם שונים (אובייקט גושני, קליפה, או צורות מנותקות). במילים אחרות, החיפוש הוביל לאיזורים אחרים במרחב הפתרונות בהם קיימות נקודות מינימום שונות.

לפיכך, תחת ההנחה ש**כיוון החיפוש** מוגדר על ידי פונקציית המטרה ואופן החקירה, שיערנו כי **אלגוריתמים שונים עשויים להוסיף מורכבויות נוספות לתוצאה.** ראשית, ההבדלים בין האלגוריתמים נובעים מאופן חיפוש שונה במרחב הפתרונות, ולכן האתגר היה למצוא מענה מהיר ואופטימלי ביחס למטרה, ואף לגלות מאפיינים ויזואליים נוספים לאובייקט התלת מימדי. **זהו העיקרון שהנחה את ניתוח ביצועי האלגוריתמים, ביחס לאופרטורים ופרמטרים שונים.**

למרות זאת, נדגיש כי עם התבהרות מרחב הבעיה העדפנו להמשיך ולחקור את מאפייניה ולנצל אותם כדי להשיג ביצועים יותר טובים ביחס לאלגוריתמיקה בסיסית ונאיבית, מאשר להעמיק באופטימיזציה משמעותית של השיטות האלגוריתמיות השונות (מחקר מעמיק של פרמטרים וסאב-רוטינות).

מתודולוגיה

במהלך המאמר נסקור את ביצועיהם של אלגוריתמים שונים ביחס לבעיה, אשר מוגדרת באמצעות פונקציות מטרה שונות אותן בחרנו למזער, וביחס לאופן השיטוט במרחב אשר מוגדר על ידי אופרטורים שונים בהם עושים האלגוריתמים שונים.

ננתח את המאפיינים המרכזיים שבאים לידי ביטוי בכל אלגוריתם ביחס לאופרטורים, ולא נתעמק במחקר היפר-פרמטרים של האופרטורים הללו. בבואנו לנתח את ביצועי האלגוריתמים נתבונן במאפיינים הבאים -

א. אופטימליות הפתרון ביחס לפונקציית המטרה (יש יותר מאחת).
ב. מאפיינים ויזואליים של הפתרון התלת-מימדי
ג. בחלק מהמקרים - נתייחס לזמן הריצה.

התהליך המחקרי מורכב משלושה שלבים -

א. בחירת פונקציית המטרה ועיקרון האקספלורציה למול מאפיין ויזואלי שנרצה להשיג.
ב. אופטימיזציה של כל אלגוריתם ביחס לאופרטורים בהם משתמש.
ג. השוואה בין האלגוריתמים השונים.

בשלב א', נבחן את השפעות פונקציית המטרה ואופן השיטוט במרחב ביחס לאלגוריתם נאיבי (baseline), ונחקור את התכונות השונות גם ביחס לסוגים שונים של קלטים. בסיום שלב זה, נגדיר לכל גישה ויזואלית קונפיגורציה - פונקציית מטרה ואופרטורים - שמכוונים את החיפוש לפתרון בעל מאפיינים ויזואליים רצויים.

בשלב ב', נגדיר קלט כמקרה בוחן קבוע (3 תמונות ממימד 40x40). עבור כל גישה אלגוריתמית (חיפוש מקומי וגנטי) נגדיר אלגוריתם נאיבי ונבחן אותו ביחס לקונפיגורציות השונות משלב א'. בכל שלב, נבצע אופטימיזציה של סוג אחד של אופרטור (יצירת שכנים, יצירת מצב התחלתי וכו') או פרמטר. **נניח הנחה מקלה של הפרדת משתנים.** למשל, נניח ששינוי לטובה בפרמטר של אופרטור עבור אלגוריתם אחד, יהיה שינוי לטובה גם עבור אלגוריתם אחר.

בשלב ג', נשווה בין האלגוריתמים "האופטימליים" שמצאנו ביחס לגישות הויזואליות השונות.

אתגרים מתודולוגיים:

א. **אקראיות** - מאחר ומרבית האופרטורים והאלגוריתמים משתמשים במרכיב נכבד של אקראיות, השוואה בין הרצה בודדת של כל קונפיגורציה מושפעת רבות ממרכיב האקראיות. אידיאלית, היינו רוצים להשוות ממוצעים של K ריצות תחת כל קונפיגורציה, אך מטעמי מגבלות זמן בחרנו להתעלם מסוגיה זו.
ב. **אפיון ההתקדמות** (ציר ה-X) - השוואה בין קונפיגורציות שונות לאותו אלגוריתם יתבצעו ביחס לכמות איטרציות. השוואה בין אלגוריתמים שונים לפי מספר איטרציות עשויה להיות בעייתית, מאחר ונקודות ההתכנסות של כל אלגוריתם עשויות להיות שונות (בדגש האלגוריתם הגנטי ביחס לאחרים).

לפיכך, **נשווה בין האלגוריתמים השונים בצורה איכותנית,** כאשר מכל אלגוריתם נבחר פתרון אופטימלי ביחס לנקודת ההתכנסות הרחוקה ביותר.

ג. **קריטריון להשוואה** - בחרנו להשוות בין איכות הפתרונות המוצעים על ידי האלגוריתמים השונים, כאשר הרצת כל אלגוריתם מתבצעת על רף שהגדרנו שרירותית (בשלב זה) על 10,000 איטרציות. יצוין כי אם אלגוריתם יציג התכנסות על פתרון בשלב מוקדם עד סוף הריצה, נוכל להסיק כי האלגוריתם מגיע לפתרון אופטימלי תוך פחות מ- 10k האיטרציות.

ד. **תמונות הקלט** - כדי לא ליצור תאימות יתר למקרה הבוחן (overfitting), נבחן את האלגוריתמים השונים בחלק ג׳ ביחס לקלטים שונים. בפועל, בלתי נמנע כי אופטימיזציה של האלגוריתמים השונים ביחס למקרה בוחן יחיד תביא ל-overfitting.

ה. **ככלל, נדרשת העמקה נוספת באפיון וניתוח האלגוריתמים והפרמטרים השונים**. במסגרת הפרויקט, נבחן כל פרמטר בנפרד על האלגוריתם הנאיבי ונשתמש בקונפיגורציות המיטביות ביחס לפתרון הנאיבי בשאר האלגוריתמים. בפועל, הסקה זו עשויה להיות לא מדויקת ואף שגויה, ונדרש ניתוח מעמיק של כל הקונפיגורציות האפשריות כדי להגיע לאלגוריתמים אופטימליים.

תהליך ההרצה (pipeline):

1. **עיבוד מקדים** - עם קבלת תמונות הקלט, התוכנה מבצעת פעולות עיבוד בסיסיות על התמונות כדי להתאימן להרצה במודלים שבחנו :
   - שינוי גודל התמונות (לפי בחירת המשתמש), והפיכתן לריבועיות.
   - קוונטיזציה - המרתן לתמונות בינאריות בעלות 2 צבעים בלבד.

2. **הרצה -** לאחר מכן, נבחרים האלגוריתמים להרצה ותהליך החיפוש מתבצע. בסופו, מתקבל פלט תלת מימדי של מערך Numpy בעל 3 מימדים, כאשר ערכיו בינאריים (0 ו-1).

3. **יצירת הפלט -**

א. כיוון שהתוכנה פועלת על מטריצה בגודל n^3, התמדנו לעבוד במימדים נמוכים יחסים (עד גודל 100 על 100 על 100). לפיכך הפלט של האלגוריתם הינו מפוקסל פעמים רבות. בכדי לייצר אובייקטים תלת מימדיים המתאימים להדפסת תלת מימד, לדוג׳, המרנו את התוצר לאובייקט mesh תלת מימדי, אותו החלקנו, בכדי להציגו באופן מרשים ולהתגבר על הרזולוציה הנמוכה יחסית בה הוא נוצר.

ב. כמו כן, בסיום ההרצה מודפסים גרפים שונים שמציגים את התקדמות האלגוריתם לאורך הריצה ביחס לפרמטרים שונים.

מאפייני הבעיה - ביחס לפונקציית המטרה ושיטת האקספלורציה

במהלך העבודה, גילינו כי **מרחב הפתרונות עשיר במאפיינים שונים** של האובייקטים התלת מימדיים. הבחנו כי במגוון האלגוריתמים שהשתמשנו, לפונקציית ה-loss אשר נבחרה ולפונקציה יוצרת השכנים (או המוטציות, במקרה הגנטי) היתה השפעה מכרעת בהשפעה על תוצאות האלגוריתם.

את התוצאות של האלגוריתם, בהינתן אוסף תמונות, ניתו להפריד בחלוקה הבאה:

- השמה תלת מימדית **מפוצלת** של נקודות מנותקות במרחב.
- השמה תלת מימדית **מחולקת** לקווים נפרדים במרחב.
- השמה תלת מימדית **גושנית מלאה**
- **כשלון** - צירוף מסויים של תמונות עלול להחזיר תוצאה לא מספקת לעין האנושית, שיראה ממרבית נקודות המבט כקשקוש אקראי יחסית של נקודות בחלל.

הטבלה הבאה מסכמת את הקשר בין פונקציית ה-loss, פונקציית יצירת השכנים, ואופי התוצאה המתקבל:

	Penalize Black Pixels	L1 / L2	Penalize Empty Pixels
נאיבי	השמה מפוצלת	השמה מפוצלת	אובייקט מלא
Neighbour Successor	השמה מפוצלת	אובייקט מלא	אובייקט מלא
Line Plotter	השמה מחולק לקווים	אובייקט מחולק לקווים	אובייקט מחולק לקווים

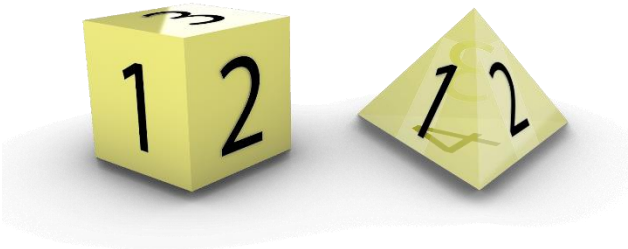
פונקציות ה-loss

נבחין כי כלל פונקציות ה- loss פועלות על המטריצה התלת מימדית אותה הן מעריכות בשני אופנים:

- אומדן המטריצה כולה.
- אומדן כל אחד מההיטלי המטריצה.

הפרמטר הנמדד על פני המטריצה כולה הינו כמות הפיקסלים המלאים \ ריקים שלה (ביחס לגודלה). מתן קנס משמעותי על פרמטר זה, גורם לאלגוריתם לתעדף מילוי מלא או ריק של המטריצה על פני דיוק מושלם לפי כל היטל והיטל. פרמטר זה משמעותי מאוד ומעודד השמה גושנית מלאה (המתאימה לייצוא האובייקט כמודל תלת מימד) או חלולה ומפוצלת (המתאימה לייצוא האובייקט כ"תעתוע ראייה").

בנוסף, **האומדן העיקרי הינו בדיקת הדמיון בין כל אחד מהיטלי המטריצה לבין התמונות הנתונות.** אומדן זה משווה כל תמונת אשר הוזנה כקלט להיטל המתאים בעבורה. בהינתן והקלט היה 3 תמונות, ההטלות יחושבו בתצורה המופיעה בצד שמאל של איור אחד (**היטל קובייתי**). בהינתן והקלט הכיל 4 תמונות, ההטלות יחושבו בתצורה המופיעה בצד שמאל של איור אחד (**היטל פירמידה**).



פונקציות ה-loss השונות אומדות את הדמיון בין כל היטל לבין תמונת הקלט המתאימה לו, על פי ה-diff הנבחר. נבדיל בין האפשרויות הבאות:

1. **False Inliers Diff -** סופרת את כמות הפיקסלים אשר הינם שחורים בתמונת הקלט, אבל שקופים בתמונות ההיטל. לפיכך, היא מקדמת צביעה מלאה של כל אזורי תמונת ההיטל, גם במחיר של "חריגה מהקווים" ויצירת רעש בהיטלים האחרים.
2. **L1 / L2 Diff -** הפונקציות סופרות את כמות הפיקסלים שאינם באותו הצבע בין הקלט וההיטל, לפי הנורמה המתאימה.
3. **False Outliers Diff -** סופרת את כמות הפיקסלים הריקים (שקופים) בתמונת הקלט, אבל שחורים בתמונת ההיטל של האובייקט. לפיכך, היא מוודאה שהצביעה לא "חורגת מהקווים" בהיטלים השונים, במחיר של השארת חללים ריקים ולא צבועים.

במהלך המחקר, **התגלה כי האלמנט הדומיננטי ביותר המבדיל בין סוגי ה-loss השונים אינו ה-diff** הנבחר (שכן ישנו יתרון מסויים ל-False Outlier Diff), **אלא דווקא אומדן <u>כמות</u> הפיקסלים** המלאים \ ריקים. על סמך תובנה זו, בחרנו להתמקד ב-3 פונקציות ה-loss הבאות:

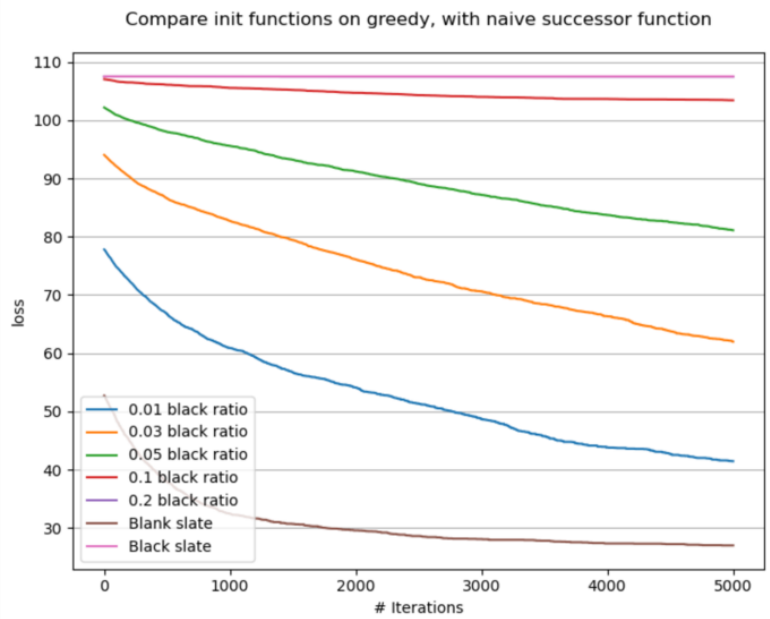
1. **Penalize Black Pixels -** משתמשת ב- False Outliers Diff בעבור מדידת המרחק בין כל היטל והתמונה המתאימה לו, ומוסיפה את כמות הפיקסלים השחורים.
2. **L2 Norm loss -** משתמשת בנורמת L2 (L2 Diff) לשם מדידת המרחק בין כל היטל והתמונה המתאימה לו, לא משכללת את כמות הפיקסלים השחורים או הריקים.
3. **Penalize Empty Pixels -** משתמשת ב- False Outliers Diff בעבור מדידת המרחק בין כל היטל והתמונה המתאימה לו, ומוסיפה את כמות הפיקסלים הריקים.

חשוב להדגיש כי הפונקציות השונות פועלות בטווח ערכים שונה, ולכן משמשות אותנו בעיקר לאפיון ההתקדמות של האלגוריתמים. בפרט, **לא ניתן להשוות ביצועי אלגוריתמים לפי ערכי פונקציות ה-Loss.** בשלב זה, בחרנו להתמקד בהשוואה ויזואלית, והשוואה כמותית למול אותה פונקציה.

אפיון שיטת האקספלורציה במרחב

מצב התחלתי -

בחנו 3 מצבים התחלתיים לאלגוריתם - מצב לוח ריק (כל הפיקסלים ללא השמה), מצב לוח מלא, ומצב רנדומי בו חלק מן הפיקסלים הינם שחורים, ומרביתם ריקים.
נתבונן בגרף הבא, המציג את השינוי בפונקציית ה-loss לפי השימוש בכל אחד ממצבי ההתחלה, בעבור הקונפיגורציה הסטנדרטית:



כאשר מסתכלים על האלגוריתם החמדן עם פונקציה יוצרת השכנים הנאיבית, כפי שציפינו, הדרך הטובה ביותר להתחיל תהיה עם לוח ריק.

כאשר שיחקנו עם יחס השחורים בתמונה הצלחנו להגיע בעזרת יחס נמוך מאוד לביצועים דומים לריק ולעיתים מעט טובים יותר אך בסופו של דבר המסקנה שקיבלנו היא שללא תלול בבחירת התמונות, הלוח הריק עדיף בתור מצב התחלתי

- מעניין לראות כי במצב של לוח מלא, פונקציית ה-loss תקועה ולא מצליחה להתקדם הלאה, שכן דרוש שינוי רב מאוד של פיקסלים בכדי להבחין בשינוי כלשהו בהטלות.
- ניתן להבחין כי לעיתים פונקציית ה-loss צונחת באופן משמעותי ברצף מצומצם של פעולות - זאת עקב השימוש באופרטורים התלת מימדיים, שביכולתם להשפיע באופן שאינו לוקאלי, ולשנות השמה של מספר רב מאוד של פיקסלים בבת אחת, ובפרט - לבטל השמות של פיקסלים שחורים.
- מצב ההתחלה האופטימלי בו בחרנו הינו המצב הריק, על פי התוצאות בגרף הנ"ל.

פונקציות יצירת שכנים -

פונקציות יצירת השכנים מייצגות אמנם אופרטור במונחי חיפוש מקומי, אך מייצגות עקרון של חקירה של מרחב הפתרון. בחלק זה ניסינו לאבחן את מאפייני השיטוט הרצויים כדי לקבל עקרונות כלליים, בהם נשתמש באלגוריתמי החיפוש המקומי ובאלגוריתם הגנטי (באדפטציה לאופרטור המוטציה על פתרון יחיד).

פונקציות יוצרות שכנים מקבלות כולן מצב קיים (השמה בינארית כלשהי למטריצה תלת מימדית בגודל n*n*n), ומשנות אלמנטים מסויים בו באמצעות אופרטור כלשהו. **את האופרטורים ניתן לחלק לסוגים הבאים:**

- שינויי השמה רנדומיים.
- שינויי השמה מכוונים - תוך ניצול מאפייניי הבעיה.
- אופרטורים תלת מימדיים על כלל האובייקט (ניצול המרחב האוקלידי).

כל אחת מן הפונקציות יוצרות השכנים מכילה קומבינציה כלשהי של הפעולות האופרטורים הנ״ל, ומגוון שכנים אותו היא מייצרת, על פי הסתברויות שונות.

1. **פונקציית Flip random bits:** הפונקציה הנאיבית ביותר - מגרילה כמות מסוימת (לדוג׳ 3-5) של פיקסלים, והופכת את ערכם. הפונקציה הנ״ל יוצרת בכל איטרציה מספר מסויים של שכנים, אשר בכל אחד מהם הוחלפו מספר פיקסלים באופן אקראי. במקרה הגנטי, התהליך משמש כמוטציה על פתרון יחיד.
2. **פונקציית Line Successors:** הפונקציה מבצעת השמות במרחב **באופן המשכי**, על ידי סימון הנקודה האחרונה שהושמה (קצה הקו), וביצוע השמות המשך לשכניה, אחד בכל פעם. כמו כן, הפונקציה מגרילה את צבע ההשמה הבאה לפי היחס המנורמל בין כמות הפיקסלים הצבועים לבין סך כל התמונה. במקרה של לוח ריק, הפונקציה תיעזר ב-Naive Successors בכדי לייצר נקודת התחלה חדשה ורנדומית, עם פיקסלים להתחיל מהם את צביעת הקווים.
3. **פונקציית Neighboring Successors:** קומבינציה של אופרטורים שונים המופעלים בהסתברויות נתונות שונות, כל שכן נוצר על ידי אופרטור אחד בלבד. במסגרת זאת -

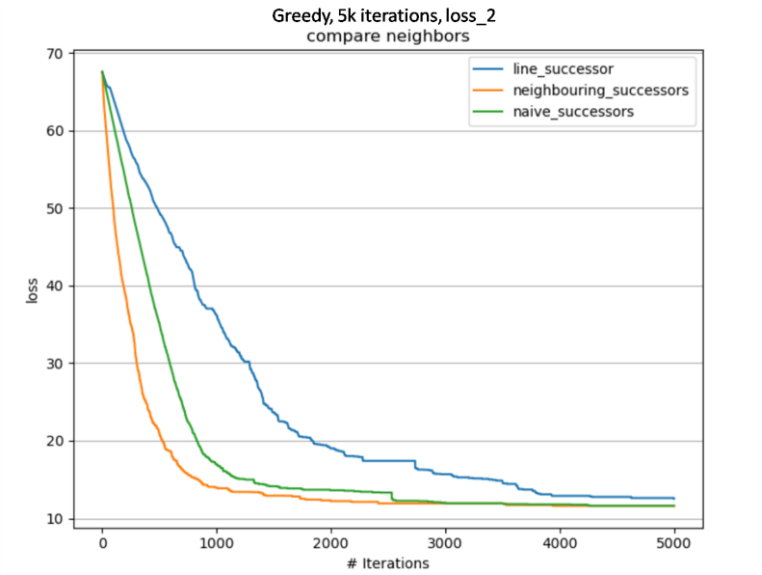
א. **מיקוד בשיפור אובייקט קיים (exploitation)** - בהסתברות הגבוהה ביותר (0.8), נבחר פיקסל שחור באקראי ומתבצעת השמה אקראית לסביבה שלו (קוביה בגודל 3x3). לפיכך, לאורך הריצה, כאשר הולכים ומתקבעים איזורים צבועים באובייקט, ההשמות של פיקסלים סמוכים לאיזורים אלו ישתנו בתדירות גבוהה בהרבה מאשר שאר הפיקסלים באובייקט. כלומר, **האופרטור ממקד את האלגוריתם לשפר ריכוזי פיקסלים מושחרים** שכבר הצליח לזהות, ומונע חיפוש של איזורים חדשים לצביעה (לרוב, רעש).

ב. **שיטוט רנדומלי (exploration)** - בהסתברות נמוכה יותר, הפונקציה תיעזור ב- Flip Random Bits ותייצר שינוי רנדומלי למספר פיקסלים.

ג. **קיבוע מבנה האובייקט הקיים והדגשת מאפייניו (exploitation) -** בהסתברות הנמוכה ביותר, מופעל אופרטור תלת-מימדי על כלל התמונה. האופרטור בוחר בהסתברות אחידה איזו פונקציה להפעיל מבין הבאות:

- פעולות פתח-סגור בוליאניות (הרחבה, צמצום, פתיחה, סגירה).
- טשטוש גאוסיאני לאורך כל צירי המטריצה.

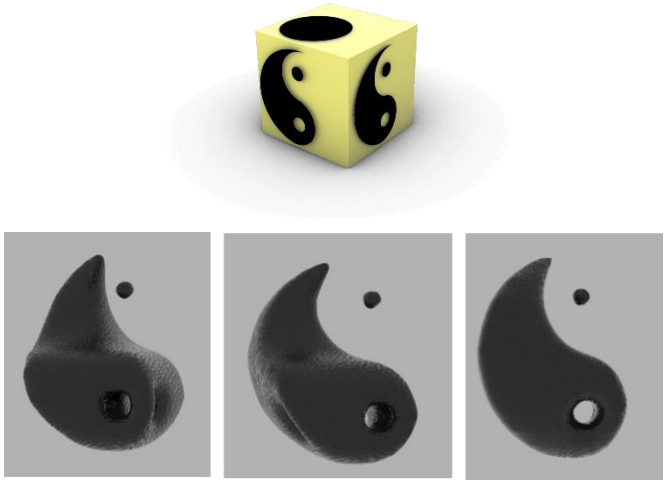
האופרטורים הנ״ל משפיעים על כלל המטריצה, ומייצרים **אחידות גבוהה יותר בין אזוריה הריקים והצבועים**. באמצעות הטשטוש **נמחקים פיקסלים צבועים בודדים**, ובאמצעות האופרטורים הבוליאניים **אזורים צבועים עשויים להתרחב, וחורים ריקים במטריצה עשויים להתמלא**. כמו כן, נבחין כי האופרטורים הנ״ל הינם ״מנקי רעשים״, ולפיכך פעמים רבות פועלים על קלט רנדומי ומורידים באופן משמעותי את ערכי פונקציית ה-loss.



נבחין כי הפונקציה Neighboring Successors משיגה את ערך מינימלי, באופן עקבי עם פונקציות Loss נוספות.

דוגמת הרצה

כדי להמחיש את מאפייני התוצר המתקבל, נתבונן בפלט. זוהי דוג׳ לקלט אפשרי לתוכנה - 3 התמונות הבאות, כאשר כל אחת מהן משוייכת לפאה אחרת של קובייה, והפלט.



ניתוח גישות אלגוריתמיות שונות לפתרון

גישה א' - חיפוש מקומי (Local search)

הקדמה

חיפוש מקומי היוריסטי היא שיטה שבעזרתה מנסים למצוא פתרון מקרב לבעיה בצורה יעילה, גם אם הבעיה המקורים היא NP קשה. במהלך הסמסטר למדנו דרכים שונות להשתמש בהיוריסטיקות, במקרה שלנו נשתמש באלגוריתמי חיפוש היוריסטיים שמהווים סוכן נוסע בתוך מרחב הבעיה שלנו. מרחב הבעיה גדול מאוד וגדל ככל שמשתמשים בתמונות ממימד גבוה יותר. נצטרך למצוא דרך לסייר בתוך מרחב המצבים העצום ולהגיע לבסוף לפתרון אופטימלי או מקרב לאופטימלי. בבעיה שלפנינו אין חשיבות לדרך שנעשתה אלא רק לתוצאה הסופית. בעזרת שימוש בפונקציות הכשירות שמעריכות את המצב הנוכחי ונותנות לו ערך היוריסטי על טיבו, נשוטט במרחב הבעיה ונחפש בדרכים שונות את הפתרון המקורב ביותר לאופטימלי שיתן לנו את הציון המינימלי לפונקציות הכשירות. החיפוש ההיוריסטי פועל למעל מטרה זו תוך ביצוע פעולות שעוזרות לנוסע להמנע מהגעה למינימום מקומי.

ייצוג הבעיה מעל מרחב מצבים במהלך המחקר על דרך פתרון הבעיה שלנו, נתקלנו באתגר לא פשוט והוא להבין מהו מרחב המצבים שלנו ואיך ניתן להגדיר למצב מסוים מצבים שכנים. בחרנו להסתכל על מצב כהשמת ספירות במרחב, לכל מצב ניתן להפיק כמות אקספוננציאלית של מצבים שכנים. הבעיה גדלה ככל שמימד התמונות גדל, לכן אנו מתייחסים לבעיה כאל LNS - large neighborhood search. פתרון בעיות מסוג זה בא לידי ביטוי בשימוש רב באלמנט האקראיות.

1. **בחירת המצב ההתחלתי:**

במחקר נבחן איזה מצב התחלתי יניב את הפתרון הטוב ביותר,

1.1. אקראי - הגרלה של נקודות במרחב שיושחרו. כוכבית - עם פרמטר של יחס עדיפות בין פיקסל מושחר ללבן.

1.2. ריק - המרחב התלת מימדי ריק מנקודות, הבעיה תהיה אילו נקודות להוסיף.

1.3. מלא - המרחב התלת מימדי מלא לחלוטין בנקודות והבעיה תהיה אילו נקודות יש להוריד

- ההנחה הראשונית הייתה שהמצב הריק יהיה הטוב ביותר לאחר מכן האקראי ואז המלא. מהר מאוד הבנו שכאשר משתמשים במצב המלא, קשה מאוד לבצע חיפוש מכיוון ששינויים קטנים במצב לאו דווקא ישנו את ההטלות על מרחבי התמונות.

2. **בחירת מצבים שכנים:**

כפי שציינו, הבעיה שלנו היא בעיה מסוג LNS, לכן נאלצנו להשתמש בפונקציות יוצרות שכנים עם אופי אקראי. בנוסף ניתן לשלוט בכמות השכנים שניצור, הרבה שכנים יובילו לביצועים פחות טובים, מעט

שכנים עלולים למנוע הגעה למצב אופטימלי. נרצה לחקור מהו המספר שיספק את הצרכים תוך כדי שמירה על ביצועים טובים.

2.1. פונקציה יוצרת שכנים נאיבית - בהינתן מצב, הפונקציה מגרילה מספר סופי של קואורדינטות במרחב התלת מימדי ויוצרת מצבים כך שכל מצב הוא שינוי של קואורדינטה אחת מתוך הקורדינטות שהוגרלו.

2.2. פונקציית בצל - נביט באובייקט התלת מימדי שלנו בתור צורה עשויה שכבות. קובייה תלת מימדית ניתן לפרק לשכבת הפאות החיצוניות ביותר שלה, השכבה הבאה תהיה שכבת הפאות השנייה בגודלה וכך נמשיך עד שנגיע למרכז האובייקט (בהשראת בצל או בבושקה). בהינתן מצב, הפונקציה תגריל שכבה במרחב בעזרת התפלגות נורמלית עם תוחלת 0 ושונות כך שיהיה סיכוי גדול יותר לקבל את השכבות הפנימיות. עבור השכבה שנבחרה, ניתן לנקודות בה הסתברות קטנה להשתנות כך שבתוחלת המצב לא ישתנה בצורה דרסטית, נשנה את הפיקסלים שנבחרו וכך ניצור שכן חדש - שינוי בשכבה אחת.

2.3. פונקציית פליפר - ניתן לכל הקואורדינטות הסתברות להשתנות ונגריל מתוכן את השינויים. פונקציה זו תתן לנו מצבים דומים עם מספר קואורדינטות שהתחלפו

2.4 הלבנת שורות\שורות - נבחר בכל איטרציה כמות סופית של עמודות\שורות\טורים ונלבין אותם. פונקציה זו בעיקר מתאימה לפתרון הבעיה כאשר מתחילים עם מצב של לוח מלא או רנדומלי.

- ההנחה שלנו הייתה שפונקציית הבצל תהיה הכי טובה מכיוון שבאינטואיציה, הבנייה ההדרגתית של השכבות תהיה נכונה. בנוסף, חשבנו שהפונקציה הנאיבית תהיה הכי פחות טובה. כמו כן, פונקציית הלבנת השורות יכולה לשמש בסיס טוב לפתרון הבעיה כאשר משתמשים בשיטת exploration vs exploitation עליה נרחיב בהמשך.

3. מדדים לאיכות הפתרון - פונקציות כשירות:

התיחסנו לבעיה בתור בעיית מינימיזציה. נרצה למצוא את המצב בו המרחק בינו לבין הפתרון האופטימלי הוא הנמוך ביותר, לשם כך מימשנו מספר פונקציות הערכה שנותנות ערך נמוך יותר לפתרון טוב יותר. בכל פונקציית כשירות, ביצענו הטלה של האובייקט התלת מימדי למישורים בהם נרצה להבחין בתמונות וחישבנו את ההבדלים בין ההטלה לתמונה המקורית בצורות שונות.

אלגוריתמים:

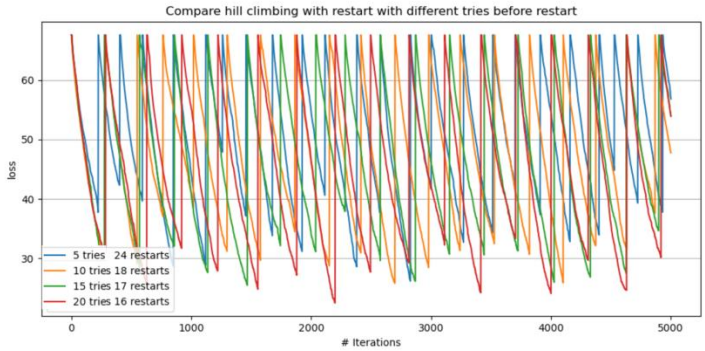
1. אלגוריתם greedy hill climbing:

האלגוריתם החמדני מתקדם בכל פעם לעבר הפתרון הטוב ביותר. הוא בוחר מבין שכניו למי להתקדם. כאמור, מספר השכנים בבעיה הוא גדול ולכן משתמשים בפונקציות יוצרות שכנים אקראיות מהן נבחר

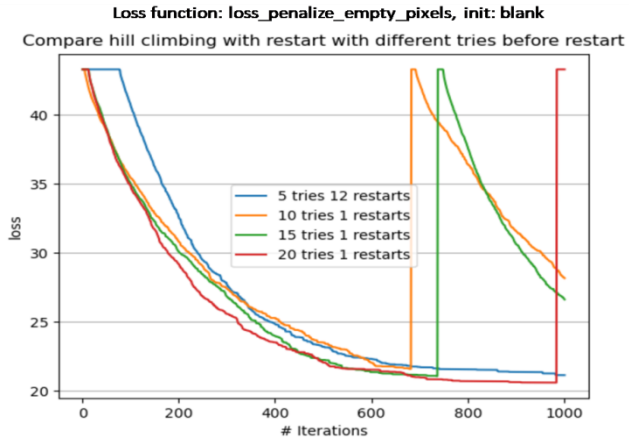
את כיוון ההתקדמות. אלגוריתם זה מביט צעד אחד קדימה ולא מעבר ולכן עלול להיתקע במינימום לוקאלי.

2. אלגוריתם hill climbing with restarts:

שיטה זו דומה לשיטה הקודמת למעט האפשרות לבצע אתחול למצב התחלתי חדש, במידע ולא מצאנו שכן טוב יותר. בעזרת התכונה הזו ניתן יהיה לצאת ממצבים של מינימום לוקאלי. בבעיה שלפנינו לא ניתן לדעת בצורה חד משמעית שהגענו למינימום לוקאלי מכיוון שלא מתבצעת בדיקה לכל המצבים השכנים לנוכחי, אך עדיין נרצה לאפשר התחלתה מחדש ולכן לאחר הגרלה של מספר שכנים פחות טובים נוכל לעשות זאת.



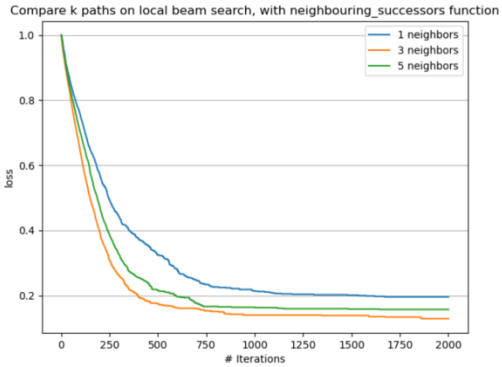
כפי שניתן להסיק מהגרף, שיטה זו עם פונקציית המחיר הזו, אינה השיטה האופטימלית לפתרון המשימה. האלגוריתם לא מספיק להגיע לציונים נמוכים כפי שהיינו רוצים ומאתחל את עצמו יותר מידיי פעמים. מהסתכלות בהרצות של אלגוריתמים אחרים, כאשר מבצעים הרבה איטרציות, לקראת הסוף יש התקדמות מינורית אך עדיין התקדמות כלשהי לעבר הפתרון המקרב, ושיטה זו מבטלת את ההתקרבות המינורית ולכן פחות



מצד שני, כאשר החלפנו את פונקציית המחיר, קיבלנו שהאלגוריתם מקבל שכנים טובים כמעט בכל הרצה ורק לקראת סוף הריצה מגיעים למינימום מסוים. גם פה לא קיבלנו איזשהו פתרון שאפשר לאמר שהוא עדיף על הפתרון החמדן ולכן נסיק ששימוש באתחולים לא מועיל לנו בבעיה.

3. אלגוריתם local beam search:

ניצור אוסף של K מצבים, בכל איטרציה נביט במצבים הקיימים ובכל המצבים השכנים להם ונבחר את הK הטובים ביותר מבין כל האוסף. כך אנו מדמים התקדמות לעבר הפתרון בK מסלולים שונים שמתקשרים זה עם זה. לבסוף נבחר את המצב הטוב ביותר מתוך הK האחרונים. החיפוש המקבילי מוביל במהירות לנטישת חיפושים לא פוריים ומעביר את המשאבים למסלולים הטובים ביותר.



ניתן לראות שאכן יש שיפור בתוצאה כאשר משווים בין כמות המסלולים עליהם אנו מסיירים.

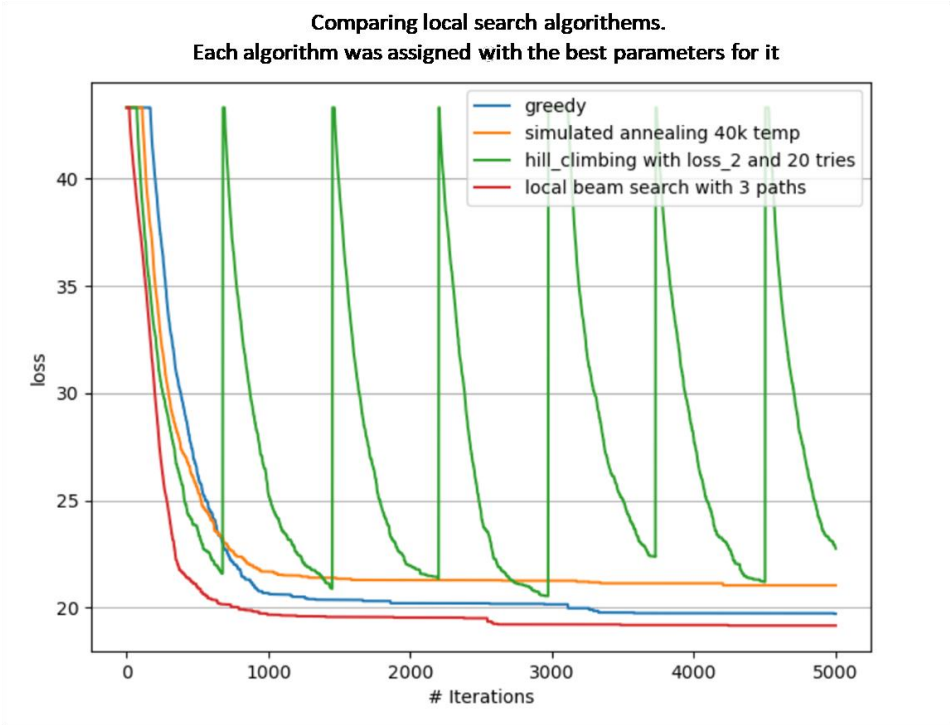
מסלול אחד זהה לאלגוריתם החמדני והוא מקבל ציון פחות טוב מאשר 3 מסלולים. כמו כן אנחנו כן יכולים להבחין כי בין 3 מסלולים ל5 מסלולים אין יחס מונוטוני כפי שהיינו מצפים. ייתכן ונראה שינויים מונוטונים משמעותיים כאשר מימד התמונה יהיה גדול יותר מ40

4. אלגוריתם simulated annealing:

חישול מדומה, היא טכניקת hill climbing הסתברותית המבוססת על חישול מתכות. תהליך טבעי זה מתרחש לאחר הסרת מקור חום ממתכת מותכת, טמפרטורת המתכת מתחילה לרדת ככל שהחום עובר לסביבה. בכל רמת טמפרטורה, האנרגיה של מולקולות המתכת פוחתת והמתכת הופכת למוצקה יותר. זה ממשיך עד שטמפרטורת המתכת שווה לטמפרטורת הסביבה, ובשלב זה האנרגיה הגיעה למינימום שלה. אלגוריתם החישול המדומה מחקה את תהליך החישול הטבעי הזה כאשר הוא מחפש אופטימום. בעזרת פרמטר טמפרטורה הולך ויורד המהווה סיכוי לעבור לשכן פחות טוב, נוכל להתקרב למצב האופטימלי.

ניתוח השוואתי:

- **השערות:** לפני הרצת התוצאות שיערנו שסדר האלגוריתמים לפי טיב הביצועים שלהם יהיה לפי סדר הופעתם כאן לעיל - החמדני הכי פחות טוב והחישול המדומה יתן את הביצועים הטובים ביותר.
- **מסקנות לאחר השוואת האלגוריתמים:**



בגרף בלפנינו השוונו בין כל אלגוריתמי החיפוש המקומי ההיוריסטי. לכל אלגוריתם השמנו את הפרמטרים שבמחקר עליהם הסקנו שהם המתאימים ביותר עבורם. הופתענו לגלות שsimulated annealing היה הכי פחות טוב מבין כולם, הסיבה לכך יכולה להיות חוסר התאמת הפרמטרים או מימד התמונה, ייתכן וכאשר נבחן את הבעיה עם תמונות גדולות במיוחד, האלגוריתמים האחרים יתקעו במינימום לוקאלי ודווקא אז simulated annealing יהיה טוב יותר. local beam search היה הכי טוב מבין השאר, הוא מנצל היטב את היכולת לעבוד במסלולים שונים תוך שמירה על תקשורת ביניהם מה שעזר להגיע לפתרונות הטובים ביותר.

גישה ב׳ - חיפוש אבולציוני

כדי להמשיך ולחקור את מרחב הבעיה, בחרנו להעמיק באלגוריתמיקה גנטית מתוך השערה כי גישה זו תניב תוצאות טובות יותר לקירוב אובייקט תלת מימדי לתמונות הקלט כאשר רמת הסיבוכיות של הבעיה עולה. בפרט, הנחנו כי מרחב הפתרונות עשיר וגדול מאחר ואין מגבלה על אופן ההצבה של נקודות במרחב התלת מימדי, וכי הבעיה הופכת מורכבת יותר כשגדלים בכמות תמונות הקלט ובמורכבותן.

אלגוריתם גנטי - הקדמה

אלגוריתם גנטי פועל בהשראת תהליכים אבולוציוניים בטבע באמצעות **אופרטורים גנטיים**. תחילה, בהינתן אוכלוסיה ראשונית של פתרונות (individuals or chromosomes) משערכים את הכשירות של כל פתרון לפי פונקציית מטרה שמנסים למזער. **בוחרים (selection)** מתוך האוכלוסיה את הפתרונות הכי מתאימים לדור הבא, ופתרונות מתאימים להורשת תכונות בתהליך של **זיווג** (crossover) כדי לייצר "צאצאים". באופן זה מרכיבים אוכלוסיה חדשה, המהווה דור המשך, כאשר הפתרונות באוכלוסיה זו עוברים **מוטציה** (mutation).

זיווג ומוטציה משרתים את הגישה הגנטית בחקירת (exploration) מרחב הפתרון ברמות שונות. אופרטור הבחירה משמש לניצול ושיפור הידע הקיים של האלגוריתם (exploitation). האיזון בין אופרטורים אלו נדרש להיות מותאם לבעיה ספציפית.

ייצוג הבעיה[2] -

א. בגרסה הבסיסית של אלגוריתמים גנטיים משתמשים **בקידוד חד-מימדי של גנים** (genes)[3]. במקרים מסוימים, **עבור בעיה רב-מימדית, מאבדים מידע חשוב** לגבי הקישור הרב-מימדי בין גנים סמוכים (לעניינו, פיקסלים סמוכים באובייקט). מכאן, **קידוד כרומוזום כמטריצה רב-מימדית** שומר על ייצוג טבעי של הבעיה ועל המבנה הפנימי של מרחב הפתרונות, ובשאיפה, כך שיערנו, יביא לביצועים טובים יותר מהקידוד הסטנדרטי.

ב. מאחר וקידוד תלת-מימדי מקשר את הייצוג של פרטים באוכלוסיה למבנה הבעיה המקורי, אז **אופן הקידוד מאפשר לאופרטורים הגנטיים לפעול ישירות על הפתרון עצמו**. באופן זה, למשל, זיווג של שני פתרונות יכול להתבטא בהורשת תכונות מרחביות (תלת מימדיות) של שני אובייקטים תלת מימדיים שונים. נתייחס לסוגיה זו בהמשך.

נציין כי נבחנו קידודים אחרים כדי לשפר את זמן ריצה אל מול תמונות ברזולוציה גבוהה יותר, שמובילות לקוביה תלת מימדית גדולה וכבדה להרצה. למשל, קידוד פתרון בתור מערך של אינדקסים של פיקסלים מושחרים וריפוד לבחירה של פיקסלים נוספים[4]. עם זאת, **במסגרת גדלי התמונות שבחנו, המודל הסטנדרטי רץ בזמן סביר, לכן לא מימשנו מנגנון לפתרון בעיה זו.**

---

[2] בהשראת מאמר של J. Merta, T. Brandejský, ראו [1] בפרק ביבליוגרפיה.

[3] פרטים באוכלוסיה (מכונים אינדיבידואלים או כרומוזומים) הינם מערכים חד מימדיים של גנים. האוכלוסיה מורכבת ממערכים באורך כמות הפיקסלים בקוביה תלת מימדית במימד n^3 כאשר n הינו אורך תמונות הקלט. כל גן מקבל ערך 1 או 0. בהתאם, פעולות הזיווג והמוטציה מתבצעות ביחס למערכים חד-מימדיים.

[4] Variable selection, רעיון שהוצג במאמר של C. Chun Gan ו- G. Learmonth. ראו [2] בפרק ביבליוגרפיה.

גרף שמשווה בין single cut והשני.

אתחול האוכלוסיה הראשונית - לכל פרט באוכלוסיה מגרילים לכל היותר K (נגזר מפרמטר black-ratio) פיקסלים שחורים בקוביה התלת-מימדית, ושאר הפיקסלים נצבעים לבן. נדגיש כי בחירת חסם עליון לכמות הפיקסלים השחורים ההתחלתית הכרחית כדי לנווט במרחב הפתרונות. זאת, מאחר ופונקציית הכשירות מבוססת ההטלה מורעשת מדי מכדי לתת ערכים שונים למצבים שונים עשירי פיקסלים מושחרים[5].

אופרטורים גנטיים

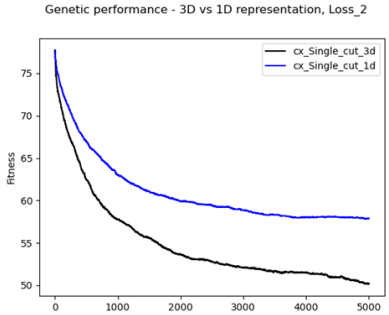
השפעת קידוד הכרומוזומים על זיווג ומוטציה

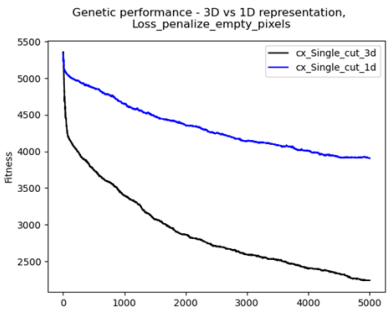
א. **אופרטור הזיווג** בייצוג החד-מימדי בדרך כלל עובד עם שני כרומוזומים הורים ומחלק אותם אקראית לשתיים או יותר קבוצות של גנים, הצאצאים יורשים חלק מהקבוצות של הורה א' וחלק אחר מהורה ב'. **פעולת המוטציה** משפיעה נקודתית על הכרומוזום - לרוב שינוי של מספר גנים מצומצם.

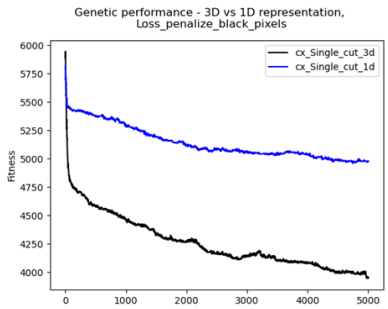
ב. כאשר הגנים מקודדים כמערך חד-מימדי, בשל הטבע הרנדומלי של האופרטורים הללו, **לא נוכל להבטיח כי האופרטורים מייצגים פעולות בעלות משמעות במרחב הבעיה המקורית** (spatial). בפרט, יתכן כי גנים סמוכים לא מייצגים נקודות סמוכות במרחב התלת-מימדי, ולכן מאבדים קשר טבעי בין הפעולות על גנים סמוכים (למשל נקודת חלוקה של הכרומוזום) להשפעה דומה על האובייקט התלת מימדי שהם מייצגים[6].

ג. לכן, נעזרנו בקידוד תלת-מימדי כדי להגדיר אופרטורים שמנצלים את מבנה הבעיה המרחבי, לרוב באמצעות הכללה של רעיונות קיימים במקרה החד-מימדי. נשווה בין סוגי האופרטורים כדי להצביע על האפקטיביות של המרת הקידוד.

ד. בפרט, בחנו אופרטורים שקולים - One-Point-cut מעל קידוד חד-מימדי ומעל קידוד תלת-מימדי, על פונקציות ה-Loss השונות. ניכר כי הייצוג התלת-מימדי מגיע לתוצאות טובות יותר.







[5] ראו סעיף בנוגע לבעיה זו בתחילת הפרק.

[6] תכונה זו, המכונה linkage, מתוארת במאמר של Kahng and Moon לצד עקרונות ייצוג נוספים שראוי לבחון במחקר המשך. ראו [3] בפרק ביבליוגרפיה.

כעת, נפנה להצגת האופרטורים הגנטיים והאלגוריתמים אותם בחנו במהלך המחקר.[7]

בחירה (selection)

א. Tournament - מספר קבוע (3) של מתמודדים נבחר באקראי מתוך האוכלוסיה, מתוכם נבחר לשמש כהורה המתמודד בעל הכשירות הגבוהה ביותר.

זיווג (crossover)

ב. 3D-single-cut - בכל ציר מוגרלת נקודת חיתוך, כך שהקוביה מחולקת ל-8 חלקים ממוספרים. הצאצא הראשון מקבל את תתי-הקוביות האי-זוגיות (1,3,5,7) מהורה א׳, ואת תתי-הקוביות הזוגיות מהורה ב׳. הצאצא השני יורש את הגנים בסדר ההפוך.

ג. Block-uniform - הקוביה המקורית מחולקת ל-8 תת-קוביות באופן אקראי. הצאצא הראשון יורש את כל הגנים של הורה א׳, ובמעבר על תתי-הקוביות שהוגרלו יורש גנים מקוביה של הורה ב׳ בהסתברות נתונה (0.1). יצירת הצאצא השני מתבצעת באותו האופן.

ד. Rectangle-style - תת-קוביה קטנה במימד לכל היותר עשירית ממימד הקוביה הכללית מוגרלת. צאצא א׳ יורש את הגנים מהורה א׳, ואת הגנים של הורה ב׳ הנמצאים בתת-הקוביה שהוגרלה. צאצא ב׳ מוגדר באופן שקול.

הערה 1: בעוד שהאופרטורים מוגדרים על כרומזום תלת-מימדי, רובם מבוססים על עקרונות זיווג בסיסיים שמוצגים במרבית האלגוריתמים הגנטיים. באותו אופן, ניתן להרחיב אופרטורים נוספים כגון k-cut crossover.

הערה 2: השתמשנו **בזיווג שמתבסס רק על חיתוך של כרומוזום לאורך הצירים, כך שנפגעת היכולת לשמר או להוריש מבנים תלת-מימדיים מורכבים** לדורות המשך. כדי להתמודד עם הבעיה, נבחן אופרטור[8] בשם geographical-crossover שמבצע את החלוקה באמצעות חיתוך לפי מספר היפר מישורים בזוויות שונות. חקירת אופרטור זה הושארה כמשימה להמשך המחקר, ולא תיכלל במאמר זה.

מוטציה (mutation)

ה. מבין הפונקציות שפיתחנו המנצלות את מאפייני הבעיה (והוצגו בפרק הראשון) נבחן את הפונקציות הבאות כאופרטור מוטציה - Flip random bits, Neighboring Successors.

פרמטרים קבועים

א. Elitistm - מוסיפים לכל דור ההמשך את הפתרון הטוב ביותר שהושג. תהליך זה מבטיח שדור ההמשך טוב לפחות כמו הדור שלפניו.

---

[7] חלק מהאופרטורים מיושמים בהשראת מאמר בסעיף [1] בפרק ביבליוגרפיה.

[8] מופיע במאמר של A. B. Kahng ו- B.R. Moon, ראו סעיף [4] בביבליוגרפיה.

אלגוריתמים:

מאחר וקיימות וריאציות רבות ושונות של אלגוריתמים גנטיים, ומפני שתעדפנו אופטימיזציה של השימוש באופרטורים גנטיים באלגוריתם, בחרנו לקבע אלגוריתם יחסית בסיסי ולמדוד את המודל ביחס אליו. בחרנו באלגוריתם לא טריוויאלי מאחר והאלגוריתם הנאיבי הציג ביצועים גרועים מדי כדי להשוות.

האלגוריתם בו נתמקד (eaMuCommaLambda) פועל באופן הבא[9]:

קלט:

- אוכלוסיה ראשונית של פרטים במרחב הבעיה.
- K - כמות צאצאים חדשים בכל אוכלוסיה.
- הסתברויות לכל אופרטור גנטי.
- כמות הדורות הרצוי (מספר איטרציות).

אופן פעולה:

1. שערוך אוכלוסיה נתונה.
2. במהלך K איטרציות, האלגוריתם בוחר הסתברותית אחת משלוש הפעולות - זיווג, מוטציה או שכפול.
במקרה של זיווג, שני פרטים מהאוכלוסיה הנוכחית נבחרים באקראי, ומזווגים. **רק אחד הצאצאים נשמר.** במקרה של מוטציה, נבחר פרט באקראי, ועובר מוטציה. שכפול גורר העתקה של פרט אקראי.
לאחר מכן, קבוצת הצאצאים משוערכת ביחס לפונקציית הכשירות ומתבצעת **בחירה מהצאצאים ומהאוכלוסיה המקורית**.
בפרט, נשים לב כי אף צאצא מאוכלוסיית ההמשך אינו תוצר של זיווג וגם רביה, אלא רק אחד מהשניים (או אף אחד מהם).

האלגוריתם הנאיבי מוגדר עם האופרטורים הבאים ועם ההסתברויות הבאות:

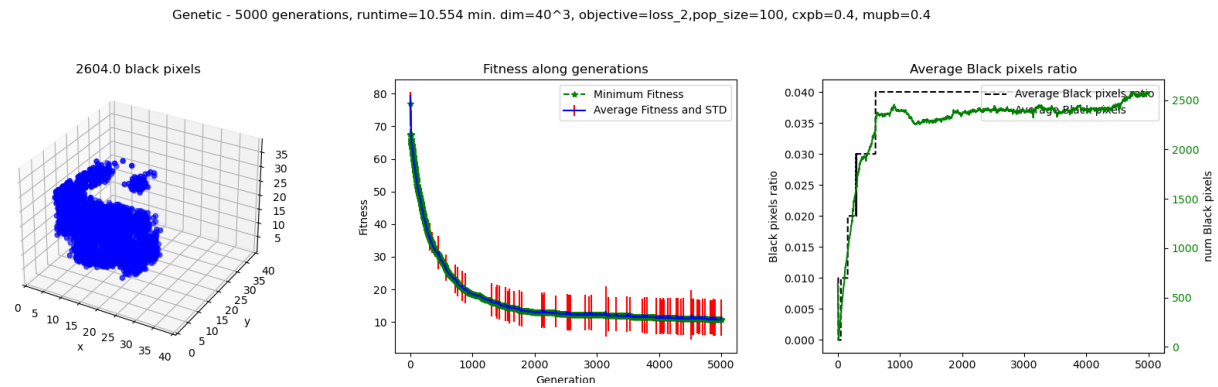
א. **אוכלוסיה ראשונית -** הגרלת נקודות במרחב לצביעה בשחור, ביחס של 0.01 מנפח הקוביה.
ב. פונקציית מטרה - Loss2.
ג. בחירת (selection) טורניר.
ד. זיווג single-point תלת מימדי (נשווה עם חד מימדי). בהסתברות 0.3
ה. מוטציה flip random bit על 5 ביטים, בהסתברות 0.3.

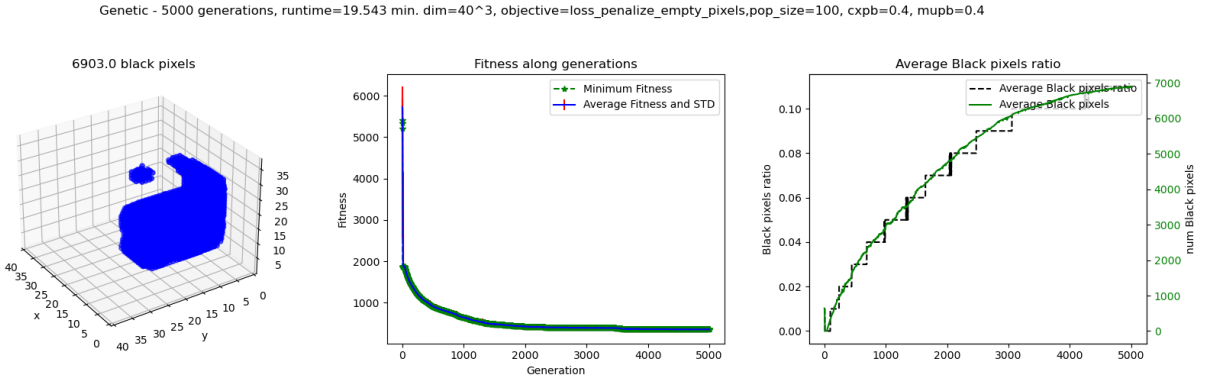
[9] לקוח מהחבילה DEAP, תיעוד ב- https://deap.readthedocs.io/en/master/api/algo.html#module-deap.algorithms

תוצאות ומסקנות

ראשית, זיהינו מיידית בבדיקה כי עבור מספר הדורות שבדקנו, פונקציית המוטציה הנאיבית (Flip random pixels) משיגה תוצאה יחסית נמוכה, בעוד שפונקציית neighbouring-mutator השיגה ביצועים גבוהים בהרבה. לאור זאת, התקשינו לזהות מאפיינים חד-משמעיים לגבי החשיבות של אופרטורי הזיווג לאור השיפור המיידי והמשמעותי בביצועים לאחר התאמת פונקציית המוטציה הנ"ל.

בגרף הבא מתוארת ריצת האלגוריתם הנאיבי ביחס לאחר התאמת אופרטור המוטציה ל-neighbouring_mutator, עבור פונקציות כשירות שונות. כצפוי, נראה כי פונקציית המטרה מסייעת לאלגוריתם "להתמקד" בדרך לפתרון מסוים, ולכן מצמצמת את השונות. גרפים אלו ממחישים את האפקט הדרסטי של exploitation מאחר והשונות באוכלוסיה הולכת ומצטמצמת כמעט לגמרי.

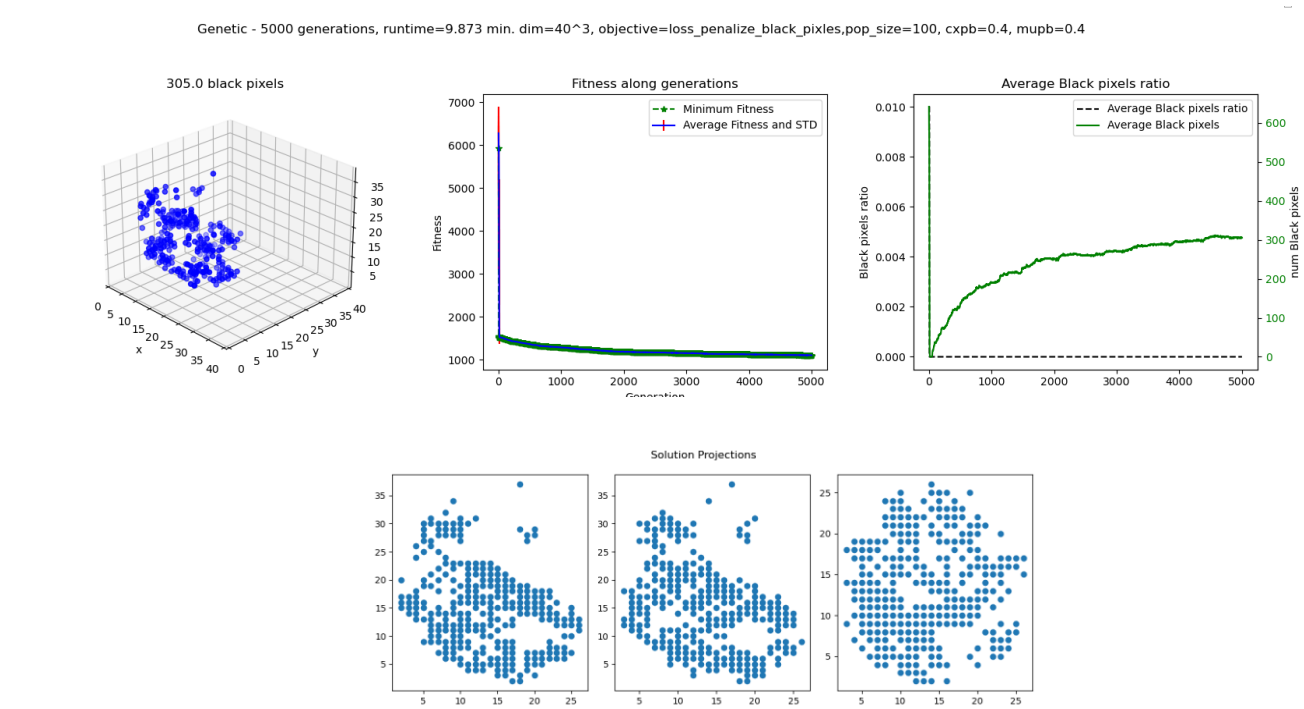


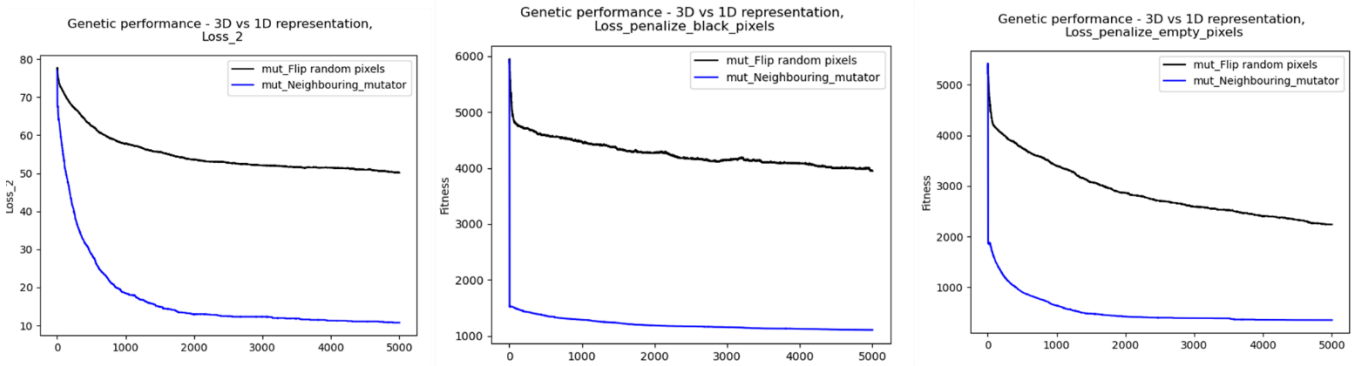


להבנתנו, הגם שהאלגוריתם המשופר הנ"ל מציג התכנסות מהירה לעבר פתרון, יתכן כי קיימים יתרונות לאקספלורציה ארוכת טווח (מעבר לכמות הדורות שבדקנו) לטובת השגת מאפיינים ויזואליים נוספים או קירוב טוב יותר.

במסגרת זאת, ניתן להבחין כי פונקציית המטרה penalize_white_pixels משיגה תוצאה מעולה בעקביות, בעוד הריצה עם loss_2 משיגה תוצאה פחות מדויקת (או ממוקדת). לעומתן, הפונקציה penalize_black_pixels משיגה את התוצאה ההפוכה - היא מציגה התכנסות מהירה וממוקדת, אך לעבר

מינימום מקומי מובהק. נקודת ההתכנסות בריצת האלגוריתם מתגלה בדור מוקדם, אך מתקבעת על אובייקט שמציג מאפיינים דומים לאובייקט היעד, אך ברמה ויזואלית נמוכה בהרבה.





נדגיש כי **נבחנו מוטציות נוספות מעל אובייקט תלת-מימדי**, שמשנות רנדומית איזורים מתוחמים במרחב האוקלידי, למשל - בחירת תת-קוביה רנדומלית בתוך האובייקט הקיים ולהפוך כל ערך בהסתברות מסוימת. טיבן של פונקציות אלו עשוי היה לשמר אופן פעולה יותר אקספלורטיבי, אך הבחנו כי שימוש בזיווג שנשען על אובייקטים מלבניים ביחס לצירים לצד מוטציות שמבוססות על מנגנון דומה, מעוותות את טיב האובייקט המתקבל - מתקבל אובייקט בעל מאפיינים ריבועיים. לכן, בין היתר, בחרנו לסמן כציר המשך בחינה של אופרטורים שמנצלים מורכבויות אחרות במרחב האוקלידי (חיתוך ופיצול של האובייקט בהתבסס על היפר מישורים בזוויות שונות). למשל -

תוצר של ריצת האלגוריתם על זיווג -3D-single
cutעם מוטציית flip_random_sub_cube

בהיבטי אופרטור הזיווג, לא זיהינו פתרון טוב יותר מהאופרטור הנאיבי, אך נדרשת העמקה נוספת.

לסיכום, הקומבינציה המיטבית שמצאנו (במגבלות היקף ומגוון הבדיקות והקלטים) הינה -

א. **אוכלוסיה ראשונית -** הגרלת נקודות במרחב לצביעה בשחור, ביחס של 0.01 מנפח הקוביה.
ב. פונקציית מטרה - Penalize_white_pixels.
ג. בחירת (selection) טורניר.
ד. זיווג single-point תלת מימדי (נשווה עם חד מימדי). בהסתברות 0.3
ה. מוטציה Neighboring_mutator על 5 ביטים, בהסתברות 0.3.

חלק ג' - מציאת השמה חוקית למשתנים (Constraint satisfaction problem)

הערה: בכל מקום בו רשום dim הכוונה לגודל של הצלע. כלומר הקלט הוא 2 או 3 תמונות בגודל 2dim והפלט הינו מטריצה מגודל 3dim.

למה בחרנו להשתמש ב CSP כדרך פתרון: כשחשבנו על הבעיה שאנחנו רוצים לפתור. ראינו שהבעיה דומה במקצת לשחור ופתור תלת - מימדי, בכך שלכל פיקסל יש 3 צירים שונים שהוא שייך אליהם (בניגוד ל2 בשחור ופתור רגיל). מכיוון ששחור ופתור הינה בעיית CSP קלאסית, חשבנו שגם הבעיה שלנו תוכל להיפתר באופן דומה ע"י CSP

הגדרת הבעיה בתור בעיית CSP: מכיוון שיש לנו פיקסלים כשכל פיקסל יכול להיות או מלא או ריק, התחלנו את דרך הפתרון בלהגדיר את הבעיה כבעיית CSP כך:

variables = {x0,,xdim3-1}

domain ={0,1}

כאשר כל 2dim משתנים/פיקסלים מייצגים שכבה במטריצה.

כעת, כדי להגדיר את ה constraints:

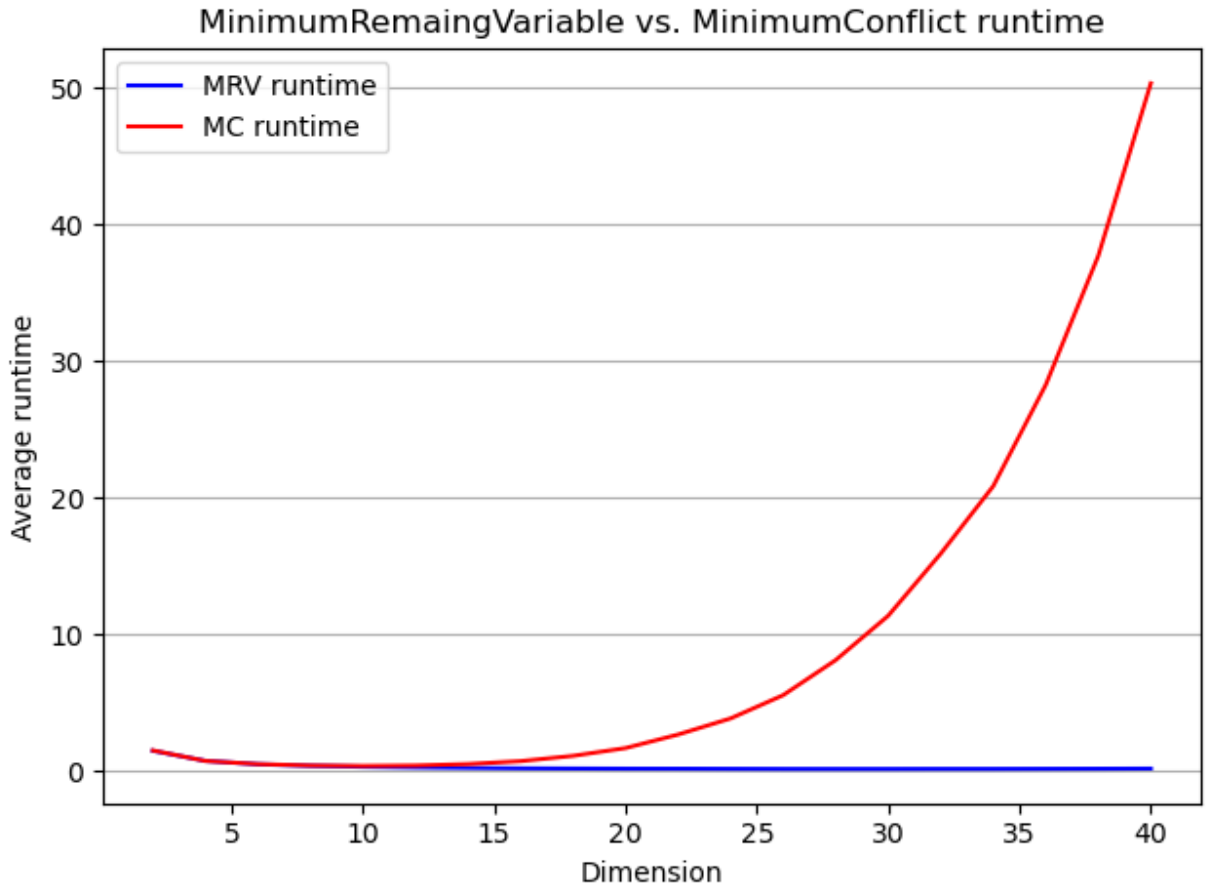
לכל תמונה בחרנו ציר, עברנו על כל פיקסל והגדרנו:

- אם הפיקסל ריק , אז סכום כל השורה בציר שמאחורי הפיקסל הזה צריך להיות 0 (שזה שווה ערך לכך שכל השורה מלאה באפסים).
- אם הפיקסל מלא , אז סכום כל השורה בציר שמאחורי הפיקסל צריכה להיות יותר מ0 (שזה אומר שיש לפחות פיקסל אחד מלא בשורה ולכן יהיה ניתן לראות את הפיקסל מהזווית הזו).

מציאת אלגוריתם מתאים: אחרי שהיה לנו את הבעיה מוגדרת, התלבטנו באיזה אלגוריתם להשתמש. Backtracking לא ממש בא בחשבון כי יש 3dim2אפשרויות לסידור הפיקסלים. ניסינו אחר כך להשתמש בחבילת קוד פתוח שפותרת בעיות csp ע"י שימוש באלגוריתם Minimum Conflict עם הבעיות שהגדרנו כדי לנסות לקבל פתרון אופטימלי (או הכי קרוב לאופטימלי אם לא קיים פתרון אופטימלי). במקביל הגענו למסקנה שמכיוון שיש לנו constraint שמחייב לנו שיהיה לנו שורה שלמה של אפסים, אנחנו יכולים להשתמש ב Minimum Remaining Variable מכיוון שכל המשתנים שנמצאים בשורות שיש בהם אפסים יש להם רק השמה אחת.

אז עברנו על כל הפיקסלים שהיו חייבים להיות 0 והשמנו אותם כ0, את כל שאר הפיקסלים (שלא חייבים להיות 0) השמנו ל1. קיבלנו שאם יש פתרון אופטימלי, האלגוריתם יחזיר לנו אותו. אם אין, האלגוריתם יחזיר לנו פתרון מקורב.

כעת, שהיו לנו 2 דרכי פתרון, הרצנו את 2 האלגוריתמים על אותם תמונות ומדדנו את זמני הריצה שלהם, אחוזי הצלחה (למציאת פתרון אופטימלי) וטיב הפתרונות (כאשר הפתרון אינו אופטימלי). קיבלנו שאחוזי ההצלחה וטיב הפתרונות היו זהים כל הזמן אבל זמן הריצה של אלגוריתם Minimum Remaining Variable היה קצר משמעותית מזמן הריצה של אלגוריתם Minimum Conflict.



מסקנות מ CSP: לאחר שניסינו לפתור את הבעיה בכמה אלגורתמי CSP (ונחלנו הצלחה חלקית) הגענו למספר מסקנות:

- כאשר יש פתרון מדויק\אופטימלי CSP יכול לתת לנו את הפתרון בצורה מהירה ויעילה.
- בניגוד למה שחשבנו בהתחלה, הבעיה שלנו אינה דומה לבעית שחור ופתור. בשחור ופתור נתון שתמיד יש פתרון מדויק, אבל מכיוון שאצלינו לא שמנו הגבלות על הקלט. לא בהכרח יהיה פתרון מדויק ולכן CSP לא בהכרח יכול לתת לנו פתרון אופטימלי כי לא קיים כזה פתרון.
- הפתרון המקורב שאנו מקבלים כשאין פתרון אופטימלי לא תמיד יותר טוב מהפתרונות שאנו מקבלים מהאלגוריתמים האחרים שניסינו, אבל הוא יכול להוות נקודת התחלה עבורם.

3. שיטת Exploration vs Exploitation:

פיצול מחקר - ניצול היא בעיה ידועה המתרחשת במקרים בהם מערכת למידה צריכה לבחור שוב ושוב עם תוצאה לא ידועה. בעיקרו של דבר, הדילמה של מערכת קבלת החלטות שיש לה ידע לא מלא בעולם הבעיה היא האם לחזור על החלטות שעבדו טוב עד כה (לנצל) או לקבל החלטות חדשות, בתקווה לזכות בתגמולים טובים יותר (לחקור).

רצינו להשתמש בעקרון המחקר אל מול ניצול בפתרון הבעיה שלנו. נבחן את העיקרון עם שתי הסתכלויות שונות:

1. פתרון הדרגתי ע״י שימוש באלגוריתמים שונים - הרעיון היה לנסות למצוא מצב התחלתי מספיק טוב כך שנוכל בקלות יותר להגיע לפתרון טוב יותר. נפתור את הבעיה עם אלגוריתם חיפוש פשוט ומהיר בזמן הריצה שלו שיתן לנו תוצאה מקרבת ולאחר מכן נשתמש בפתרון שנתן ונעביר לאלגוריתם טוב יותר אך כבד יותר. כך ננצל את האלגוריתמים הפשוטים יותר למען הגעה למצב מספיק טוב שהאלגוריתמים הטובים יותר יוכלו לפתח ולחקור.
2. פתרון הדרגתי ע״י פתירת הבעיה במימדים הולכים וגדלים - נתחיל במימד תמונה נמוך ונפתור עם אלגוריתם חיפוש מסוים, לאחר מכן ניקח את התוצאה שהחיפוש נתן לנו, נבצע מניפולציה על הפתרון ונגדיל את מימדו. נתחיל מהמצב הזה עם אלגוריתם חיפוש אחר. כך נמשיך עד שנגיע למימד הרצוי. בכל פעם התחלנו את אלגוריתם החיפוש ממצב התחלתי שהיה הגדלה של פתרון לבעיה קטנה יותר. ננצל את העובדה שככל שהמימד קטן יותר, כך האלגוריתמים עובדים מהר יותר.

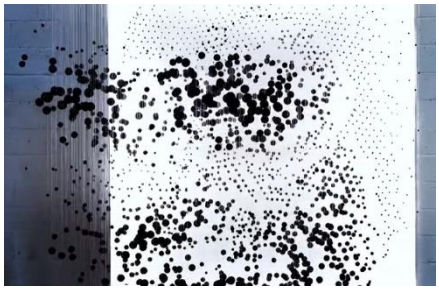
כיוונים להמשך והרחבות -

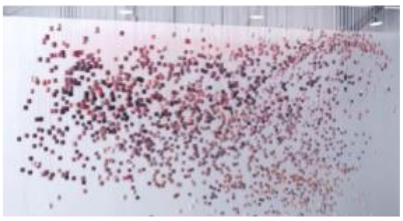
1. ככלל, נדרשת **העמקה נוספת בפונקציונליות** השונה באלגוריתמים, בדגש האלגוריתמיקה הגנטית שהציגה זווית מבטיחה להשגת תוצאות מעניינות ויזואלית.
2. שימוש באלגוריתמי אופטימזציה כדי לייעל את בחירת הפרמטרים והאלגוריתמים למול הבעיה, בדגש על **קלטים שונים.**
3. **Scale-up למרחב הבעיה.**

א. צבע, מימד.

ב. מעבר להשמה של צורות וקטוריות (תלת ודו מימדיות) - כדורים או עיגולים, ריבועים דו-מימדיים, צורות לפי גוונים ועוד.

ג. ניסיון להתחקות אחר מאפיינים ויזואליים נוספים.





ביבליוגרפיה

1. THREE-DIMENSIONAL GENETIC ALGORITHM FOR THE KIRKMAN'S SCHOOLGIRL PROBLEM, by Jan Merta, Tomáš Brandejský, University of Pardubice, Czech Republic
2. An improved chromosome formulation for genetic algorithms applied to variable selection with the inclusion of interaction terms, by Chee Chun Gan, Gerard Learmonth, University of Virginia.
3. A Survey on Chromosomal Structures and Operators for Exploiting Topological Linkages of Genes, by Dong-Il Seo and Byung-Ro Moon, School of Computer Science & Engineering, Seoul National University.
4. Toward More Powerful Recombinations, by Andrew B. Kahng and Byung Ro Moon, UCLA Computer Science Department