

# שימוש באלגוריתם Particle Swarm Optimization לפתרת פazel

מגישיים:  
תומר עמרן  
נדב שקד

## PSO Puzzle Algorithm - Git

### הקדמה

בעית הפazel היא בעיה מוכרת לפני מאות שנים, בעיה זו היא NP-Complete כפי שהוכח במאמר של אלטמן ואחרים [1]. במתלה זו התמקדנו בגרסה לבעה בה אין ידע מקדים כלל והחקקים הם ריבועים מושלמים. בעית הפazel ניתן לחלק לשני רבדים: כימות טיב הפתרון - כאשר מוצג הרכיב כלשהו של החלקים נרצה להעניק ציון גבוה להרכבה שקרובה לפתרון וציוו נמור להרכב רחוק ממנו. כיים לא קיימים אומדן שנותן בוודאות את הציון הגבוהה ביותר לפתרון הנכון. לאחר קריאה של כמה מאמרים בנושא, בחרנו להשתמש באומדן אשר הוצג במאמר של אוחד ושותפי [2]. אלגוריתם החיפוש - אחרי שהושג יחס סדר לפתרונות אופציונליים ניתן לבנות אלגוריתם חיפוש. בספרות ישן מספר גישות, חלק מהגישות שקרהנו עליהם: פתרון של מערכת לינארית במאמר של פרנדנד ואחרים [3], מימוש של אלגוריתם גנטיאטי במאמר של ג'יימס ואחרים [4] ושיטות נוספות מהמאמר של אוחד ואחרים [2]. כדי שהוצע על-ידי אוחד, מוחנו למטריה, אנחנו השתמשנו בגרסה בדידה לאלגוריתם PSO. Particle Swarm Optimization הינו אלגוריתם למציאת נקודת קיצון גlobלית בפונקציה רציפה. בעית הפazel הינה בעיה קומבינטורית ולכן נדרשנו למצוא גרסה בדידה לPSO, במתלה זו ממשנו את הגישה המופיע במאמר של מחמט ואחרים [5].

### סיכום חלק מהמאמרים בהם נעזרנו לצורך המטלה

במאמר של אוחד ושותפי [2], הוצג אלגוריתם חמדן לפתרון פazel ללא ידע מקדים כלל. המאמר הציג שיפור משמעותי במספר החלקים שהוא יכול לפתור בזמן סביר. המאמר זהה לקחנו את החלק של כימות טיב הפתרון. עד למאמר זה רוב הפתרונות שהוצעו השתמשו במרקח האוקלידי בין דפנות החתיכות. זו מטריקה אינטואיטיבית אך המאמר הוכיח שהיא מאוד לא אופטימלית. המאמר שילב שני רעיוןות, אחד היה ליצור חיזוי בעזרת טור טילור על שתי השורות האחרונות של דופן אחד כדי לחזות הצעה סבירה למה צריכה להיות הדופן של החלק הבא.

$$D_{p,q}(x_i, x_j, r) = \left( \sum_{k=1}^K \sum_{d=1}^3 (|x_i(k, K, d) - x_j(k, 1, d)|)^{\frac{3}{10}} \right)^{\frac{5}{24}}$$

והשני הוא להשתמש בדבורה שונה מ $L_2^{5/24}$ . הם עשו בדיקה אמפירית וגלו שדבורה  $L_2^{5/24}$  נותנת תוצאות מאד טובות ביחס לנורמות אחרות.

התוצאות נבדקו על-ידי למידה כל חלק ובדיקה האם החלק הנכון שאמור להתחבר אליו בדופן מסוים גם מדווגה היכי גבוהה לפי המטריקה בחיבור זה.

החלק השני של המאמר גם כן מענין אך לא השתמשו בו בפתרון שלנו, בו סורקים את הפתרון שהגיעו אליו עד כה ומוצאים איזוריים בפתרון הכל מוצלחים לפי המטריקה שנבחרה. לוקחים את האזור היכי מוצלח וממשיכים לבנות ממנו פתרון, זה נותן מענה מעולה לאופי של בעית הפazel בה איזוריים מסוימים מרכיבים טובים אך ממוקמים לא נכון בתוך הפazel השלם.

מטריקה נוספת בה השתמשו הייתה ה Best Buddies Metric שאומرت שקיים קשר בין שני חלקים אם עבור חלק X בדופן מסוים הציון הגבוה ביותר הוא של חלק Y ובדופן המנוגד לו עבור חלק Y הציון הגבוה ביותר הוא של חלק X.

במאמר של מחמט ושותפי [5] הציגו פתרון לבעה Permutation Flowshop Sequencing Problem, באמצעות תכניות שונות, בעזרת 3 מעבדים שונים בעוצמתם, כאשר בכל רגע נתון כל מעבד יכול

להריצ' תוכנית אחת בלבד, בעזרת PSO. במליה שלנו נעזרים באלגוריתם VNS אשר הוזג ושותש במאמר זה. במאמר הציגו שני סוגי של PSO, הראשון תוך התחשבות בקיצון מקומי (*gbest*), והשני תוך התחשבות גם בקיצון הגלובלי (*gbest*) באותה האיטרציה. עבור שיטות ה החלפה בין המשימות הוזגו 3 שיטות SPV, GA ו-VNS. ניסויי המאמר הראו כי השיטה המדויקת ביותר הינה VNS אך עם זאת זיהי גם השיטה היקרה ביותר מבחינת חישוביות וזמן ריצה, לשיכום מאמר זה מראה כי PSO ישים גם עבור בעיות אופטימיזציה קומבינטוריות.

## המטרה

במקרה התבקשנו מואחד להרכיב פאלז' בעזרת אלגוריתם PSO, כמו שהצרכנו קודם לכן PSO הוא אלגוריתם שעוצב למציאת נקודת קיצון גלובלי עבור פונקציות רציפות. אנחנו משתמשים בגרסתה שmbattat פרמוטציה (אובייקט DISKRT) באופן הבא:  
לפרמוטציה בגודל  $N$  חלקיקים נחזיק וקטור  $N \times [0, 1]$ , כלומר וקטור רציף בין 0 ל-1 עם  $N$  כניסה.  
כדי לתרגם את הווקטור הרציף לפרמוטציה אנחנו פשוט בודקים מה הסדר של הכניסות מבחינת הגודל.  
לדוגמא, הווקטור הבא  $\{0.5, 0.2, 0.8\}$  יתרגום לפרמוטציה  $\{2, 0, 1\}$ .  
כדי להציג ולשפר את תוצאות האלגוריתם שלנו במליה עבור כל הZZה של חלקיקים במרחב השתמשנו באלגוריתם Neighborhood Search Variable, אלגוריתם VNS הומצא כדי לשפר את מרחב החיפוש למציאת שכנים על ידי איפוס השכנים הנוכחיים ויצירת פיתרון חדש באופן רנדומלי תוך התחשבות בקבוצות השכנים שנמצאו בפיתרון הנוכחי. אלגוריתם VNS הבסיסי מתחילה עם פיתרון רנדומלי וחזר על הערבוב וההZZה עד שתנאי העצירה מתקיים, בדרך כלל תנאי העצירה הוא המקיים של מספר השכנים. בכל צעד של ערבוב, מגרילים שכן אופציוני לפיתרון הנוכחי ובמציעים איפוס מקומי עבור שכן טוב יותר, ולאחר מכן מבצעים את ההZZה. לבסוף משווים את הפיתרון שהתקבל לאחר ההZZה עם הפיתון המקורי לפני הערבוב, ולוקחים את הפיתון בעל הZZן האופטימלי.  
האלגוריתם בו השתמשנו במליה:

```

 $s_0 = \pi^t$  permutation of global best;
 $\eta = \text{rnd}(1,n); \kappa = \text{rnd}(1,n); \eta \neq \kappa$ 
 $s = \text{insert}(s_0, \eta, \kappa);$ 
 $loop = 0;$ 
 $do\{$ 
     $kcount = 0; max-method = 2;$ 
     $do \{$ 
         $\eta = \text{rnd}(1,n); \kappa = \text{rnd}(1,n); \eta \neq \kappa$ 
         $if (kcount = 0) then s_1 = \text{insert}(s, \eta, \kappa)$ 
         $if (kcount = 1) then s_1 = \text{interchange}(s, \eta, \kappa)$ 
         $if (f(s_1) < f(s)) then\{$ 
             $kcount = 0;$ 
             $s = s_1;\}$ 
         $else \{kcount++;}$ 
         $while (kcount < max-method)$ 
     $loop++;$ 
     $while (loop < n * (n - 1))$ 
     $if (f(s) \leq f(\pi^t))\{$ 
         $\pi^t = s;$ 
         $\text{repair}(G^t); \}$ 

```

רעיון נוסף שהעלנו היה להסיר אחז קטן של הנקודות בעלות הציון הנמוך לאחר מספר גדול של איטרציות, הגרלה של מספר פרמטריזיות אקרניות אחרות והוספתן לאלגוריתם, מתוך מחשבה כי סביר להניח שרוב הנקודות אותן הסרנו יגיעו לקיצון המקומי אך אין סביר כי הוא הקיצון הגלובלי מכך שקיימות נקודות רבות אחרות שציון גבואה יותר.

## תיאור האלגוריתם

- קבלת תמונה.
- ניתוח של כל החיבורים האפשריים:
  - ייצרת טבלה המיפה את הציון של החיבור בין כל שני חלקים (אופקי ואנכי).
  - אנחנו משתמשים בנוסחת הציון שהוצגה במאמר של אווד ושותפי [2].
- אתחול PSO:
  - בחירת מספר קלשו של חלקיקים.
  - לכל חלקיק מאותחל וקטור רציף בין 0 ל-1 בגודל N המיצג פרמטריזיה.
- כל חלקיק מחזיק שדה  $g_{best}$  השווה לערך של ציון הפתרון הכי טוב שהחלקיק מצא עד כה.
  - ובנוסף עבור כל החלקיים מחזיקים שדה  $g_{best}$  השווה לערך של החליק בעל הציון הכי גבוה עד כה, שהוא בעצם הפרמטריזיה הכי מוצלת.
- איטרציה של הPSO:
  - עדכון:
    - עדכון של  $g_{best}$

- עדכון של  $h_{best}$  לכל חלקיק
- תזוזה:
- את הוקטור הרציף של כל חלקיק אנחנו מזיזים קצת לכיוון התוצאה טובה ביותר שמצאנו באופן כללי ( $g_{best}$ ) וקצת לכיוון התוצאה הכי טובה שמצאנו באופן מקומי ( $h_{best}$ ). השתמשנו בפרמטרים מהמאמר של מוחמד ושותפיו [5].
- הנוסחה:

$$\bullet \quad x_{ij}^t = x_{ij}^{t-1} + v_{ij}^t.$$

$$v_{ij}^t = w^{t-1}v_{ij}^{t-1} + c_1r_1(p_{ij}^{t-1} - x_{ij}^{t-1}) + c_2r_2(g_j^{t-1} - x_{ij}^{t-1}),$$

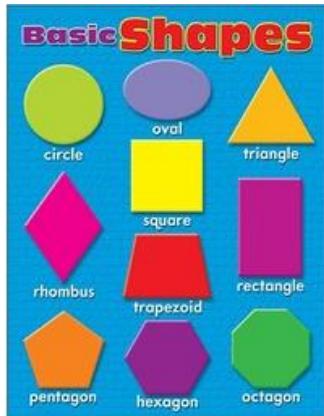
- חיפוש מקומי (SNS):
- עוברים על כל חלקיק
- מבצעים חיפוש מקומי של פתרון טוב יותר בעזרת אלגוריתם SNS.

## תוצאות

התמונות בהן השתמשנו לבחינת האלגוריתם:

.2

.1



השתמשנו בפי ארבע חלקים ממספר החלקים בפאזל. כאמור של מחמת ושותפי [5] הוצע יחס של פי 2 אך מצאנו שיחס של פי ארבע נותן תוצאות טובות יותר בזמן קצר יותר.

כדי לבחון את יעילות האלגוריתם השתמשנו בתמונות עם מספר הולך וגדל של חלקים, ואלו התוצאות:

עבור דוגמא 1:

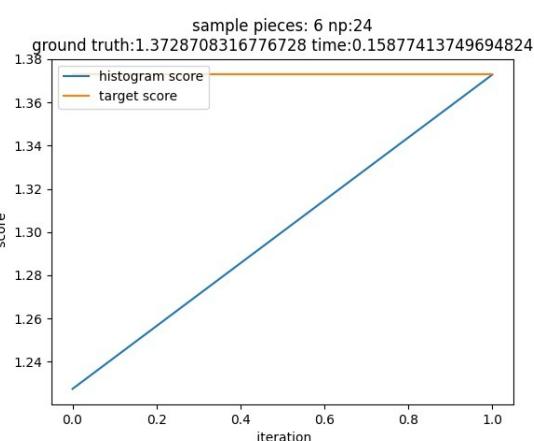
6 חלקים :

אחרי



לפני

גרף התקדמות

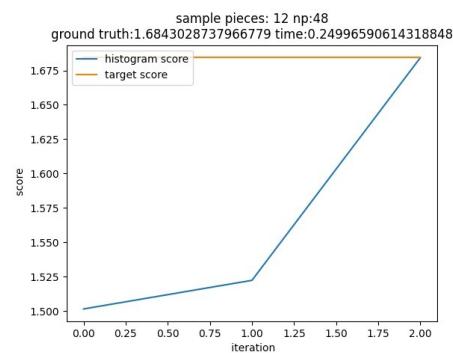


12 חלקים :  
לפני

אחרי



גרף התקדמות

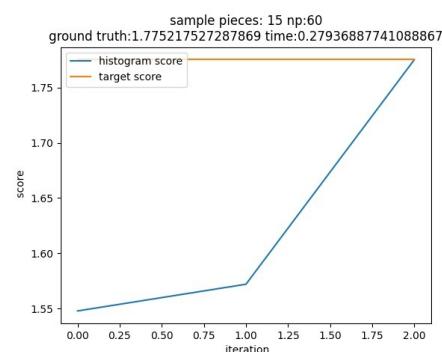


15 חלקים :  
לפני

אחרי



גרף התקדמות

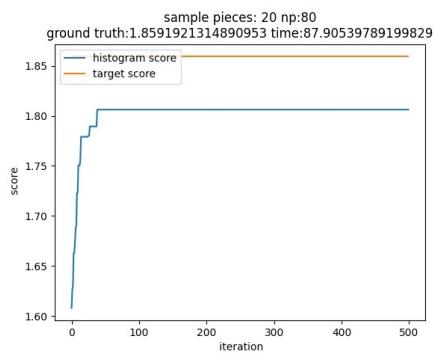


20 חלקים:  
לפני

אחרי

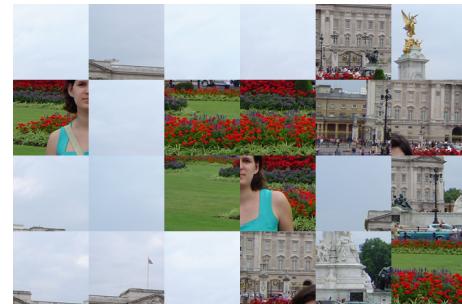
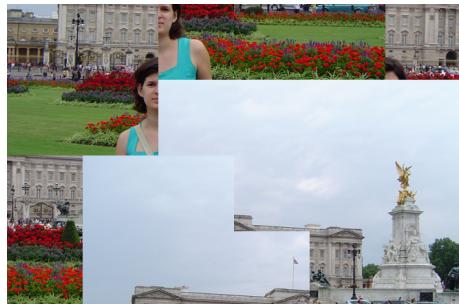


גרף התקדמות

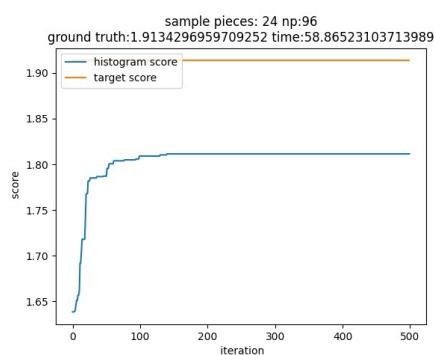


24 חלקים:  
לפני

אחרי



גרף התקדמות

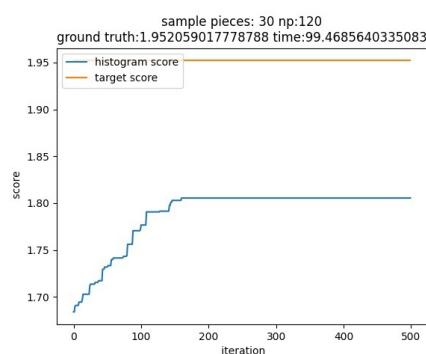


30 חלקים:  
לפני

אחרי

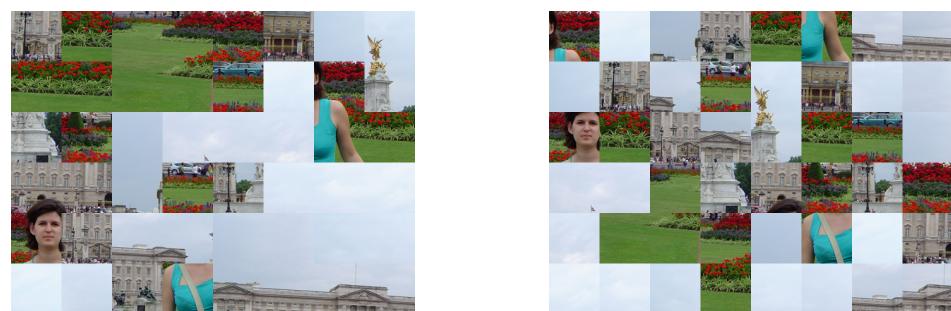


גרף התקדמות

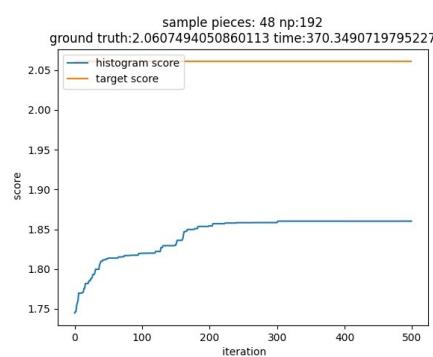


48 חלקים:  
לפני

אחרי



גרף התקדמות

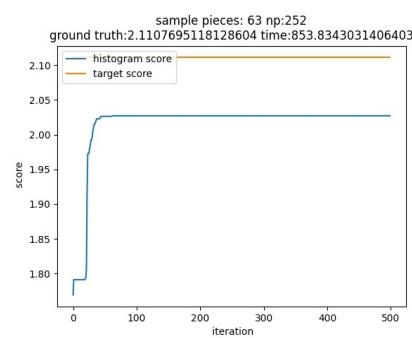


63 חלקים:  
לפני

אחרי



גרף התקדמות

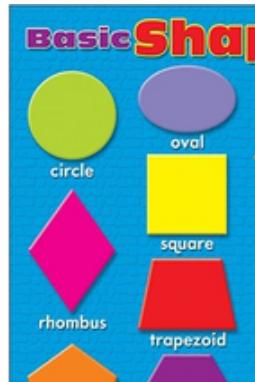


עבור דוגמא 2:

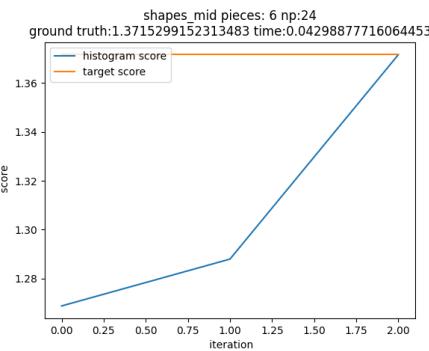
6 חלקים:

לפני

אחרי



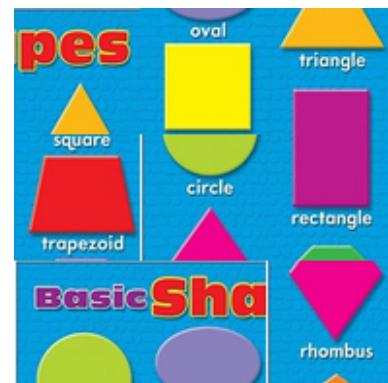
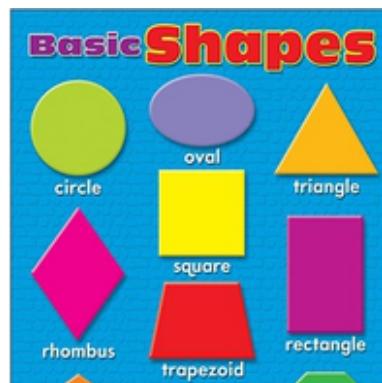
גרף התקדמות



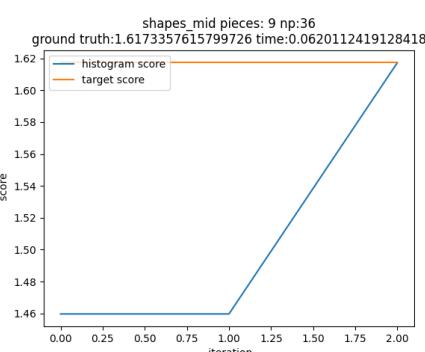
6 חלקים:

לפני

אחרי



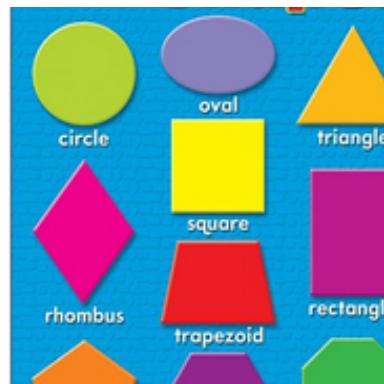
גרף התקדמות



12 חלקים:

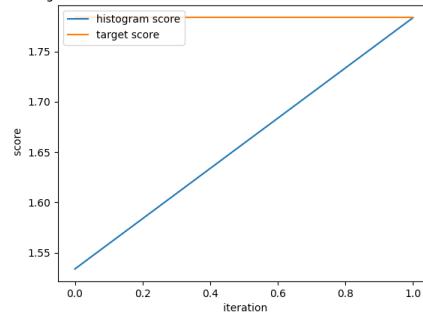
לפני

אחרי



גרף התקדמות

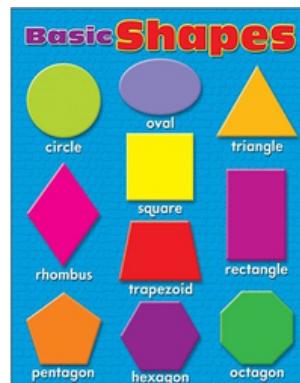
shapes\_mid pieces: 12 np:48  
ground truth:1.7837916827947544 time:0.0670778751373291



20 חלקים:

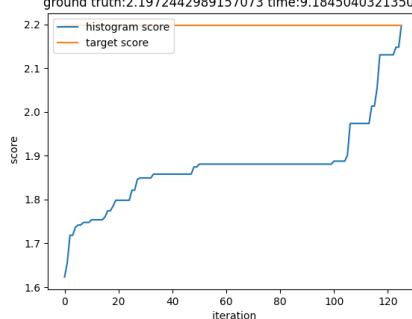
לפני

אחרי



גרף התקדמות

shapes\_mid pieces: 20 np:80  
ground truth:2.1972442989157073 time:9.18450403213501



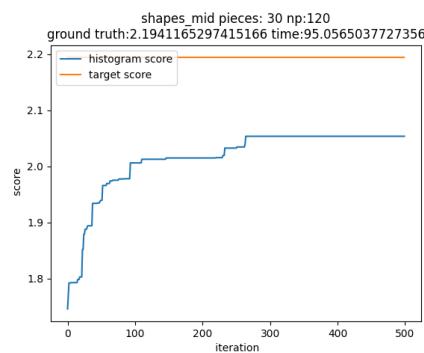
30 חלקים :

לפני

אחרי



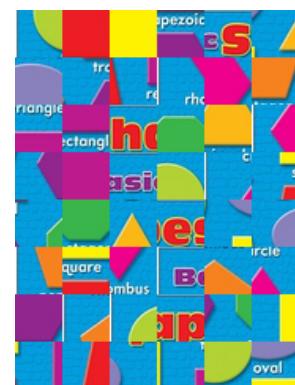
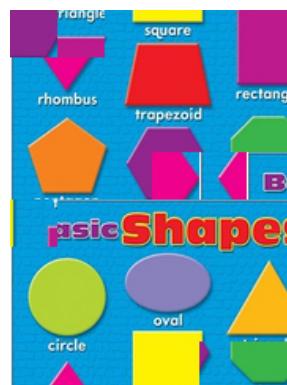
גרף התקדמות



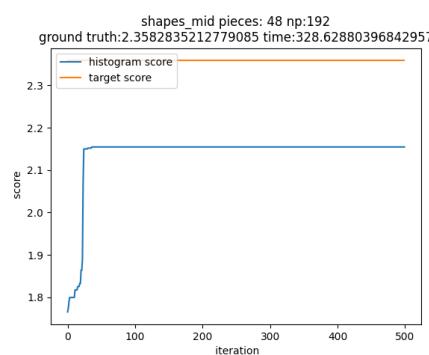
48 חלקים :

לפני

אחרי



גרף התקדמות



### **סיכום תוצאות**

כפי שניתן לראות ככל שמספר החלקים בפאלן גדול כך גם התוצאה פחותה מדויקת, בנוסף ניתן לראות כי לרוב האלגוריתם עוצר במקומות לוקאליים לאחר כמה עשרות איטרציות ואין משתפר גם לאחר מספר רב של איטרציות.

### **הצעות לשיפור**

אומנם הצלחנו לפתור פאלן של 20 חלקים באופן זה, שזו תוצאה לא טריוויאלית, אך עבודות קודמות הצליחו לפתור פאללים של אלפי חלקים. האלגוריתם שלנו משתמש בשיטות כלליות של פתרת פרמטריזציה ללא התייחסות לביעת הפאלן. לדעתנו ללא אלמנט של התייחסות לפאלן, כמו שאוהד ושותפיו עשו במאמר שלהם [2] קשה מאוד להגיע לתוצאות טובות מספיק.

קו מחשבה שהיינו רוצים לנסות הוא להשתמש במטריקה של ה Best Buddies שהוצע במאמר של אוהד ושותפיו [2] כדי לזהות אזורים מוצלחים, לאחד אותם ולהתייחס אליהם כחלק יחיד בפאלן ובכך להקטין את גודל הבעיה שכפי שראינו מביאה לתוצאות מדויקות יותר.

## אזכורים

- [1] T.Altman. Solving the jigsaw puzzle problem in linear time. *Applied Artificial Intelligence*, 3(4):453–462, 1990.
- [2] Ohad Ben-Shahar, Michal Shemesh, Dolev Pomeranz. A fully automated greedy square jigsaw puzzle solver.
- [3] Fernanda A. Andalo, Gabriel Taubin, Siome Goldenstein. PSQP: Puzzle Solving by Quadratic Programming, 2017.
- [4] Kilho Son, James Hays, David B. Cooper. Solving Square Jigsaw Puzzle by Hierarchical Loop Constraints, 2019.
- [5] M. Fatih Tasgetiren , Yun-Chia Liang, Mehmet Şevkli, Gunes Gencyilmaz. A particle swarm optimization algorithm for makespan and total flowtime minimization in the permutation flowshop sequencing problem.