בס"ד

Multi-Objective optimization

Genetics algorithms

מגישים:

ליאב טרבלסי 315870345

הדס בביוב 322807629

בס"ד

<u>Multi-Objective optimization</u> – אופטימיזציה רב מטרתית, הוא תחום של קבלת החלטות מרובות עם בעיות אופטימיזציה מתמטיות הכוללות יותר מפונקציית מטרה אחת שתמוטב בו זמנית.

דוגמאות לתחומים בהם יש שימוש באופטימיזציה מהסוג הנ"ל - הנדסה, כלכלה ולוגיסטיקה.
בתחומים אלו יש לקבל החלטות אופטימליות בנוכחות פשרות בין שתי מטרות סותרות או יותר.

דוגמאות לבעיות עם יותר ממטרה אחת בעת רכישת רכב למשל:
1. מזעור עלויות תוך מקסימום נוחות.
2. מקסום ביצועים תוך מזעור צריכת הדלק ופליטת מזהמים.

לבעיות אלו לא קיים פתרון יחיד שמייעל את כל המטרות. במקרה הזה נאמר כי פונקציות המטרה **סותרות** וקיים מספר (שעלול להיות אינסופי) של פתרונות אופטימליים.

פתרון נקרא **אופטימלי Pareto** אם לא ניתן לשפר את אחת מפונקציות המטרה מבלי לפגוע באחת מפונקציות המטרה האחרות. כל הפתרונות האופטימליים של Pareto נחשבים טובים באותה מידה.
אוסף הפתרונות האופטימליים של Pareto יהיו חסומים ע"י שני וקטורים שיהוו גבול עליון וגבול תחתון עבור ערכי הפונקציה של הפתרונות האופטימליים (כפי שהגדרנו).
וקטור הגבול העליון נקרא nadir's objective vector.
וקטור הגבול התחתון נקרא ideal objective vector.

<u>באופן פורמלי</u>, נרצה להגדיר בעיות מהסוג הזה כך:
- נמזער את כל פונקציות המטרה (אם אחת מהן הייתה מקסימום – נמזער את המינוס שלה).
- אילוצים מסוג קטן שווה (נכפיל במינוס אילוץ מסוג גדול שווה).
- אילוצי שוויון.
- טווח חוקי למשתני ההחלטה.

$$
\begin{array}{lll}
\min & f_m(x) & m = 1,..,M \\
\text{s.t.} & g_j(x) \le 0 & j = 1,..,J \\
 & h_k(x) = 0 & k = 1,..,K \\
 & x_i^L \le x_i \le x_i^U & i = 1,..,N
\end{array}
$$

<u>דוגמא מתמטית</u> לבעיה עם מספר מטרות:

maximize F1 = x1

maximize F2 = 3x1 + 4x2

constraints: x1 <= 20

x2 <= 40

5x1 + 4x2 <= 200

אז כפי שניתן לראות – בבעיה זו נרצה למקסם <u>שתי פונקציות שונות</u>, בעלות שני משתנים.
לבעיה זו קיימים שלושה אילוצים.

דוגמאות לשימושים באופטימיזציה מהסוג הנ"ל בעולמות השונים:

- **כלכלה** – בתחום הזה, בעיות רבות מורכבות ממטרות מרובות. למשל הגבלת ייצור בהתאם לכמות המשאבים של חברה מסוימת עם רצון למקסם רווח ולמזער הוצאות. דוגמא נוספת, ניהול פעולות בשוק הפתוח כך שגם שיעור האינפלציה וגם שיעור האבטלה יהיו קרובים ככל האפשר לערכים הרצויים שלהם (קירוב הערך למטרה).
- **הנדסה** – נרצה לקרב את הערך לערך האופטימלי של המטרה, לדוגמא – צריכת דלק, כיוון של רקטה.

אופציות לפתרון בעיות עם מספר מטרות:

כפי שאמרנו, צפוי להיות יותר מפתרון אופטימלי יחיד כיוון שמדובר בהתעסקות עם מספר מטרות סותרות. ראשית, נרצה למצוא את אוסף הפתרונות האופטימליים Pareto, ולאחר מכן נבחר את הפתרון הטוב ביותר עבורנו (לפי העדפה כלשהי).

שיטות למציאת אוסף הפתרונות האופטימליים:

- **Simulated annealing** – שיטה סטוכסטית, שגם אותה למדנו בכיתה, הורחבה לאופטימיזציה עם מספר מטרות כדי למצוא את אוסף פתרונות האופטימליים המקורבים.

- **Genetics algorithm** – כפי שלמדנו בכיתה. ההבדל יהיה במיוחדות פונקציית הכושר וההערכה של צאצאים, כיוון שכעת יש לנו כמה פונקציות מטרה שונות.

כעת, נפרט מספר אופציות למציאת פתרון האופטימלי (לפי העדפות) מתוך אוסף הפתרונות האופטימליים:

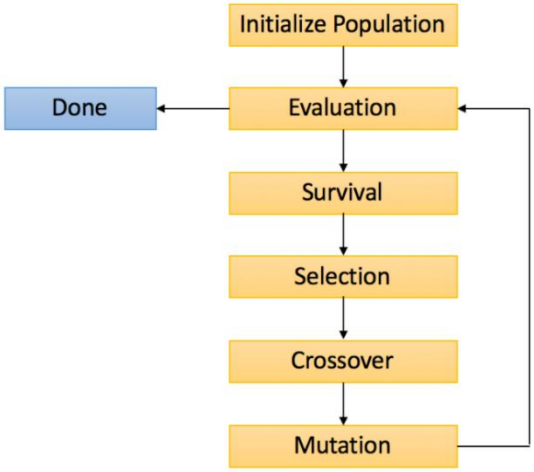
- **Aphrodiar methods** – ראשית נבדוק מהן ההעדפות, ולאחר מכן נמצא את הפתרון שעונה בצורה הטובה ביותר על ההעדפות הללו.

- **Backdoor methods** – ראשית נמצא קבוצת פתרונות Pareto אופטימליים, ולאחר מכן, לפי העדפות, מקבל החלטות אנושי יבחר את הפתרון שעונה בצורה הטובה ביותר על ההעדפות הנ"ל.

- **Interactive methods** – מקבל ההחלטות יחפש בצורה איטרטיבית את הפתרון הטוב ביותר. מקבל ההחלטות יכול להתעסק בפתרונות שמעניינים אותו ורשאי להפסיק את החיפוש בכל זמן שירצה.

בפרויקט שלנו, נרצה לחקור את פתרון הבעיות הללו באמצעות אלגוריתמים גנטיים.
אז כפי שלמנו בכיתה, האלגוריתם הגנטי מתבצע באופן הבא:

- אתחול אוכלוסייה (דגימה)
- הערכה (באמצעות פונקציה שהוגדרה בהתאם לבעיה)
- הישרדות
- בחירת פריטים להזדווגות
- הזיווג עצמו, נולדים צאצאים
- מוטציות, ליצירת גיוון באוכלוסייה



האלגוריתמים הגנטיים, כפי שלמדנו בכיתה, יועדו לבעיית אופטימיזציה בעלת פונקציית מטרה אחת.
באלגוריתם הנ"ל, הייתה פונקציית הערכה, שמשערת את טיב הצאצא. נרצה שהחזקים ביותר ישרדו.
כשיש לנו פונקציית מטרה אחת – נרצה להעריך את טיב כל הצאצאים ולגרום לצאצאים החזקים ביותר להתרבות. אז נפעיל את פונקציית ההערכה, ונמיין אותם לפי הסדר.
אך מה קורה שיש יותר מפונקציית מטרה אחת? כיצד נעריך את טיב הצאצאים? אז כמובן שיש מספר שיטות להתמודד עם כך.

<u>בהינתן elitism, קיימות שתי גישות לפונקציות הערכה בMulti-objective:</u>

Non-Elitist MOEAs

- Vector evaluated GA
- Vector optimized ES
- Weight based GA
- Random weighted GA
- Multiple-objective GA
- Non-dominated Sorting GA
- Niched Pareto GA

Elitist MOEAs

- Elitist Non-dominated Sorting GA
- Distance-based Pareto GA
- Strength Pareto GA
- Pareto-archived ES

כפי שניתן לראות, ישנן שיטות מבוססות מרחקים, משקלים וכו. ניתן לשים דגש על דברים שונים בהתאם להגדרות הבעיה ובהתאם לרצון של הממ"מ.

בס"ד

כעת, נעבור לדוגמא שנממש בpython.

נרצה לפתור את בעיית הMulti-objective הבאה באמצעות שימוש באלגוריתם גנטי:

$$\begin{aligned} \min \quad & f_1(x) = (x_1^2 + x_2^2) \\ \max \quad & f_2(x) = -(x_1 - 1)^2 - x_2^2 \end{aligned}$$

$$\begin{aligned} \text{s.t.} \quad & g_1(x) = 2\,(x_1 - 0.1)\,(x_1 - 0.9)\,/\,0.18 \le 0 \\ & g_2(x) = -20\,(x_1 - 0.4)\,(x_1 - 0.6)\,/\,4.8 \le 0 \\ & -2 \le x_1 \le 2 \\ & -2 \le x_2 \le 2 \end{aligned}$$

נרצה למזער את שתי פונקציות המטרה ← את הפונקציה השנייה נכפיל במינוס 1 ובמקום max נעשה min.

סה"כ נקבל את הבעיה השקולה הבאה:

$$\begin{aligned} \min \quad & f_1(x) = (x_1^2 + x_2^2) \\ \min \quad & f_2(x) = (x_1 - 1)^2 + x_2^2 \end{aligned}$$

$$\begin{aligned} \text{s.t.} \quad & g_1(x) = 2\,(x_1 - 0.1)\,(x_1 - 0.9)\,/\,0.18 \le 0 \\ & g_2(x) = -20\,(x_1 - 0.4)\,(x_1 - 0.6)\,/\,4.8 \le 0 \\ & -2 \le x_1 \le 2 \\ & -2 \le x_2 \le 2 \end{aligned}$$

כעת, נרצה למצוא את אוסף הפתרונות האופטימליים ולהציגם באמצעות גרף.

ראשית, נגדיר את הבעיה -

מדובר בבעיה בעלת שני משתנים (x_1, x_2),

שתי פונקציות מטרה, שני אילוצים,

והתחום של כל משתנה החלטה הוא $[-2,2]$.

```python
class MyProblem(Problem):
    def __init__(self):
        super().__init__(n_var=2,
                         n_obj=2,
                         n_constr=2,
                         xl=np.array([-2, -2]),
                         xu=np.array([2, 2]))
```

נגדיר את פונקציית ההערכה כך -

מקבלת X בגודל מספר הפריטים באוכלוסייה

על מספר משתני ההחלטה.

היא מבצעת את החישובים ומוסיפה

את הערכים למילון לפי מפתחות

(F- מטרה, G- אילוצים).

מתודה זו תיקרא בדיוק פעם אחת בכל איטרציה,

תבצע הערכה של כל אינדיבידואל באוכלוסייה, ותשמור את החישובים במילון out.

```python
def _evaluate(self, X, out, *args, **kwargs):
    f1 = X[:, 0] ** 2 + X[:, 1] ** 2
    f2 = (X[:, 0] - 1) ** 2 + X[:, 1] ** 2

    g1 = 2 * (X[:, 0] - 0.1) * (X[:, 0] - 0.9) / 0.18
    g2 = - 20 * (X[:, 0] - 0.4) * (X[:, 0] - 0.6) / 4.8

    out["F"] = np.column_stack([f1, f2])
    out["G"] = np.column_stack([g1, g2])
```

המשכנו והגדרנו בקוד את האילוצים והמטרות –

```python
objectives = [
    lambda x: x[0] ** 2 + x[1] ** 2,
    lambda x: (x[0] - 1) ** 2 + x[1] ** 2
]
```

```python
constrains = [
    lambda x: 2 * (x[0] - 0.1) * (x[0] - 0.9) / 0.18,
    lambda x: - 20 * (x[0] - 0.4) * (x[0] - 0.6) / 4.8
]
```

בס"ד

לאחר מכן, הגדרנו את פרטי האלגוריתם –
גודל האוכלוסייה – 40.
נייצר 10 צאצאים (ולא 40, זה מסייע להתכנסות מהירה יותר עבור בעיות מהסוג הנ"ל).

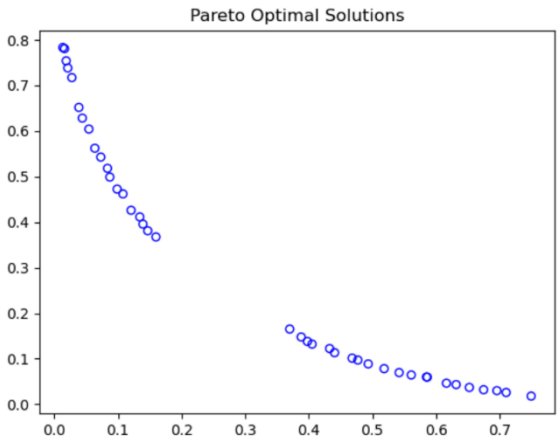
```
algorithm = NSGA2(
    pop_size=40,
    n_offsprings=10,
    sampling=get_sampling("real_random"),
    crossover=get_crossover("real_sbx", prob=0.9, eta=15),
    mutation=get_mutation("real_pm", eta=20),
    eliminate_duplicates=True
)
```

במשתנה האחרון (הבוליאני), אנו מאפשרים בדיקה כפולה, מוודאים שההזדווגות מייצרת צאצאים שונים מהאוכלוסייה הקיימת.
בשאר המשתנים (הקשורים לזיווג ולמוטציות), מדובר בערכי ברירת המחדל.

כעת, נפעיל את האלגוריתם. נרצה למזער את פונקציות המטרה שברשותנו –
נשלח את הבעיה והאלגוריתם שהגדרנו לפני כן.
נגדיר שסיום האלגוריתם יתרחש לאחר 40 דורות.
יתאפשר שיחזור תוצאות.
ההיסטוריה תישמר באובייקט התוצאה.
verbose – הדפסת תוצאות תוך כדי ריצה.

```
res = minimize(problem,
               algorithm,
               get_termination("n_gen", 40),
               seed=1,
               save_history=True,
               verbose=True)
```

בשלב הזה, אנו מחזיקים את הפתרון לבעיה (בעת הרצת הקוד, נתוני הפתרון מודפסים למסך). נרצה למחיש אותו בצורה ויזואלית.
כפי שהסברנו, מדובר באוסף פתרונות אופטימליים, לכן נצפה לראות יותר מנקודה אחת בודדת שמהווה את האופטימום.

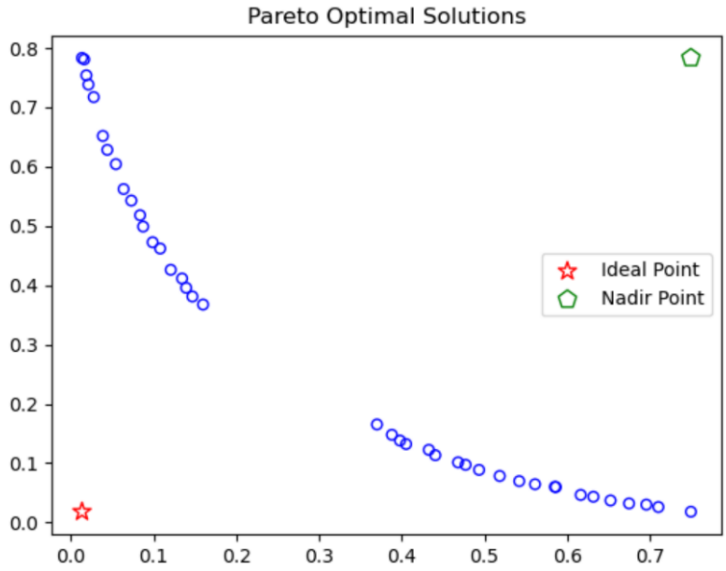


כפי שציפינו, קיבלנו אוסף פתרונות אופטימליים. כעת, לפי העדפות ושיקולי דעת, נוכל לבחור מתוך הקבוצה הזו את הפתרון הטוב ביותר עבורנו.

ראינו, כי קיימים שני וקטורים שמהווים חסם עליון ותחתון לאוסף הפתרונות האופטימליים.
נחשב קירוב לנקודת החסם העליון ונקודת החסם התחתון –

```
approx_ideal_value = res.F.min(axis=0)
approx_nadir_value = res.F.max(axis=0)
```

כעת, נרצה להוסיף לגרף את שתי הנקודות הללו.
נקבל:



כלומר, nadir's objective vector יעבור דרך הנקודה הירוקה ויהווה את החסם התחתון לאוסף הפתרונות האופטימליים.
באופן דומה, ideal objective vector יעבור בנקודה האדומה ויהווה את החסם העליון לאוסף הפתרונות האופטימליים.

בס"ד

סיכום, מסקנות ומה למדנו מהפרויקט:

הגדרנו מהי בעיית Multi-Objective. ראינו כי מספר מטרות עלולות לסתור אחת את השנייה ולכן לא יהיה פתרון אופטימלי יחיד.
זה הוביל אותנו להגדרת פתרון אופטימלי Pareto, שאומר שפתרון נחשב אופטימלי אם לא ניתן לשפר את אחת מפונקציות המטרה מבלי לפגוע באחת אחרת. ראינו כי קיימים שני ישרים שחוסמים את אוסף הפתרונות מלעיל ומלרע.
אלו בעיות שימושיות מעולמות ההנדסה והכלכלה (ראינו דוגמאות).

נחשפנו לשתי שיטות למציאת אוסף הפתרונות האופטימליים – אלגוריתמים גנטיים ו- simulated annealing. בחרנו להתעסק באלגוריתמים הגנטיים.
לאחר מציאת אוסף הפתרונות, יש מספר שיטות למציאת הפתרון היחיד – בהתחשב בהפעלת שיקול דעת ובהעדפה מסוימת.

נזכרנו באלגוריתם הגנטי שלמדנו בכיתה וראינו כי בבעיה מרובת מטרות, נרצה פונקציית הערכה שמתחשבת בכל המטרות, ולפיהן מעריכה את טיב הצאצאים.

עברנו לדוגמא קונקרטית שעזרה להבין איך הדברים קורים בפועל. מימשנו את הבעיה בpython באמצעות הספרייה pymoo שתומכת בין היתר באלגוריתמים גנטיים.

התחלנו עם בעיה מסוימת, המרנו אותה להיות מוצגת בצורה שתהיה נוחה לשימוש והתחלנו במימוש.
כמובן שמאחורי הקלעים הספרייה מבצעת את כל השלבים שהזכרנו – אתחול אוכלוסייה, הערכה, הישרדות, בחירת פריטים להזדווגות, צאצאים ומוטציות.

בסופו של דבר, קיבלנו את אוסף הפתרונות האופטימליים והצגנו אותם בגרף.
הזכרנו שניתן לבחור פתרון אחד מתוך האוסף על ידי הפעלת שיקול דעת בצורה איטרטיבית למשל.

בסוף התהליך מצאנו קירוב לnadir and ideal points והצגנו גם אותן בגרף.

סה"כ, הצלחנו לחקור ולפתור לעומק את בעיית הMulti-Objective.

ניתן להוריד את הקוד מה-github בקישור כאן.
הוראות כיצד להריץ ואיזה ספריות להוריד מופיעות בREADME.