* שאלה 1: התחכמות בטוחה באלגוריתם נאש

למדנו שאלגוריתם נאש (מיקסום המכפלה) לחלוקת תקציב רציף אינו מגלה-אמת. אבל כמה מידע בדיוק צריך כדי לחשב התחכמות בטוחה? [תזכורת: התחכמות בטוחה היא דיווח לא-אמיתי, שבמקרים מסוימים יכול להגדיל את תועלת השחקן, ובאף מקרה לא יכול להקטין אותה].

האם לשחקן, שאין לו שום מידע על פעולות השחקנים האחרים, יכולה להיות התחכמות בטוחה? אם כן – הראו דוגמה והוכיחו שההתחכמות בטוחה. אם לא – הוכיחו שבשום מקרה אין התחכמות בטוחה.

<u>תשובה:</u>

נראה שאם לשחקן אין מידע כלל על פעולות השחקנים האחרים – לא קיימת לו התחכמות בטוחה.

<u>הוכחה:</u>

נניח שקיימת לשחקן ה i התחכמות בטוחה, כאשר קבוצת ההעדפות שלו היא A, והוא מצהיר על הקבוצה A'. בגלל שזאת התחכמות בטוחה שהפלט של אלגוריתם שייתן חלוקה יעילה נאש ייתן שהתועלת אחרי ההתחכמות תעלה.

נניח שכל שאר שההעדפות של שאר השחקנים זהות להעדפות האמיתיות של השחקן הi.

<u>במקרה שi ידווח אמת</u>: ההעדפות של כולם זהות, ולכן מיקסום המכפלות יקצה את כל התקציב לכל הנושאים בקבוצה A, ולכן התועלת של השחקן הi תהיה C (התקציב כולו).

<u>במקרה שi ינסה לעשות מניפולציה:</u> אז הוא חייב להצביע לנושא שהוא לא מעדיף (כי ההעדפה שונה והיא בינארית). בגלל שאף שחקן אחר לא מעדיף את אותו הנושא, אז כדי למקסם את המכפלה, כדי לא לכפול ב0 חייב שהנושא הזה יקבל תקציב כלשהו, בגלל שהוא לא נושא שi רוצה שהתועלת שלו תרד.

ולכן במקרה הזה לא קיימת התחכמות בטוחה.

■

שאלה 2: זכויות לפי גובה המס

נניח שאזרח i משלם מס בגובה C_i (התקציב C שווה לסכום המיסים שמשלמים כל האזרחים). אנחנו רוצים להגדיר מערכת לחלוקת תקציב, שתתן זכויות רבות יותר לאזרחים המשלמים יותר מיסים. לדוגמה, ההגדרה של תכונת "הוגנות ליחידים" תהיה: "התועלת של אזרח i היא לפחות C_i".

א. כתבו הגדרה מוכללת של תקציב הוגן לקבוצות, ושל תקציב פריק.

ב. הוכיחו, שכל תקציב פריק הוא הוגן לקבוצות – בהתאם להגדרות של סעיף א.

ג. נגדיר "תקציב נאש מוכלל" כתקציב d הממקסם את הסכום:

$$\text{Sum}[i=1,\dots,n]\ C_i * \log(u_i(d))$$

(מכפילים את הלוג של אזרח i בגובה המס ששילם אזרח i).

הוכיחו שתקציב נאש מוכלל הוא פריק (לפי ההגדרה של סעיף א).

הדרכה: הכלילו את ההוכחות מההרצאה.

א. הרעיון יהיה שבמקום שיהיה לו חלק פרופורציונלי לתקציב $(C\backslash n)$ כל אחד יקבל C_i.

הגדרה מוכללת לתקציב הוגן לקבוצות: לכל קבוצה K בגודל k הסכום הכולל המועבר לנושאים שאחד מחברי הקבוצה תומך בהם הוא לפחות $\sum_{i\in K} C_i$.
באופן קצת יותר פורמלי, עבור הקובצה K, אם A היא קבוצת הנושאים שנתמכים ע"י אחד מחברי הקבוצה אז מתקיים:

$$\sum_{j\in A} d_j \geq \sum_{i\in K} C_i$$

הגדרה מוכללת של תקציב פריק: תקציב $d_1,\dots,d_m$ נקרא פריק אם קיימים סכומים $d_{i,j}$ לכל אזרח i וכל נושא j כך ש:

לכל נושא j מתקיים: $\sum_{i=1}^{n} d_{i,j} = d_j$

לכל אזרח i מתקיים: $\sum_{i=1}^{m} d_{i,j} = C_i$

ומתקיים $d_{i,j} > 0 \Leftrightarrow u_{i,j} > 0$

ב. נוכיח את הטענה:

נניח שהתקציב d הוא פריק באופן המוכלל, אז קיים לו פירוק שמקיים את הנ"ל. ניקח קבוצת אזרחים בגודל k, נקרא לקבוצה K. נסמן בA את קבוצת כל בנושאים שלפחות אחד מחברי הקבוצה תומך בהם.
התקציב הכולל המוקצה לנושאים האלו הוא $\sum_{j\in A} d_j$. לפי הגדרת פריקות מתקיים $\sum_{i=1}^{n} d_{i,j} = d_j$, כלומר

$$\sum_{j\in A} d_j = \sum_{j\in A}\sum_{i=1}^{n} d_{i,j}$$

נפצל את הסכום הפנימי לשני חלקים – אזרחים (באינדקס i) שנמצאים בקבוצה, וכאלה שלא.

$$\sum_{j\in A} d_j = \sum_{j\in A}\left(\sum_{i\in K} d_{i,j} + \sum_{i\notin K} d_{i,j}\right) = \sum_{j\in A}\sum_{i\in k} d_{i,j} + \sum_{j\in A}\sum_{i\notin k} d_{i,j}$$

בנוסף לפי ההגדרה המוכללת של פריקות, אזרח i תורם רק לנושאים שהוא תומך בהם (התנאי השלישי), כלומר

$$\sum_{j\in A} d_{i,j} = \sum_{i=1}^{m} d_{i,j} = C_i$$

ולכן האיבר הראשון בסכום הוא (אחרי החלפת סדר הסכימה):

$$\sum_{i\in K} C_i$$

האיבר השני בסכום מייצג את התרומות של אזרחים שלא בK לנושאים שנתמכים ע"י אחד מחברי K. בגלל שהסכום הזה הוא לא שלילי (הרי התקציבים שלנו חיוביים) אז נקבל:

$$\sum_{j\in A} d_j \geq \sum_{i\in K} C_i$$

ג.

שאלה 3:

כדי למצוא את הפירוק ניעזר בהוכחה מהסיכום:

להוכחת הכיוון השני, נבנה רשת זרימה המייצגת את התקציב. רשת זרימה (flow network) היא גרף מכוון, עם צומת מקור המסומן ב-s, וצומת יעד המסומן ב-t. לכל קשת יש מספר חיובי המציין את הקיבולת שלה. הזרם יוצא מהמקור s ומתפצל בין הקשתות, כך שהזרם העובר בכל קשת קטן או שווה לקיבולת שלה.סכום הזרם הנכנס לכל צומת (פרט ל-s ו-t) חייב להיות שווה לסכום הזרם היוצא ממנה. בהינתן תקציב כלשהו d, נבנה רשת-זרימה באופן הבא:

- כל אחד מהאזרחים i הוא צומת. יש קשת מכוונת מהמקור s אל כל אזרח i; קיבולת הקשת היא C/n – החלק ההוגן של שחקן i.
- כל אחד מהנושאים j הוא צומת. יש קשת מכוונת מכל אזרח i אל כל נושא j שהוא תומך בו ($u_{i,j}=1$); קיבולת הקשת היא C/n.
- יש קשת מכוונת מכל נושא j אל היעד t; קיבולת הקשת היא d_j.

אם בגרף זה קיימת זרימה, שגודלה הכולל הוא C (הגודל המירבי האפשרי), אז התקציב d הוא פריק, כאשר כמות הזרם העובר על כל קשת מאזרח i לנושא j הוא הרכיב $d_{i,j}$ בפירוק.

קישור לקוד בgist github: https://gist.github.com/ShayGali/9f7a94ecfccf90047e5c1662c050bdc5

גילוי נאות – אני עובד עם copilot שעזר לי להשלים את הקוד, ולדעת איך לעבוד עם התקייה, ועם ההערות של הקוד.

```python
import networkx as nx
from typing import List, Set, Optional


def find_decomposition(
    budget: List[float],
    preferences: List[Set[int]],
) -> Optional[List[List[float]]]:
    """
    Check whether a given continuous budget vector is decomposable, and
    return one valid decomposition if it exists.

    Parameters
    ----------
    budget : list[float]
        The amount d_j allocated to each project j.  The sum equals C.
    preferences : list[set[int]]
        For every citizen i, the set A_i of projects she approves.

    Returns
    -------
    None | list[list[float]]
        * None              - the budget is **not** decomposable.
        * list of lists     - an n X m matrix d_{i,j}; each row i sums
          to C/n, each column j sums to d_j, and d_{i,j} > 0 only if
          j ∈ A_i.
    """

    n = len(preferences)
    m = len(budget)
    C = sum(budget)
```

```python
    # Each citizen's "fair share" according to the assignment text
    share = C / n

    # Build a flow network:
    #   source  ->  citizens  ->  projects  ->  sink
    G = nx.DiGraph()
    src, sink = "s", "t"

    # Source → citizen edges: capacity
    for i in range(n):
        G.add_edge(src, f"i{i}", capacity=share)

    # Citizen → project edges: capacity = share
    for i, proj in enumerate(preferences):
        for j in proj:
            G.add_edge(f"i{i}", f"p{j}", capacity=share)

    # Project → sink edges: capacity = budget assigned to the project
    for j, dj in enumerate(budget):
        G.add_edge(f"p{j}", sink, capacity=dj)

    # Compute the maximum flow
    flow_value, flow_dict = nx.maximum_flow(G, src, sink)

    # If the max flow is smaller than C (up to numerical tolerance),
    # the budget cannot be decomposed.
    if abs(flow_value - C) > 1e-6:
        return None

    # Extract the decomposition matrix d_{i,j} from the flow result
    decomposition = [
        [flow_dict.get(f"i{i}").get(f"p{j}", 0.0) for j in range(m)]  # 0 if no edge
        for i in range(n)
    ]

    return decomposition


# ---- Example usage ----
if __name__ == "__main__":
    budget = [400, 50, 50, 0]  # d_j (must sum to C = 500)
    preferences = [{0, 1}, {0, 2}, {0, 3}, {1, 2}, {0}]  # A_i sets

    d = find_decomposition(budget, preferences)
    if d is None:
        print("The budget is NOT decomposable.")
    else:
        print("A valid decomposition matrix d_{i,j}:")
        for i, row in enumerate(d):
            print(f"Citizen {i}: {row}")
```