פרק 5 – חלוקה מיטבית של חפצים בדידים

"אָז תָּבִין צֶדֶק וּמִשְׁפָּט וּמֵישָׁרִים, כָּל מַעְגַּל טוֹב"
(משלי ב ט)

5.1 מבוא לפרק

בפרק הקודם ראינו אלגוריתמים למציאת חלוקה של חפצים בדידים שהיא "כמעט" הוגנת. אכן, במקרה הגרוע אי־אפשר להבטיח יותר מזה, למשל כשיש מספר איזוגי של חפצים זהים. אבל במקרים רבים אפשר להשיג חלוקות טובות יותר. השאיפה שלנו בפרק זה תהיה למצוא את החלוקה הטובה ביותר האפשרית, בהתחשב בנתונים.

בסעיף הבא נעסוק ב**בעיית החלטה**: בהינתן חפצים ושחקנים, האם קיימת חלוקה הוגנת לגמרי (ולא רק בקירוב)?

בסעיפים שאחריו נעסוק ב**בעיית מיטוב** (אופטימיזציה, optimization): מציאת חלוקה שבה פונקציה מסויימת מקבלת ערך גדול ביותר, מבין כל החלוקות האפשריות. נעסוק בשלוש פונקציות־המטרה שתיארנו בפרק 3, בהקשר של חלוקה יעילה של משאבים רציפים:

- **הערך האוטיליטרי** (utilitarian) – סכום הערכים שמקבלים השחקנים בחלוקה;
- **ערך נאש** (Nash) – מכפלת הערכים שמקבלים השחקנים בחלוקה;
- **הערך האגליטרי** (egalitarian) – הערך הקטן ביותר שמקבל שחקן בחלוקה.

5.2 בדיקה האם קיימת חלוקה הוגנת

בעיית חלוקת־המספרים (באנגלית: Partition או Number Partitioning) היא בעייה שאפשר לראות בה מקרה פרטי של בעיית החלוקה ההוגנת: יש רק שני שחקנים, יש להם **הערכות זהות**, וסכום הערכים של כל החפצים הוא 2V, כאשר V הוא מספר שלם כלשהו. במצב זה, חלוקה היא הוגנת (פרופורציונלית וללא־קנאה) אם־ורק־אם כל שחקן מקבל ערך V בדיוק. המטרה היא לבדוק האם קיימת חלוקה הוגנת. ההגדרה המקובלת לבעיית חלוקת־המספרים היא פשוטה יותר, ואינה מתייחסת כלל לשחקנים או להוגנות אלא למספרים בלבד:

הגדרה 1.5: בעיית חלוקת־המספרים

הקלט: רשימה של מספרים שלמים חיוביים שסכומם 2V.
הפלט: "כן" אם קיימת חלוקה של המספרים לשתי קבוצות שסכומן V; אחרת "לא".

בעיית חלוקת־המספרים היא בעייה קשה חישובית: כבר בשנת 1972, הוכיח ריצ'ארד קארפ שהבעיה הזאת היא NP־קשה,[1] ולכן – לפי ההנחה המקובלת על רוב מדעני המחשב בימינו – לא קיים לה פתרון בזמן פולינומיאלי. מכאן, שגם בעיית החלוקה ההוגנת – שהיא בעיה כללית יותר – היא NP־קשה:

משפט 2.5

א. הבעיה הבאה – מציאת חלוקה ללא־קנאה – היא NP־קשה:
הקלט: קבוצה של n שחקנים עם הערכות חיבוריות על אוסף חפצים בדידים;
הפלט: "כן" אם קיימת חלוקה ללא־קנאה; אחרת "לא".
ב. גם בעיית מציאת חלוקה פרופורציונלית, המוגדרת באופן אנלוגי, היא NP־קשה.

קושי חישובי אינו סיבה לייאוש. ישנן כמה דרכים המאפשרות לפתור את הבעיה בזמן סביר, לפחות במקרים קטנים יחסית.

5.2.1 תיכנות ליניארי בשלמים

מבחינה מעשית, הדרך הפשוטה ביותר לפתרון בעיית החלוקה היא **תיכנות ליניארי בשלמים** (integer linear programming). התוכניות שנכתוב דומות לתוכניות שראינו בפרק 3 לפתרון בעיות חלוקת משאבים. ההבדל העיקרי הוא, שבחלוקת חפצים בדידים, יש להוסיף דרישה שהמשתנים הם מספרים שלמים.

הקלט: מטריצה **v**. לכל שחקן i ולכל חפץ g, הערך $v_{i,g}$ במטריצה מציין את הערך של חפץ g בעיני שחקן i.
המשתנים: לכל שחקן i ולכל חפץ g, נגדיר משתנה $x_{i,g}$ המציין את האחוז של חפץ g הניתן לשחקן i.

כדי לבדוק אם קיימת חלוקה פרופורציונלית, נפתור את התוכנית הליניארית הרשומה למטה.

$$\sum_i x_{i,g} = 1 \qquad \text{for every item } g$$

$$0 \le x_{i,g} \le 1 \qquad \text{for all } i, g$$

$$x_{i,g} \in Z \qquad \text{for all } i, g$$

$$v_i(x_i) \ge \frac{v_i(all)}{n} \qquad \text{for every player } i$$

שימו לב: בתוכנית זו אין פונקציית־מטרה – יש רק אילוצים. ישנן ארבע קבוצות של אילוצים:

- הקבוצה הראשונה קובעת, שכל חפץ מחולק פעם אחת בדיוק.
- הקבוצה השניה קובעת, שאחוזי-הבעלות על חפץ כלשהו חייבים להיות בין 0 ל־1.
- הקבוצה השלישית קובעת, שאחוזי-הבעלות על חפץ כלשהו חייבים להיות מספר שלם. אלה האילוצים המייחדים את בעיית חלוקת החפצים הבדידים. יחד עם האילוצים מהקבוצה השניה, נובע שהמשתנים חייבים להיות 0 או 1, כלומר, כל חפץ נמסר בשלמותו לשחקן אחד.
- הקבוצה הרביעית קובעת, שהחלוקה היא פרופורציונלית – סכום הערכים שמקבל כל שחקן הוא לפחות n/1 מסכום הערכים הכללי שלו. בשורה זו השתמשנו בסימון v $_i(x_i)$ כקיצור לסכום: $sum_g\ v_{i,g}*x_{i,g}$, המייצג את הערך של שחקן i בחלוקה x. זו פונקציה ליניארית של המשתנים $x_{i,g}$, ולכן ניתן להשתמש בו בתוכנית ליניארית.

אפשר לפתור תוכנית כזאת בעזרת חבילות־תוכנה לפתרון בעיות־מיטוב ליניאריות, המאפשרות להגדיר משתנה כמספר שלם (integer). חבילות כאלו יודעות לקבל אוסף של אילוצים, ולהחליט האם קיים פתרון כלשהו העומד באילוצים. אם התשובה היא "כן", אז

החבילות גם מחזירות את הפתרון. במקרה שלנו, הפתרון הוא המשתנים $x_{i,g}$. בעזרת המשתנים הללו ניתן לחשב את החלוקה הפרופורציונלית, אם היא קיימת. כיוון שהבעיה NP־קשה, במקרה הגרוע הפתרון עשוי לקחת זמן מעריכי בגודל הקלט. אבל כאשר הבעיה מספיק קטנה, הפתרון בדרך־כלל מהיר.

בחנו את עצמכם 4.5

כתבו תוכניות ליניאריות בשלמים לפתרון הבעיות הבאות:

א. בדיקה האם קיימת חלוקה ללא־קנאה.
ב. בדיקה האם קיימת חלוקה פרופורציונלית לשחקנים עם זכויות שונות.

5.2.2 חיפוש במרחב המצבים

שיטה נוספת, המאפשרת לפתור בעיות קטנות בזמן סביר, היא **חיפוש במרחב המצבים (state-space search)**. בשיטה זו, האלגוריתם עובר על מרחב החלוקות האפשריות, אבל מסנן חלקים ממרחב זה, שבוודאות לא יובילו לפתרון מיטבי.

האלגוריתם מחזיק קבוצה של **מצבים (states)**. כל מצב הוא וקטור בגודל n+1, המייצג הקצאה חלקית – הקצאה של חלק מהחפצים. לכל i בין 1 ל־n, איבר i בווקטור מציין את הערך של הסל הנוכחי של שחקן i, בעיני שחקן i. האיבר האחרון בווקטור מציין את מספר החפצים שחולקו: אם כתוב שם t, המשמעות היא שהווקטור מייצג חלוקה של חפצים 1 עד t.

המצב ההתחלתי הוא וקטור שכולו אפסים, המייצג את החלוקה הריקה: ערך הסל של כל שחקן הוא אפס, וחולקו אפס חפצים. בהינתן מצב כלשהו המייצג חלוקה של חפצים 1,...,t, ושחקן כלשהו i, אפשר ליצור מצב חדש המשקף את נתינת החפץ ה־t+1 לשחקן i, על־ידי הוספת הערך $v_i(t+1)$ לאיבר i בוקטור, והוספת 1 לאיבר n+1 בוקטור.

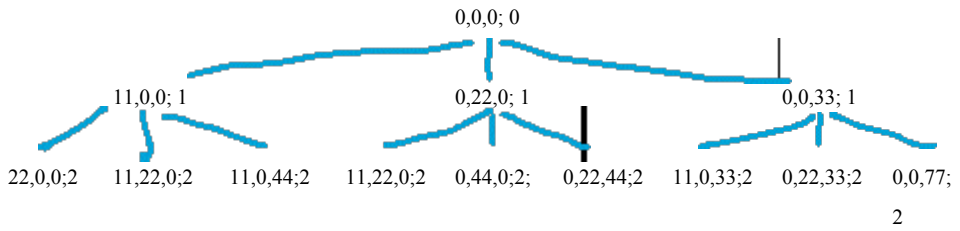
ניתן לסדר את כל וקטורי־המצב האפשריים במבנה של **עץ**. **שורש** העץ הוא המצב ההתחלתי (וקטור האפסים); לשורש ישנם n ילדים המייצגים את כל n האפשרויות לתת את חפץ מספר 1; ובאופן כללי, לכל קודקוד פנימי במרחק t מהעץ ישנם n ילדים המייצגים את כל n האפשרויות לתת את חפץ מספר t+1.

דוגמה 5.5

נניח שיש שלושה שחקנים. החפץ הראשון שווה 11 לשחקן 1, 22 לשחקן 2, ו־33 לשחקן 3. אז המצב הראשון הוא [0,0,0; 0], ואפשר ליצור ממנו שלושה מצבים חדשים המייצגים את שלוש האפשרויות לנתינת החפץ הראשון:

[11,0,0; 1]; [0,22,0; 1]; [0,0,33; 1].

נניח שהחפץ השני שווה 11 לשחקן 1, 22 לשחקן 2, ו־44 לשחקן 3. אז שתי הרמות הראשונות של עץ החלוקות ייראו כך:



כל **עלֶה** בעץ מייצג חלוקה שלמה – חלוקה של כל m החפצים. איך יודעים אם עלה מסויים מייצג **חלוקה פרופורציונלית**? – בודקים אם לכל שחקן i, הערך ה־i בווקטור

הוא לפחות הערך המגיע לשחקן i בחלוקה פרופורציונלית, שהוא n/1 מהערך שהשחקן מייחס לכל החפצים.

במחשבה ראשונה נראה, שכדי למצוא חלוקה פרופורציונלית, צריך לבנות את כל עץ־החלוקות.
זהו עץ מאד גדול: מספר העלים בעץ הוא n^m. אולם אפשר להקטין את מרחב־החיפוש על־ידי תהליך שנקרא **גיזום** (באנגלית pruning) – הורדת מצבים וענפים מיותרים מעץ־החלוקות. הנה שני כללים פשוטים שאפשר להשתמש בהם כדי לגזום את עץ החלוקות.

כלל א: מצבים זהים. אם מתגלים שני וקטורי־מצב זהים ברמה כלשהי בעץ, אפשר לגזום אחד מהם. בדוגמה 5.5, ברמה השניה בעץ ישנם שני וקטורי־מצב זהים – השני משמאל והרביעי משמאל, ולכן האלגוריתם יגזום אחד מהם ולא ימשיך לבנות את תת־העץ שמתחתיו.

כלל ב: חסמים אופטימיים. לכל קודקוד פנימי בעץ, נחשב "חלוקה אופטימית" – חלוקה טובה ביותר שאפשר להגיע אליה מקודקוד זה. החלוקה לא חייבת להיות מציאותית – היא יכולה להיות אופטימית יותר מדי. לדוגמה, אפשר לחשב את החלוקה שתתקבל אם נשכפל את כל החפצים שנשארו, וניתן אותם לכל אחד ואחד מהשחקנים. ניתן לחשב את החלוקה הזאת בקלות, ע"י הוספת ערכי כל החפצים שנשארו לכל אחד מהאיברים בוקטור. אם החלוקה האופטימית הזאת אינה פרופורציונלית, אז אפשר לגזום את כל הענף, כי אף חלוקה שתתקבל בענף זה לא תהיה טובה יותר מהחלוקה האופטימית. בהמשך לדוגמה 5.5, נניח שאחרי הרמה השניה נשארו עוד שמונה חפצים, שכולם שווים 1 עבור שחקן 1. מכאן שסכום הערכים של שחקן 1 הוא 8+11+11=30, וכדי שהחלוקה תהיה פרופורציונלית, הוא צריך לקבל סל שערכו לפחות 10. לכן ניתן לגזום את הענף החמישי משמאל (0, 44, 0; 2). בענף זה, גם בחלוקה האופטימית, שחקן 1 מקבל רק 8, והחלוקה אינה פרופורציונלית. באותו אופן ניתן לגזום את הענף השישי, השמיני והתשיעי משמאל. על־פי ערכי החפצים הנותרים עבור שחקנים 2 ו־3, ייתכן שאפשר לגזום ענפים נוספים.[א]

במקרים מסויימים, ניתן להשתמש בכללי גיזום נוספים.

בחנו את עצמכם 6.5

הציעו כלל גיזום המתאים לבעיית חלוקה שבה ההערכות של כל השחקנים **זהות**.
רמז: ישנו כלל גיזום המאפשר להקטין את מרחב החיפוש פי $n!$ (n עצרת) לפחות.

באיזה סדר נבנה את הקודקודים בעץ החלוקות? האפשרות הפשוטה ביותר היא לבנות **לרוחב**, כלומר, לבנות קודם את כל הקודקודים ברמה 1, ואז את כל הקודקודים ברמה 2, ואז את כל הקודקודים ברמה 3, וכו'. בשיטה זו, אנחנו שומרים בכל שלב t רק את קבוצת המצבים של רמה t. נסמן קבוצה זו ב S[t]. הנה האלגוריתם המתקבל:

א בספרות, במקום המושג "חסם אופטימי", משתמשים במושג **חסם תחתון (lower bound)** עבור בעיית מינימום, ו**חסם עליון (upper bound)** עבור בעיית מקסימום. אנחנו מעדיפים את המושג **חסם אופטימי**, המתאים לשני סוגי הבעיות.

אלגוריתם 7.5: בדיקה האם קיימת חלוקה פרופורציונלית – חיפוש לרוחב

א. אתחול: S[0] := קבוצה עם מצב אחד, שהוא וקטור עם n+1 אפסים.

ב. לכל t בין 1 ל־m, חשב את קבוצת המצבים S[t] באופן הבא:
* לכל מצב בקבוצה S[t-1], חשב את n המצבים הנובעים מנתינת חפץ t לכל אחד מ־n השחקנים.
* מחק מצבים בהתאם לכללי הגיזום.
* הוסף לקבוצה S[t] את כל המצבים החדשים שלא נגזמו.

ג. אם קבוצת המצבים S[m] מכילה מצב המתאים לחלוקה פרופורציונלית, ענה "כן";
אחרת, ענה "לא".

אפשרות שניה היא לבנות **לעומק**, כלומר, לבנות קודם את כל המסלול מהשורש ועד לעלה השמאלי ביותר, ואז לחזור אחורה ולבנות את העלה שמימינו, וכו'. ניתן לממש שיטה זו בעזרת מחסנית, כפי שמוסבר בספר על תורת הגרפים [להשלים הפניה]. היתרון של בניה לעומק הוא, שאנחנו מקבלים חלוקות שלמות באופן מיידי. אם התמזל מזלנו, ואחת החלוקות הראשונות שמצאנו היא חלוקה פרופורציונלית, אנחנו יכולים להפסיק לחפש מייד, ואיננו צריכים לבנות את שאר העץ. יתרון נוסף של בנייה לעומק הוא חיסכון בצריכת הזיכרון: בבנייה לעומק, אין צורך לשמור את כל המצבים ברמה הנוכחית, אלא מספיק לשמור את המצבים שבמסלול מהשורש עד המיקום הנוכחי. ב

כדי שהאלגוריתם יוכל להחזיר את החלוקה הפרופורציונלית במקרה שהיא קיימת, הוא צריך לשמור מידע שיאפשר לו לחזור מהסוף להתחלה: בכל פעם שנוצר מצב חדש בשלב t, יש לשמור את המצב הקודם לו (שממנו הוא נוצר) בשלב t-1. בסוף האלגוריתם, אם קיים מצב המתאים לחלוקה פרופורציונלית, אפשר לחזור אחורה למצב הקודם ולזהות למי ניתן החפץ ה־m; מכאן אפשר לחזור אחורה למצב שלפניו, ולזהות למי ניתן החפץ ה־m–1; וכו'.

כעת נחשב את **סיבוכיות זמן הריצה** של האלגוריתם. זמן הריצה של סריקת עץ הוא ליניארי במספר הקודקודים בעץ. ניתן לחשב שני חסמים עליונים על מספר הקודקודים בעץ החלוקות:

- מספר הקודקודים בכל רמה t הוא לכל היותר n^t. מספר הקודקודים הכולל הוא סכום של n^t כאשר t בין 0 ל־n. זה סכום של טור הנדסי, שהוא $O(n^m)$.

ב **הערה לגבי מינוחים**. בספרות ישנם שני מושגים שמשתמשים בהם כדי לתאר אלגוריתמים מסויימים של חיפוש במרחב המצבים.

* אלגוריתם **סיעוף וחסימה (branch and bound)** הוא אלגוריתם המבצע חיפוש **לעומק**, וגוזם לפי **חסם אופטימי** (כלל ב) בלבד.
* אלגוריתם **תיכנות דינאמי (dynamic programming)** הוא אלגוריתם המבצע חיפוש **לרוחב**, גוזם רק **מצבים זהים** (כלל א), ומשתמש במערך בינארי כדי לזכור את המצבים שנראו עד כה. [18, 15]

בספר זה בחרנו לתאר אלגוריתם כללי – חיפוש במרחב המצבים – שבו גם כללי־הגיזום וגם סדר־החיפוש ניתנים לבחירה. הדבר מאפשר להשתמש בשני סוגי כללי־הגיזום בו־זמנית עם כל אחד מהסדרים.

- נסמן באות V את המספר הכולל של ערכים אפשריים עבור סל כלשהו לשחקן כלשהו. מספר המצבים בכל רמה t, לאחר גיזום מצבים זהים (לפי כלל א), הוא לכל היותר V^n. יש m רמות, ולכן מספר המצבים הכולל הוא לכל היותר $m \cdot V^n$.

החסם העליון הראשון הוא מעריכי במספר החפצים. החסם העליון השני ליניארי במספר החפצים ומעריכי במספר השחקנים. בבעיות חלוקה טיפוסיות, מספר השחקנים קטן יחסית, אבל מספר החפצים עשוי להיות גדול. לדוגמה, בבעיית חלוקת ירושה, עשויים להיות שלושה או ארבעה אחים, אבל כמה עשרות חפצים. במצב זה, החסם העליון הראשון אינו מועיל. החסם העליון השני עשוי להיות מועיל - זה תלוי בגודל של V – מספר הערכים האפשריים של כל שחקן. לשם המחשה, נתבונן באתר ספליידיטג – אתר אינטרנט לחלוקה הוגנת. אחד היישומים באתר הוא חלוקת חפצים בדידים. האתר מבקש מכל שחקן להגיד איזה ערך הוא מייחס לכל חפץ. הערכים חייבים להיות מספרים שלמים, וסכום הערכים חייב להיות 1000. במקרה זה, V=1000, וכבר כשיש שלושה שחקנים, מספר המצבים יכול להגיע לעשרות מיליארדים. כדי לייעל את האלגוריתם, אפשר לתת לכל שחקן רק 100 נקודות לחלוקה. לחלופין, אפשר לתת לכל שחקן לדרג את כל החפצים ב"כוכבים" – מכוכב אחד (הכי פחות טוב) לחמישה כוכבים (הכי טוב). במקרה זה, מספר הערכים האפשריים לכל שחקן הוא 5m, ומספר המצבים הוא $O(m^{n+1})$ – פולינומיאלי במספר החפצים, והרבה יותר טוב מ־ $O(n^m)$.

שימו לב: בחישוב סיבוכיות זמן הריצה לא התייחסנו לגיזום לפי חסם אופטימי (כלל ב). כלל זה אינו משפיע על סיבוכיות זמן הריצה, כי במקרה הגרוע ביותר ייתכן שהחסם האופטימי נותן תמיד חלוקה פרופורציונלית. עם זאת, בניסויים עם קלטים אקראיים, מתברר שדווקא כלל־הגיזום הזה מועיל ומשפר את זמן הריצה הרבה יותר מאשר גיזום מצבים זהים. [18] מעבר לכך: גיזום מצבים זהים דורש לשמור בזיכרון את כל המצבים שראינו. כשמספר המצבים גדול, הזיכרון הנדרש לשם כך עלול להיות גדול מהזיכרון שברשותנו. במקרים אלה, גיזום מצבים זהים אינו מעשי, אבל גיזום לפי חסם אופטימי עדיין אפשרי ומועיל.

בחנו את עצמכם 8.5

כתבו אלגוריתמי חיפוש במרחב המצבים לפתרון הבעיות הבאות:

- א. בדיקה האם קיימת חלוקה ללא־קנאה.
- ב. בדיקה האם קיימת חלוקה פרופורציונלית לשחקנים עם זכויות שונות.

תארו את וקטורי־המצב ואת כללי־הגיזום. חשבו את סיבוכיות זמן־הריצה של האלגוריתמים.

5.2.3 אלגוריתמים נוספים **

האלגוריתמים שתיארנו בסעיף זה מניחים שלשחקנים ישנן **העדפות קרדינליות** – כל שחקן יודע להגיד כמה בדיוק שווה עבורו כל חפץ. לפעמים לשחקנים ישנן רק **העדפות אורדינליות** – כל שחקן יודע לדרג את החפצים מהטוב ביותר להכי פחות טוב, אבל

ג האתר http://www.spliddit.org/apps/goods

** ** תת-סעיף זה מומלץ למתקדמים בלבד.

לא יודע לתת ערך מספרי מדוייק לכל חפץ (ראו הגדרות בפרק 4). התוצאות המרכזיות עבור שחקנים כאלה הן:

- קיים אלגוריתם הבודק בזמן פולינומיאלי האם קיימת חלוקה **ודאי־פרופורציונלית**. האלגוריתם עובד גם עם שחקנים בעלי זכויות שונות.[2]
- בעיית ההחלטה, האם קיימת חלוקה **ודאי־ללא־קנאה**, היא NP־שלמה כשיש שלושה שחקנים או יותר.[3]
- קיים אלגוריתם הבודק האם קיימת חלוקה **אולי־פרופורציונלית**. האלגוריתם רץ בזמן פולינומיאלי כשלכל השחקנים יש דירוגים חזקים, או כשמספר השחקנים קבוע. סיבוכיות זמן הריצה כשיש שחקנים עם דירוגים חלשים, וכשמספר השחקנים לא קבוע, היא שאלה פתוחה.
- בעיית ההחלטה, האם קיימת חלוקה אולי־ללא־קנאה, היא NP־שלמה בדרך־כלל, אבל יש לה אלגוריתם עם זמן־ריצה פולינומי כשלכל השחקנים יש דירוגים חזקים, או כשיש רק שני שחקנים.

5.3 מיקסום סכום הערכים

כזכור מפרק 3, אחד מעקרונות היעילות הוא **העקרון האוטיליטרי**, הקובע שיש למקסם את סכום הערכים של כל השחקנים. מציאת חלוקה אוטיליטרית (הממקסמת את סכום הערכים) היא משימה קלה יחסית, אבל אינה מבטיחה הוגנות.

בחנו את עצמכם 9.5

נתונה בעיית חלוקת חפצים בדידים, שבה לכל השחקנים יש הערכות חיבוריות.

- א. כתבו אלגוריתם למציאת חלוקה אוטיליטרית. הוכיחו את נכונות האלגוריתם.
- ב. הוכיחו, שכל חלוקה אוטיליטרית של חפצים בדידים היא יעילה־פארטו.
- ג. הראו דוגמה שבה כל חלוקה אוטיליטרית **אינה** פרופורציונלית.

רמז: התשובות נמצאות בפרק 3.

כדי להבטיח הוגנות, אפשר לחפש חלוקה הממקסמת את סכום הערכים **מבין כל החלוקות ההוגנות**.

בחנו את עצמכם 10.5

כתבו אלגוריתמי תיכנות ליניארי, ואלגוריתמי חיפוש במרחב המצבים, לפתרון הבעיות הבאות:

- א. חלוקה הממקסמת את סכום הערכים, מבין כל החלוקות הפרופורציונליות (אם אין חלוקה פרופורציונלית, האלגוריתם יאמר "אין").
- ב. חלוקה הממקסמת את סכום הערכים, מבין כל החלוקות ללא-קנאה (אם אין חלוקה ללא-קנאה, האלגוריתם יאמר "אין").

חשבו את סיבוכיות זמן־הריצה של האלגוריתמים שכתבתם.

הבעיה היא, שאם אין בכלל חלוקה הוגנת, האלגוריתמים הללו אינם מועילים לנו.

פתרון שלישי, המאפשר לשלב את העקרון האוטיליטרי עם הוגנות, הוא למקסם את הערכים **מבין כל החלוקות ההוגנות־בקירוב**. כזכור, בפרק 4 הוכחנו שתמיד קיימת חלוקה ללא־קנאה עד־כדי חפץ אחד (EF1). לכן הגיוני לשאול, מבין כל החלוקות ה־EF1, מהי החלוקה שבה סכום־הערכים גדול ביותר? אנחנו נתאר כאן אלגוריתמים המתייחסים לתנאי־הוגנות פשוט יותר – פרופורציונליות עד־כדי חפץ אחד (PROP1). נשאל: **מבין כל החלוקות ה־PROP1, מהי החלוקה שבה סכום־הערכים גדול ביותר?** נתאר תחילה אלגוריתם חיפוש במרחב המצבים ואחר־כך אלגוריתם תיכנות ליניארי בשלמים.

5.3.1 חיפוש במרחב המצבים

האתגר המרכזי כאן הוא להגדיר את וקטור המצב. כדי שנוכל לבדוק האם חלוקה מסויימת היא PROP1, אין זה מספיק לשמור את הערך של כל שחקן, אלא יש לשמור גם

את החפץ בעל הערך הגדול ביותר **שאינו** משוייך לשחקן זה. לפיכך, וקטור המצב שלנו יהיה באורך 2n+1. לכל שחקן i יהיו שני איברים בווקטור־המצב:

- איבר אחד, שנסמן ב VAL_i, מייצג את הערך הנוכחי של הסל של i בעיני i;
- איבר שני, שנסמן ב $BEST_i$, מייצג את הערך הגדול ביותר, בעיני i, של חפץ המשוייך לשחקן אחר כלשהו.

בנוסף, כמו בסעיף 5.2.2, יהיה עוד אלמנט אחד, המייצג את מספר החפצים שחולקו עד כה.

המצב ההתחלתי הוא וקטור שכולו אפסים, המייצג את החלוקה הריקה. בהינתן מצב כלשהו המייצג חלוקה של חפצים 1,...,t, ושחקן כלשהו i, ניצור מצב חדש המשקף את נתינת החפץ ה־t+1 לשחקן i, באופן הבא:

- נוסיף את הערך $v_i(t+1)$ ל־ VAL_i;
- לכל שחקן $j \neq i$, אם $BEST_j < v_j(t+1)$, אז נעדכן את $BEST_j$ ל־ $v_j(t+1)$.

בכל פעם שנגיע לעלה, נבדוק אם החלוקה מקיימת את תנאי PROP1 לכל שחקן i, ע"י חישוב VAL_i+BEST_i והשוואת התוצאה לערך הפרופורציונלי של שחקן i – סכום הערכים שלו חלקי n. סיבוכיות זמן־הריצה של האלגוריתם היא כמספר המצבים. נחשב את מספר המצבים כפונקציה של מספר החפצים m, והמספר הכולל של ערכים אפשריים עבור סל כלשהו לשחקן כלשהו, שנסמן באות V:

- לכל שחקן i, השדה VAL_i יכול לקבל עד V ערכים.
- לכל שחקן i, השדה $BEST_i$ יכול לקבל עד m ערכים.
- השדה המייצג את מספר החפצים שחולקו, יכול לקבל m ערכים.

לכן מספר המצבים הכולל הוא $m \cdot m^n \cdot V^n$.

בחנו את עצמכם 11.5

א. כתבו אלגוריתם חיפוש במרחב המצבים למציאת חלוקה הממקסמת את סכום הערכים, מבין כל החלוקות ה־EF1.

ב. חשבו את סיבוכיות זמן־הריצה של האלגוריתם, כפונקציה של V, m, n.

ג. הוכיחו שהבעיה היא NP־קשה.

רמז: התשובות נמצאות במאמר.[4]

5.3.2 תיכנות ליניארי בשלמים

המבנה הכללי של התוכנית הליניארית שלנו יהיה דומה לזה שתואר בסעיף 5.2.1, בתוספת מקסימיזציה על סכום ערכי השחקנים:

$$\text{maximize} \quad \sum_i v_i(x_i)$$

subject to:

$$\sum_i x_{i,g} = 1 \qquad \text{for every item } g$$

$$0 \le x_{i,g} \le 1 \quad \text{and} \quad x_{i,g} \in Z \qquad \text{for all } i, g$$

x_i is PROP1 for all i

הסימון $v_i(x_i)$ הוא קיצור לסכום: $sum_g\ v_{i,g}*x_{i,g}$, המייצג את הערך של שחקן i בחלוקה x.
האתגר העיקרי כאן הוא להגדיר במדוייק את האילוץ האחרון – האילוץ הקובע שהחלוקה מקיימת את תנאי PROP1. איך נוכל להגדיר אילוץ לינארי המייצג את "החפץ עם הערך הגדול ביותר שאינו בסל של שחקן i"? הנה פתרון אפשרי.

נגדיר, עבור כל שחקן i וחפץ g, משתנה נוסף בשם $y_{i,g}$. נוסיף את האילוצים הבאים:

$$\sum_i y_{i,g} = 1 \qquad \text{for every agent } i$$

אראל: הסכום בשורה למעלה צריך להיות על g ולא על i.

$$0 \le y_{i,g} \le 1 \quad \text{and} \quad y_{i,g} \in Z \qquad \text{for all } i, g$$

$$y_{i,g} \le 1 - x_{i,g} \qquad \text{for all } i, g$$

שני האילוצים הראשונים אומרים, שעבור כל שחקן i, בדיוק משתנה אחד $y_{i,g}$ יהיה שווה 1, וכל שאר המשתנים יהיו 0. האילוץ השלישי אומר, שאותו משתנה יחיד $y_{i,g}$ השווה 1, חייב להתאים לחפץ g כלשהו שעבורו $x_{i,g} = 0$, כלומר, חפץ כלשהו **שאינו** משוייך לשחקן i. לסיכום, אוסף המשתנים $y_{i,g}$ מנחים את הפותר לבחור, עבור כל שחקן i, חפץ יחיד כלשהו שאינו בסל של i. עכשיו נשתמש במשתנים אלה כדי להגדיר את התכונה PROP1:

$$v_i(x_i) + \sum_g v_{i,g} y_{i,g} \ge \frac{v_i(all)}{n} \qquad \text{for every player } i$$

הסכום $\sum_g v_{i,g} y_{i,g}$ כולל איבר אחד בלבד השונה מאפס, והאיבר הזה שווה לערך של חפץ כלשהו, שאינו נמצא בסל של שחקן i. אם כך, משמעות האילוץ היא: "אם ניקח את הערך ששחקן i מייחס לסל שקיבל, ונוסיף לו ערך של חפץ כלשהו שאינו נמצא בסל שלו, התוצאה תהיה לפחות $\frac{v_i(all)}{n}$". זו בדיוק ההגדרה של חלוקה PROP1!

בחנו את עצמכם 12.5

כתבו תוכנית ליניארית בשלמים למציאת חלוקה הממקסמת את סכום הערכים, מבין כל החלוקות ה־EF1.

5.4 מיקסום מכפלת הערכים

בפרק 3 למדנו על **חלוקה אופטימלית־נאש** – חלוקה הממקסמת את מכפלת ערכי השחקנים, או את סכום הלוגריתמים של ערכי השחקנים (שתי ההגדרות שקולות). הוכחנו, שכאשר מחלקים משאבים רציפים לשחקנים עם הערכות חיבוריות, כל חלוקה אופטימלית־נאש היא גם ללא־קנאה וגם יעילה־פארטו. לכן אפשר לצפות, שלחלוקה אופטימלית־נאש יהיו תכונות טובות גם עבור חפצים בדידים. אבל יש כמה בעיות.

הבעיה הראשונה היא, שכאשר מחלקים חפצים בדידים, לפעמים לא קיימת חלוקה שבה מכפלת־הערכים היא חיובית.[ד] לדוגמה, כשמספר החפצים קטן ממספר השחקנים, מכפלת הערכים **בכל** חלוקה היא אפס, ולכן כל חלוקה ממקסמת את מכפלת הערכים. בפרט, חלוקה הממקסמת את מכפלת־הערכים עשויה להיות לא יעילה־פארטו ולא הוגנת. לכן נשפץ את ההגדרה באופן הבא.

הגדרה 13.5

חלוקה אופטימלית־נאש היא חלוקה הממקסמת את מספר השחקנים המקבלים ערך חיובי, ובכפוף לזה, ממקסמת את מכפלת הערכים של השחקנים המקבלים ערך חיובי.

5.4.1 התכונות של חלוקה אופטימלית־נאש

משפט 14.5

כל חלוקה אופטימלית־נאש על־פי הגדרה 5.13 היא יעילה־פארטו.

הוכחה: נתונה חלוקה א שהיא אופטימלית-נאש על פי הגדרה 5.9.

נניח בשלילה שהחלוקה אינה יעילה פארטו. אזי קיימת חלוקה ב שהיא שיפור פארטו שלה. בחלוקה ב לכל השחקנים יש ערך לפחות כמו בחלוקה א ולחלק מהשחקנים יש ערך גבוה יותר.

נחלק לשני מקרים: מקרה ראשון, קיים שחקן בחלוקה ב עם ערך חיובי אשר אין לו ערך חיובי בחלוקה א. כלומר בחלוקה ב מספר השחקנים עם ערך חיובי גדול יותר מחלוקה א – בסתירה לכך שחלוקה א היא אופטימלית-נאש וממקסמת את מספר השחקנים שמקבלים ערך חיובי.

מקרה שני: מספר השחקנים עם ערך חיובי בחלוקה ב זהה למספר השחקנים עם ערך חיובי בחלוקה א. בחלוקה ב לכל השחקנים יש ערך חיובי לפחות כמו בחלוקה א ולחלק יש ערך חיובי גבוה יותר ולכן חלוקה ב בעלת מכפלת ערכים גבוהה יותר מחלוקה א – בסתירה לכך שחלוקה א היא אופטימלית-נאש וממקסמת את מכפלת הערכים של השחקנים המקבלים ערך חיובי.

מש"ל.

ד בחלוקת משאבים רציפים לשחקנים עם הערכות חיוביות, תמיד קיימות חלוקות עם מכפלת־ערכים חיובית, למשל, החלוקה השווה.

האם חלוקה אופטימלית־נאש היא תמיד ללא־קנאה? ודאי שלא. הרי יש בעיות חלוקה שבהם חלוקה ללא־קנאה כלל אינה קיימת, בעוד שחלוקה אופטימלית־נאש תמיד קיימת. אולם כל חלוקה אופטימלית־נאש היא "כמעט" ללא קנאה – היא מקיימת את תנאי EF1 – ללא־קנאה־מלבד־חפץ־אחד – שהוגדר בפרק 4. המשפט הוכח ע"י שישה חוקרים: קרגיאניס, קורוקאווה, מולין, פרוקצ'יה, שאה, וואנג.[5]

משפט 15.5

כל חלוקה אופטימלית־נאש על־פי הגדרה 5.13 היא ללא־קנאה־מלבד־חפץ־אחד.

הוכחה: נתונה חלוקה אופטימלית־נאש כלשהי $X_1,\dots,X_n$.

שלב א. אם קיים חפץ כלשהו, שערכו עבור כל השחקנים הוא אפס – אז נסלק אותו מהמערכת ונתעלם ממנו; הדבר אינו משנה את הערכים ואת מכפלתם.

שלב ב. כעת, כל חפץ בהכרח נתון לשחקן המייחס לו ערך חיובי ממש. אחרת, היה אפשר להעביר את החפץ לשחקן אחר כלשהו, המייחס לו ערך חיובי. העברה זו היתה מגדילה את מספר השחקנים עם ערכים חיוביים, או את מכפלת הערכים של השחקנים עם ערכים חיוביים, בסתירה להגדרה של חלוקה אופטימלית־נאש.

שלב ג. נניח ששחקן כלשהו i מקנא בשחקן אחר כלשהו j. מכאן שבסל של j יש לפחות חפץ אחד, וערכו עבור שני השחקנים חיובי. נסתכל על כל החפצים בסל של j. לכל חפץ g, נחשב את יחס הערכים בין השחקנים (היחס מוגדר היטב, כי לפי שלב ב המכנה חיובי):

$$r(g) := \frac{v_i(g)}{v_j(g)}$$

נבחר חפץ g שיחס־הערכים שלו גדול ביותר, ונעביר אותו מ־j ל־i. כיוון שהחלוקה X אופטימלית־נאש, מכפלת הערכים בחלוקה החדשה שווה או קטנה מהמכפלה ב־X, ולכן:

$$\left[v_i(X_i)+v_i(g)\right]\cdot\left[v_j(X_j)+v_j(g)\right]\le v_i(X_i)\cdot v_j(X_j)$$

$$\Rightarrow\ v_j(X_j)\cdot\frac{v_i(g)}{v_j(g)}\le v_i(X_i)+v_i(g)$$

$$\Rightarrow\ v_j(X_j)\cdot r(g)\le v_i(X_i)+v_i(g)\quad(*)$$

החפץ g נבחר כך שיחס־הערכים שלו הוא הגדול ביותר מבין כל החפצים בסל X_j. לכן לכל חפץ h בסל X_j, מתקיים:

$$r(h)\le r(g)$$

נעבור על כל החפצים h בסל X_j, נכפיל את אי־השיוויון ב־ $v_j(h)$, ונסכום:

$$\sum_h r(h)\, v_j(h) \le \sum_h r(g)\, v_j(h)$$

$$\Rightarrow \sum_h v_i(h) \le r(g) \cdot \sum_h v_j(h)$$

$$\Rightarrow v_i(X_j) \le r(g) \cdot v_j(X_j)$$

כי ההערכות חיבוריות. נציב אי־שיוויון זה באי־השיוויון המסומן ב (*) למעלה, ונקבל:

$$\Rightarrow v_i(X_j) \le v_i(X_i) + v_i(g)$$

$$\Rightarrow v_i(X_j) - v_i(X_i) \le v_i(g),$$

וזו בדיוק ההגדרה של EF1.

מש"ל.

5.4.2 חישוב חלוקה אופטימלית־נאש

משפט 16.5

הבעיה של חישוב חלוקת חפצים בדידים הממקסמת את מכפלת־הערכים היא NP־קשה.

הוכחה: ההוכחה היא ברדוקציה מבעיית חלוקת־המספרים (הגדרה 5.1). נתונה בעייה של חלוקת־מספרים עם m מספרים חיוביים שסכומם 2V. נגדיר בעיית מיקסום־מכפלת־ערכים עם שני שחקנים ו-m חפצים. לשני השחקנים יש הערכה חיבורית זהה: הערך של כל חפץ j בעיניהם שווה למספר ה-j בבעיית חלוקת־המספרים.

אם קיימת חלוקת־מספרים עם סכום שווה, אז קיימת חלוקת־חפצים עם מכפלה V^2. אם לא קיימת חלוקת־מספרים עם סכום שווה, אז בכל חלוקת מספרים, הסכומים הם $V - d$ ו- $V + d$ עבור מספר שלם d כלשהו, ולכן מכפלת הערכים המקסימלית היא $(V + d)(V - d) = V^2 - d^2$, שהיא קטנה יותר מ- V^2. מציאת חלוקה הממקסמת את מכפלת־הערכים מאפשרת, בפרט, לזהות אם המכפלה המקסימלית היא V^2 או קטנה יותר, ולכן מאפשרת לזהות אם חלוקת־מספרים שווה קיימת או לא.

מש"ל.

למרות הקושי החישובי, ניתן למצוא חלוקה אופטימלית־נאש בעזרת שני הכלים שראינו בסעיף 5.2: חיפוש במרחב המצבים, ותיכנות ליניארי בשלמים. לשם פשטות נתמקד במקרה הנפוץ יותר, שבו קיימת חלוקה עם מכפלת־ערכים חיובית. במקרה זה, מספיק למצוא חלוקה הממקסמת את מכפלת־הערכים.

אלגוריתם **חיפוש במרחב המצבים** למציאת חלוקה אופטימלית־נאש דומה בעיקרו לאלגוריתם 5.7: הוא בונה את הקודקודים בעץ־החלוקות, גוזם קודקודים מיותרים, ומחפש עלֶה שבו מכפלת הערכים גדולה ביותר. גיזום מצבים זהים (לפי כלל א) מתבצע

באופן זהה. אולם גיזום לפי חסם אופטימי (כלל ב) מתבצע באופן שונה במקצת. האלגוריתם מחזיק **חסם פסימי**, שהוא החלוקה השלמה עם מכפלת־הערכים הגדולה ביותר שנמצאה עד כה. זה "חסם פסימי" כי בתרחיש הרע ביותר לא נמצא חלוקה עדיפה, ונשתמש בחלוקה שכבר יש בידינו. עבור כל קודקוד חדש, האלגוריתם מחשב חלוקה אופטימית, למשל, חלוקה שבה משכפלים את כל החפצים שנשארו ונותנים עותק לכל שחקן. מכפלת הערכים בחלוקה זו היא **חסם אופטימי** עבור הקודקוד החדש. אם החסם האופטימי הזה אינו טוב יותר מהחסם הפסימי (כלומר: מכפלת הערכים בחלוקה האופטימית שווה או קטנה ממכפלת הערכים הטובה ביותר שמצאנו עד כה), אז אפשר לגזום את הקודקוד החדש. בכל פעם שהאלגוריתם מוצא עלֶה שבו מכפלת הערכים גדולה יותר מהחסם הפסימי, הוא מעדכן את החסם הפסימי בהתאם. בסוף האלגוריתם, לאחר שכל הקודקודים נסרקו או נגזמו, האלגוריתם מחזיר את החלוקה של החסם הפסימי. גיזום זה יעיל ביותר כאשר בניית העץ מתבצעת לעומק. זאת, כיוון שבבנייה לעומק אנחנו מקבלים באופן מיידי חלוקה שלמה שניתן להציב בחסם הפסימי, ומכאן והלאה ניתן לגזום כל קודקוד שהחסם האופטימי שלו אינו טוב יותר.

סיבוכיות זמן־הריצה של האלגוריתם זהה לזו של אלגוריתם 5.7: היא פולינומיאלית במספר החפצים ובערך הגדול ביותר של שחקן, ומעריכית במספר השחקנים.

בחנו את עצמכם 17.5

תארו אלגוריתם חיפוש במרחב המצבים למציאת חלוקה אופטימלית־נאש במקרה הכללי – כאשר ייתכן שלא קיימת חלוקה עם מכפלת ערכים חיובית.

אלגוריתם **תיכנות־ליניארי בשלמים** למציאת חלוקה אופטימלית־נאש דומה בעיקרו לאלגוריתם 5.3. כאמור בפרק 3, מיקסום המכפלה שקול למיקסום סכום הלוגריתמים:

Maximize $\log(v_1(x_1)) + \ldots + \log(v_n(x_n))$

such that x is a partition

ניתן לכתוב תוכנית זו בעזרת אותם משתנים $x_{i,g}$ וקבועים $v_{i,g}$ שהשתמשנו בהם באלגוריתם 5.3. הביטוי $v_i(X_i)$ מציין את הערך שמייחס שחקן i לסל שהוא מקבל:

$$v_i(x_i) := \sum_g v_{i,g} x_{i,g}$$

והאילוץ "x is a partition" הוא קיצור לשלושת האילוצים המגדירים את העובדה שהמשתנים $x_{i,g}$ מייצגים חלוקה תקנית של חפצים בדידים, כפי שהוסבר באלגוריתם 5.3:

$\sum_i x_{i,g} = 1$ for every item g

$0 \le x_{i,g} \le 1$ for all i, g

$x_{i,g} \in Z$ for all i, g

הבעיה היא, שפונקציית־המטרה אינה ליניארית. בעיית מיטוב בשלמים עם פונקציית־מטרה לא ליניארית היא קשה מאד לפתרון. אולם ששת החוקרים הנזכרים למעלה[3] לא התייאשו. הם מצאו שיטה המאפשרת להפוך את הבעיה ללא־ליניארית, **במקרה הפרטי שבו כל אחד מהערכים מוגבל למספר בין 1 ל-1000.** כיוון שממשק־המשתמש באתר ספלידיט[3] ממילא מאפשר להכניס רק ערכים בין 1 ל-1000, המקרה הפרטי הזה הוא משמעותי.

כדי להבין את הפתרון, נציג את בעיית המיטוב בצורה שונה. נגדיר לכל שחקן i משתנה L_i המייצג חסם תחתון על הלוגריתם של הערך שלו, ונפתור את הבעיה הבאה:

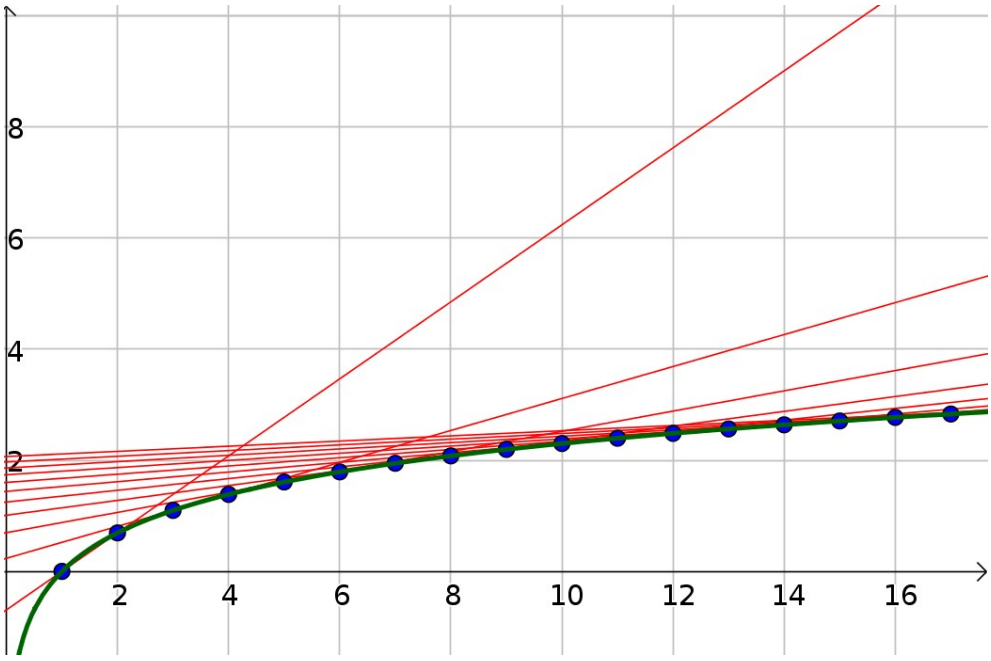
$$\begin{aligned} &\text{Maximize} && L_1 + \ldots + L_n \\ &\text{such that} && L_i \le \log(v_i(x_i)) \quad \text{for all } i \in \{1,\ldots,n\}; \\ &\text{and} && x \text{ is a partition.} \end{aligned}$$

הבעיה הזו שקולה לבעיה המקורית. אבל הבעיה עדיין קשה לפתרון, כי האילוצים אינם ליניאריים. כעת נחליף את האילוץ האמצעי, עבור כל שחקן i, ב-1000 אילוצים ליניאריים:

$$L_i \le \log(k) + [\log(k+1) - \log(k)] \cdot [v_i(x_i) - k]$$

$$\text{for all } k \in \{1,\ldots,1000\}; \quad \text{for all } i \in \{1,\ldots,n\}.$$

הביטוי בצד ימין מתאר קו ישר, החוסם את הפונקציה $\log(v_i(x_i))$ מלמעלה, ומשיק לפונקציה בנקודה k. באיור הבא ניתן לראות את הפונקציה $\log(v_i(x_i))$ עצמה בירוק, ואת הקוים החוסמים אותה מלמעלה באדום, עבור $k = 1,\ldots,16$:



כיוון שכל הערכים של $v_i(x_i)$ הם מספרים שלמים בין 1 ל־1000, מרחב הפתרונות של הבעיה החדשה זהה למרחב הפתרונות של הבעיה המקורית. בבעיה החדשה יש אמנם

הרבה יותר אילוצים (פי 1000), אבל היא בעיה ליניארית, ולכן ניתן לפתור אותה במהירות יחסית בעזרת כלים לפתרון בעיות־מיטוב ליניאריות.

יש כאן גישה מעניינת: אם מסתכלים על הבעיה כבעיה תיאורטית, ומנסים לפתור אותה לכל הקלטים האפשריים – היא קשה חישובית. אבל אם מסתכלים עליה כבעיה מעשית, ומנסים לפתור אותה רק עבור הקלטים שבני־אדם בפועל מזינים לאלגוריתם – הבעיה נעשית פתירה.

5.4.3 חלוקה אופטימלית־נאש עם זכויות שונות

ניתן להכליל את המושג של חלוקה אופטימלית־נאש לשחקנים שיש להם זכויות שונות, כפי שהסברנו בפרק 3. בעיית המיטוב המתאימה היא:

$$\begin{aligned} &\text{Maximize} && w_1 \cdot \log(v_1(x_1)) + \ldots + w_n \cdot \log(v_n(x_n)) \\ &\text{such that} && x \text{ is a partition.} \end{aligned}$$

כאשר המשקלים $w_1,\ldots,w_n$ מייצגים את זכויות השחקנים. אולם במצב זה, עקרון מקסום המכפלה מאבד את תכונותיו הטובות – הוא כבר לא גורר אף אחת מההכללות של EF1 שראינו בפרק 4.[6] לשם המחשה, נתייחס להכללה הנובעת משיטת ווסטר: רמת הקנאה המשוקללת בין כל שני שחקנים i, j צריכה להיות לכל היותר

$$v_i(z) \cdot \frac{\left(\frac{1}{w_i} + \frac{1}{w_j}\right)}{2}.$$

נניח שיש שני חפצים זהים בשווי 1, ושני שחקנים עם זכויות 1, 10. רמת הקנאה המשוקללת המותרת היא לכל היותר:

$$1 \cdot \frac{\left(\frac{1}{w_2} + \frac{1}{w_1}\right)}{2} = 1 \cdot \frac{\left(\frac{1}{10} + \frac{1}{1}\right)}{2} = \frac{1.1}{2} = 0.55$$

קיימת חלוקה אחת ויחידה שבה המכפלה חיובית (עם או בלי משקלים), והיא החלוקה שבה כל שחקן מקבל חפץ אחד. לכן זו החלוקה האופטימלית־נאש היחידה. בחלוקה זו, רמת הקנאה המשוקללת של השחקן עם זכות 10 בשחקן עם זכות 1 היא:

$$\frac{1}{w_2} - \frac{1}{w_1} = \frac{1}{1} - \frac{1}{10} = 0.9$$

שהיא מעל הרמה המותרת.

בחנו את עצמכם 18.5

הראו דוגמה שבה אף חלוקה אופטימלית־נאש אינה מקיימת את הכללת EF1 הנובעת משיטת אדאמס, ודוגמה (אחרת) שבה אף חלוקה אופטימלית־נאש אינה מקיימת את הכללת EF1 הנובעת משיטת ג'פרסון.

5.4.4 אלגוריתמים נוספים **

כאמור, מציאת חלוקה הממקסמת את מכפלת הערכים היא בעייה NP־קשה. אולם בשנים האחרונות פותחו כמה אלגוריתמי קירוב־גורם־קבוע (constant-factor approximation) – אלגוריתמים המוצאים חלוקה עם מכפלת־ערכים שהיא לפחות המכפלה הגדולה ביותר חלקי מספר קבוע כלשהו.[7] כמו כן, ישנם אלגוריתמים למיקסום מכפלת־הערכים במקרה הפרטי שבו ההערכות הן **בינאריות** – כל שחקן מעריך כל חפץ כ־0 או כ־1.[8]

אלגוריתם מיקסום־המכפלה מוצא חלוקה יעילה־פארטו והוגנת־בקירוב, אבל הוא לא רץ בזמן פולינומיאלי. ישנם אלגוריתמים למציאת חלוקה יעילה־פארטו והוגנת־בקירוב מיקסום מכפלת־הערכים, הרצים בזמן פולינומיאלי. אלגוריתמים אלה משתמשים בשיטות שנלמד בפרק 6 (סעיף 6.4), ולכן נתאר אותם שם.

5.4.5 שאלות נוספות לתירגול וחזרה

1. דוגמאות

תנו דוגמאות לחלוקה של ארבעה חפצים בדידי בין שני שחקנים, שבהן:

א. החלוקה האופטימלית־נאש היא ללא קנאה.

ב. החלוקה האופטימלית־נאש נותנת שני חפצים לכל שחקן.

ג. החלוקה האופטימלית־נאש נותנת חפץ אחד לשחקן א ושלושה חפצים לשחקן ב.

2. הכיוון ההפוך

הוכחנו שחלוקה אופטימלית־נאש היא גם יעילה־פארטו וגם כמעט ללא־קנאה. הראו שהכיוון ההפוך אינו נכון: הראו חלוקה שהיא יעילה־פארטו, וממש ללא־קנאה (לא רק "כמעט"), שאינה ממקסמת את מכפלת־הערכים.

3. תיכנות

ממשו את אלגוריתם מיקסום־המכפלה בשפת-תיכנות כלשהי לבחירתכם (השתמשו בספריה לפתרון בעיות מיטוב קמורות). כתבו פונקציה, המקבלת כקלט מטריצה של ערכי החפצים לשחקנים, ומחזירה את החלוקה. לדוגמה, אם הקלט הוא:

```
[[1,4,9],
 [9,5,1]]
```

אז הפלט הוא:

```
Agent #1 gets item #3.
Agent #2 gets items #1, #2.
```

*** 4. פרופורציונליות**

הוכיחו או הפריכו: אם קיימת חלוקה פרופורציונלית, אז כל חלוקה אופטימלית־נאש היא פרופורציונלית.

** ** תת-סעיף זה מומלץ למתקדמים בלבד.

5.5 מיקסום הערך הקטן ביותר

חלוקה אגליטרית היא חלוקה הממקסמת את הערך הנמוך ביותר של שחקן. בפרק 3 הוכחנו, שכאשר ההערכות **מנורמלות** (כל שחקן מייחס את אותו ערך לכל המשאבים), וקיימת חלוקה פרופורציונלית, אז כל חלוקה היא אגליטרית. לפי זה, החלוקה האגליטרית מבטאת שאיפה להתקרב ככל האפשר לעקרון הפרופורציונליות: אם קיימת חלוקה פרופורציונלית – חלוקה שבה כל שחקן מקבל $\frac{1}{n}$ מהשווי הכללי – אז נחזיר חלוקה פרופורציונלית; אחרת, נמצא ערך כלשהו x, שהוא קטן מ־$\frac{1}{n}$, ונחזיר חלוקה שבה כל שחקן מקבל לפחות x מהשווי הכללי; נבחר את ה־x הגדול ביותר האפשרי במסגרת אילוצי הבעיה.

בפרק 3, כשחילקנו משאבים רציפים, יכולנו לחשב חלוקה אגליטרית במהירות ע"י פתרון בעיית־מיטוב קמורה. כשהחפצים בדידים, הבעיה – כצפוי – קשה יותר.

בחנו את עצמכם 19.5

א. הוכיחו שבעיית חישוב חלוקה אגליטרית היא NP־קשה. **רמז**: העזרו בהוכחת משפט 5.16.

ב. תארו אלגוריתם לחישוב חלוקה אגליטרית בעזרת תיכנות ליניארי בשלמים. **רמז**: היעזרו בפרק 3, בסעיף על חישוב חלוקה אגליטרית, ושלבו אותו עם אלגוריתם 5.3. הסבירו מה בדיוק צריך לשנות באלגוריתם.

ג. תארו אלגוריתם חיפוש במרחב המצבים לחישוב חלוקה אגליטרית. רמז: העזרו באלגוריתם 5.7. הסבירו מה בדיוק צריך לשנות באלגוריתם.

לשני האלגוריתמים – תיכנות ליניארי בשלמים וחיפוש במרחב המצבים – זמן־ריצה שאינו פולינומיאלי בגודל הקלט, והם עלולים לקחת זמן רב כשהקלט גדול. כדי להתמודד עם בעיה זו, נלמד עכשיו כמה אלגוריתמי־קירוב מהירים.

5.5.1 חלוקה אגליטרית עם הערכות זהות – קירוב גורם־קבוע

אלגוריתם **קירוב גורם־קבוע** (constant-factor approximation) לבעיית מיטוב הוא אלגוריתם המחזיר, לכל קלט של הבעיה, פתרון שערכו לפחות r כפול הפתרון המיטבי, כאשר r הוא מספר ממשי קבוע (שאינו תלוי בקלט).

בסעיף זה נראה אלגוריתם קירוב גורם־קבוע עבור מקרה פרטי שבו לכל השחקנים יש אותן הערכות – הם מסכימים על ערכי החפצים. זה מצב נפוץ, למשל, כאשר הערכים הם מחירי־שוק, והמטרה היא למצוא חלוקה אגליטרית על־פי מחירי השוק.

אלגוריתמי-הקירוב הראשונים לבעיה זו פותחו עבור בעיה הנדסית, שבמבט ראשון אין כל קשר בינה לבין הוגנות: בעיית **תיזמון תהליכים במחשב מרובה־מעבדים** (באנגלית: multiprocessor scheduling; נקראת גם machine scheduling או job scheduling). יש לנו מחשב עם כמה מעבדים זהים; אנחנו רוצים לבצע בעזרתו משימה חישובית מסויימת, המורכבת ממספר רב של תהליכים היכולים לרוץ במקביל. אנחנו יודעים מראש כמה זמן דרוש למעבד לבצע כל תהליך. זמן־הסיום של המשימה כולה (באנגלית: **makespan**) הוא זמן־הסיום של התהליך האחרון. השאלה היא, איך לחלק את התהליכים בין המעבדים, כך שזמן־הסיום הכולל יהיה הקצר ביותר האפשרי?

דוגמה 20.5

יש ארבעה מעבדים ותשעה תהליכים. אורכי התהליכים בשניות הם: [ה]
4, 4, 4, 5, 5, 6, 6, 7, 7.

נבדוק שתי חלוקות אפשריות של תהליכים למעבדים:

- חלוקה א: מעבד ראשון מסיים ב-5+6, מעבד שני מסיים ב-5+6, מעבד שלישי מסיים ב-4+7, מעבד רביעי מסיים ב-4+4+7. כלומר שלושה מעבדים מסיימים תוך 11 שניות ,אבל המעבד הרביעי מסיים תוך 15 שניות, ולכן זמן־הסיום של המשימה כולה (makespan) הוא 15.
- חלוקה ב: מעבד ראשון מסיים ב-6+6, מעבד שני מסיים ב-5+7, מעבד שלישי מסיים ב-5+7, מעבד רביעי מסיים ב-4+4+4.
  כל המעבדים מסיימים תוך 12 שניות, ולכן זמן־הסיום הכולל הוא 12. זוהי החלוקה היעילה ביותר בדוגמה זו: כיוון שסכום אורכי התהליכים הוא 48, בכל חלוקה קיים מעבד כלשהו שזמן־הסיום שלו הוא לפחות 12.

הסיבה הראשונה לעיסוק בבעיה זו היתה, כאמור, למצוא חלוקת־עבודה יעילה ומהירה. אך למעשה, בעיה זו שקולה לחלוטין לבעייה של חלוקה אגליטרית של חפצים עם **ערך שלילי** בין שחקנים עם **הערכות זהות**. דוגמה לחפצים בעלי ערך שלילי היא **מטלות**, כגון תורנויות בבית או בבסיס צבאי. רוב האנשים אינם אוהבים לבצע מטלות, אבל הערך (השלילי) שהם מייחסים לכל מטלה עשוי להיות שונה.

דוגמה 21.5

יש ארבעה שחקנים הצריכים לבצע ביחד תשע מטלות. הם מייחסים למטלות ערכים שליליים (כולם מסכימים על הערכים של כל מטלה).

-4, -4, -4, -5, -5, -6, -6, -7, -7.

אנחנו רוצים למצוא חלוקה אגליטרית. נבדוק שתי חלוקות:

ה על־פי טל גרינשפון, "אלגוריתמי זימון ושיבוץ", קורס באוניברסיטת אריאל.

- חלוקה א: שחקן אחד מקבל שלוש מטלות עם ערכים -4, -4, -7; שחקן שני מקבל שתי מטלות עם ערכים -4, -7; ועוד שני שחקנים מקבלים כל אחד שתי מטלות עם ערכים -6, -5. ערכי השחקנים הם: -11, -11, -11, -15, והערך האגליטרי הוא -15.
- חלוקה ב: שחקן אחד מקבל שלוש מטלות עם ערכים -4, -4, -4; שחקן שני מקבל שתי מטלות עם ערכים -6, -6; ועוד שני שחקנים מקבלים כל אחד שתי מטלות עם ערכים -7, -5. ערכי השחקנים הם: -12, -12, -12, -12, והערך האגליטרי הוא -12.

הערך האגליטרי בחלוקה ב גדול יותר מבחלוקה א. למעשה, חלוקה ב היא החלוקה האגליטרית במקרה זה: כיוון שסכום ערכי המטלות הוא -48, בכל חלוקה יש לפחות שחקן אחד שערכו לכל היותר -12.

קל לראות, שהדבר נכון באופן כללי: כל חלוקת תהליכים למעבדים, הממזערת את זמן־הריצה המקסימלי, שקולה לחלוקת מטלות לאנשים, הממקסמת את הערך (השלילי) המינימלי – חלוקה אגליטרית. הבעיה היא NP־קשה – ההוכחה זהה להוכחה עבור חפצים עם ערכים חיוביים (בחנו את עצמכם 5.19).

כעת נראה אלגוריתם חמדני פשוט למציאת חלוקה אגליטרית־בקירוב. לצורך הפשטות, כדי שנוכל לעבוד עם מספרים חיוביים, נגדיר את **העלות** (cost) של כל מטלה כערך של המטלה **בערך מוחלט**, והמטרה היא למצוא חלוקה שבה העלות **המקסימלית** היא **קטנה ביותר**. בעולם תיזמון התהליכים, האלגוריתם הזה נקרא "תיזמון רשימה" (List Scheduling) כי הוא מתזמן את התהליכים לפי רשימה קבועה מראש. אנחנו משתמשים באלגוריתם לא לצורך תיזמון אלא לצורך חלוקת מטלות, ולכן נקרא לו "אלגוריתם הרשימה".

אלגוריתם 22.5: אלגוריתם הרשימה (חלוקת מטלות לשחקנים עם הערכות זהות)

1. עבור על כל המטלות לפי הסדר שבו הן נמצאות ברשימה.
2. תן את המטלה הבאה ברשימה לְשחקן, שסכום העלויות הנוכחי שלו קטן ביותר.
3. אם יש עוד מטלות, חזור לשורה 2.

בדוגמה 5.20, המטלה הראשונה בסדרה היא ה־4 והאחרונה היא ה־7. האלגוריתם נותן את ארבעת המטלות הראשונות לארבעת השחקנים בסדר כלשהו, למשל, א-4, ב-4, ג-4, ד-5. שלוש המטלות הבאות – 6,5,5 – ניתנות לשחקנים שהעלות הנוכחי שלהם קטנה ביותר, שהם א, ב, ג; והמטלה הבאה – 6 – ניתנת לשחקן ד. העלויות לאחר הסיבוב השני הן א-9, ב-9, ג-10, ד-11. שתי המטלות האחרונות ניתנות לשחקנים א, ב, והעלויות הסופיות הן א-16, ב-16, ג-10, ד-11. הערך האגליטרי הוא -16. אנחנו רואים שהאלגוריתם אינו מוצא את החלוקה האגליטרית (עם עלויות 12, 12, 12, 12), אבל הוא גם לא כל כך רחוק – הערך האגליטרי שלו גדול רק פי 4/3=16/12 מהערך המיטבי.

כשמנתחים אלגוריתמי קירוב, התכונה המעניינת היא **יחס הקירוב** שלהם (approximation ratio) – היחס בין ערך הפתרון שהאלגוריתם מוצא, לבין ערך הפתרון הטוב ביותר.

בבעיות מינימיזציה, כמו הבעיה שלנו, יחס הקירוב גדול מ־1, כי הפתרון של האלגוריתם גדול יותר מהפתרון האופטימלי; בבעיות מקסימיזציה, יחס הקירוב קטן מ־1, כי הפתרון של האלגוריתם קטן יותר מהפתרון האופטימלי. בדוגמה 5.20, יחס הקירוב של אלגוריתם הרשימה היה 4/3.

משפט 23.5

יחס־הקירוב של אלגוריתם הרשימה (אלגוריתם 5.22) קטן מ־2.

הוכחה: בהינתן קלט מסויים לבעיה, נסמן ב־OPT את העלות האגליטרית (העלות הגדולה ביותר בחלוקה האגליטרית). לצורך ההוכחה, ננרמל את עלויות כל המטלות ע"י חלוקה ב־OPT. לאחר הנירמול, סכום העלויות של המטלות שמקבל שחקן כלשהו בחלוקה האגליטרית קטן או שווה 1. לכן, העלות של כל מטלה ומטלה קטנה או שווה 1. כמו כן, סכום העלויות של כל המטלות קטן או שווה n (מספר השחקנים).

בכל שלב־ביניים באלגוריתם, סכום העלויות של כל המטלות שכבר חולקו קטן ממש מ־n. לכן, לפי כלל שובך־היונים, סכום העלויות הקטן ביותר של שחקן כלשהו קטן מ־1. לכן, האלגוריתם נותן את המטלה הבאה לשחקן שסכום העלויות שלו קטן מ־1. סכום העלויות החדש של שחקן זה קטן מ־2. מכאן, שלכל שחקן המקבל מטלה, כולל השחקן האחרון, סכום העלויות קטן מ־2.

מש"ל.

ע"י שינוי קל בהוכחה, ניתן להוכיח משפט מעט חזק יותר.

בחנו את עצמכם 24.5

הוכיחו: יחס הקירוב של אלגוריתם הרשימה בחלוקה ל־ n שחקנים הוא לכל היותר $2-\frac{1}{n}$.

אלגוריתם הרשימה יכול לטפל במטלות בכל סדר שבו הן מגיעות, ולכן הוא שימושי גם כאשר המטלות אינן ידועות מראש. חישבו, למשל, על עובדים המקבלים בכל יום מטלות חדשות, וצריכים לחלק את המטלות ביניהם מייד, כך שהחלוקה הכוללת (בסיכום חודשי או שנתי) תהיה הוגנת.

אם כל המטלות ידועות מראש, ניתן להשתמש באלגוריתם המשיג יחס־קירוב טוב יותר. הוא מסדר את המטלות בסדר יורד של העלות, ואז מחלק אותן לפי אלגוריתם הרשימה. בעולם תיזמון התהליכים, האלגוריתם הזה נקרא Longest Processing Time first, או בקיצור LPT, כי הוא מתזמן קודם את התהליכים שזמן־העיבוד שלהם ארוך ביותר. בעולם חלוקת המספרים, אלגוריתם זה נקרא פשוט **האלגוריתם החמדני**.[1]

אלגוריתם 25.5: האלגוריתם החמדני (חלוקת מטלות לשחקנים עם הערכות זהות)

1. סדר את המטלות בסדר יורד של העלות – מהגדולה לקטנה.

[1] במדעי־המחשב, אלגוריתם המסוגל לטפל בכל קלט מייד כאשר הוא מגיע, ואינו צריך לדעת את כל רשימת הקלטים מראש, נקרא **אלגוריתם מקוון** (online algorithm). אלגוריתם היודע מראש את כל הקלטים נקרא **אלגוריתם לא־מקוון** (offline algorithm).

2. תן את המטלה הבאה לְשחקן, שסכום העלויות הנוכחי שלו קטן ביותר.
3. אם יש עוד מטלות, חזור לשורה 2.

בדוגמה 5.20, המטלה הראשונה בסדרה היא ה־7 והאחרונה היא ה־4. האלגוריתם נותן את ארבעת המטלות הראשונות לארבעת השחקנים בסדר כלשהו, למשל, א-7, ב-7, ג-6, ד-6. שתי המטלות הבאות – 5,5 – ניתנות לשחקנים שהעלות הנוכחי שלהם קטנה ביותר, שהם ג ו-ד. שתי המטלות הבאות ניתנות לשחקנים שהעלות הנוכחית שלהם היא קטנה ביותר, שהם א ו-ב. עכשיו, סכום העלויות של כל ארבעת השחקנים הוא 11, ולכן המטלה האחרונה ניתנת לשחקן כלשהו באופן שרירותי, נניח לשחקן א. בסופו של דבר, עלויות השחקנים הן 11, 11, 11, 15. בדיוק כמו בחלוקה א שבדוגמה 5.20. יחס־הקירוב של האלגוריתם בדוגמה זו הוא 15/12 = 5/4 – טוב יותר משל אלגוריתם הרשימה. המשפט הבא מנתח את יחס־הקירוב של האלגוריתם החמדני באופן כללי.

משפט 26.5

לכל n, יחס־הקירוב של אלגוריתם 5.25 בחלוקת מטלות אגליטרית ל־ n שחקנים הוא לכל היותר $\frac{4}{3} - \frac{1}{3n}$.

למשל, כאשר $n = 4$, יחס הקירוב הוא לכל היותר 4/3 – 1/12 = 15/12 = 1.25, בדיוק כמו בדוגמה 5.20. ניתן להכליל את הדוגמה ולהראות, עבור כל n, דוגמה שבה יחס־הקירוב הוא בדיוק הביטוי שבמשפט 5.26, כך שהמשפט הוא הדוק.

המשפט הוכח ע"י רונאלד גראהאם (Ronald Graham) כבר בשנת 1969, בקשר לבעיית חלוקת תהליכים בין מעבדים.[9] אנחנו נראה הוכחה ישירה ומעט פשוטה יותר.[10]

הוכחת משפט 5.26: בהינתן קלט מסויים לבעיה, נסמן ב־OPT את העלות הגדולה ביותר של סל בחלוקה המיטבית לקלט זה. ננרמל את העלויות ע"י חלוקת ערכי כל המטלות ב־OPT. לאחר נירמול זה, עלויות כל הסלים בחלוקה המיטבית קטנות או שוות 1. לכן סכום עלויות כל המטלות קטן או שווה n, והעלות של כל מטלה לאחר הנירמול קטנה או שווה 1. נחלק את המטלות לשני סוגים:

* מטלות עם עלות גדולה ממש מ־1/3 ייקראו **גדולות**;
* מטלות עם עלות קטנה או שווה ל־1/3 ייקראו **קטנות**.

מהנירמול נובע, שכל סל בחלוקה המיטבית כולל לכל היותר שתי מטלות גדולות, ובסך־הכל יש לכל היותר 2n מטלות גדולות.

* בדוגמה 5.20 (שבה $n = 4$), הנירמול מתבצע ע"י חלוקה ב־12. יש שש מטלות גדולות, עם עלויות 7/12, 7/12, 6/12, 6/12, 5/12, 5/12. שאר המטלות הן קטנות, ולכל אחת מהן עלות 4/12. בחלוקה המיטבית יש שלושה סלים עם שתי מטלות גדולות, בסכום כולל 1, ועוד סל אחד עם שלוש מטלות קטנות, בסכום כולל 1.

האלגוריתם החמדני, על-פי הגדרתו, מחלק קודם את כל המטלות הגדולות, ואז את כל המטלות הקטנות. אנחנו נוכיח את המשפט בשתי טענות־עזר: טענה א מתייחסת לשלב חלוקת המטלות הגדולות, וטענה ב מתייחסת לשלב חלוקת המטלות הקטנות.

טענה א: לאחר שהאלגוריתם סיים לחלק מטלות גדולות, סכום העלויות בידי כל שחקן הוא לכל היותר 1.

הוכחה: אם מספר המטלות הגדולות הוא לכל היותר n, אז האלגוריתם החמדני נותן מטלה גדולה אחת בלבד לכל שחקן, וברור שעלויות כל השחקנים קטנות או שוות 1. לכן נניח שמספר המטלות הגדולות הוא n+t, עבור מספר שלם כלשהו t בין 1 ל־n. לכן בחלוקה המיטבית ישנם t סלים עם שתי מטלות גדולות, ועוד n–t סלים עם מטלה גדולה אחת. נקרא לשתי מטלות גדולות:

* **משודכות** – אם הן נמצאות יחד בסל אחד בחלוקה המיטבית;
* **תואמות** – אם סכום העלויות שלהן קטן או שווה 1.

כל שתי מטלות משודכות הן תואמות. העלות של כל מטלה, התואמת למטלה גדולה אחרת, קטן מ־2/3. ישנם t זוגות של מטלות גדולות משודכות (ולכן תואמות), ועוד n–t מטלות לא־משודכות. לכן:

* מתוך מטלות 1 עד n–t+1, לפחות אחת משודכת (ולכן תואמת) למטלה אחרת כלשהי. לכן, המטלה הקטנה ביותר בקבוצה זו, שהיא מטלה n–t+1, בהכרח תואמת למטלה הקטנה ביותר מבין הגדולות, שהיא מטלה n+t.
* מתוך מטלות 1 עד n–t+2, לפחות שתיים משודכות (ולכן תואמות) למטלות אחרות כלשהן. לכן, המטלה הקטנה ביותר בקבוצה זו, שהיא מטלה n–t+2, בהכרח תואמת למטלה השניה בקוטנה מבין הגדולות, שהיא מטלה n+t–1.
* משיקולים דומים, מטלה n–t+3 תואמת למטלה n+t–2, ובאופן כללי, מטלה n–t+k תואמת למטלה n+t–k+1 לכל k בין 1 ל־t. בפרט, מטלה n תואמת למטלה n+1.

עכשיו נחזור לאלגוריתם החמדני. האלגוריתם מחלק את מטלות 1, ..., n, מטלה אחת לכל שחקן. כשהוא מגיע למטלה גדולה n+1, הוא נותן אותה לשחקן שקיבל את המטלה הקטנה ביותר שחולקה עד כה, שהיא מטלה n. כאמור בנקודה השלישית למעלה, מטלה n תואמת למטלה n+1, ולכן סכום העלויות של שחקן זה הוא לכל היותר 1.

כשהאלגוריתם מגיע למטלה גדולה n+2, הוא נותן אותה לשחקן שקיבל את מטלה n–1. כאמור בנקודות למעלה, שתי המטלות הללו תואמות, כלומר סכומן לכל היותר 1, וכל אחת מהן קטנה מ־2/3. לעומת זאת, סכום העלויות של השחקן שקיבל את המטלות n, n+1 גדול מ 2/3. לכן האלגוריתם אכן נותן את מטלה n+2 לשחקן שקיבל את n–1, וערכו של שחקן זה נשאר לכל היותר 1.

באותו אופן, לכל k בין 1 ל־t, האלגוריתם נותן את מטלה n+t–k+1 לשחקן שקיבל את n–t+k. הזוגות הללו תואמים לכל k, ולכן עלויות כל השחקנים לכל היותר 1. בכך הוכחנו את טענה א.

טענה ב: כאשר האלגוריתם נותן מטלה קטנה לשחקן כלשהו, סכום העלויות החדש של השחקן (כולל המטלה שקיבל) קטן או שווה $\frac{4}{3} - \frac{1}{3n}$.

הוכחה: נסמן את עלות המטלה ב־x. סכום העלויות של המטלות שחולקו עד כה הוא לכל היותר $n - x$. לפי כלל שובך־היונים, סכום העלויות של השחקן המקבל את המטלה הוא לכל היותר

$\frac{n-x}{n} = 1 - \frac{x}{n}$ לפני שקיבל את המטלה, ולכן לכל היותר $1 - \frac{x}{n} + x = 1 + x \cdot \left(1 - \frac{1}{n}\right)$

לאחר שקיבל את המטלה. כיוון שהמטלה קטנה, $x \le \frac{1}{3}$. לכן סכום העלויות החדש של

השחקן הוא לכל היותר $1 + \frac{1 - \frac{1}{n}}{3} = \frac{4}{3} - \frac{1}{3n}$.

בכך סיימנו את הוכחת טענה ב. משתי הטענות יחד נובעת נכונות המשפט.

מש"ל.

בחנו את עצמכם 27.5

הוכיחו, שיחס הקירוב במשפט 5.26 הוא הדוק לכל n. כלומר: לכל n, הראו דוגמה של בעיית חלוקת מטלות עם n שחקנים, שבה הערך האגליטרי המיטבי הוא 3n–, אבל הערך האגליטרי של החלוקה המוחזרת ע"י אלגוריתם 5.25 הוא (4n–1)–.

רמז: נסו להכליל את דוגמה 5.21 (המתאימה למקרה הפרטי n=4).

עד עכשיו דיברנו על חלוקת **מטלות** (עם ערכים שליליים). ניתן להשתמש באלגוריתם החמדני גם לחלוקת **חפצים** (עם ערכים חיוביים). האלגוריתם עובד באותו אופן: הוא מסדר את החפצים בסדר יורד של הערך שלהם, ונותן כל חפץ לשחקן שהערך הנוכחי שלו קטן ביותר. יחס־הקירוב של האלגוריתם נתון ע"י המשפט הבא:

משפט 28.5

לכל n, יחס־הקירוב של אלגוריתם 5.25 בחלוקת חפצים אגליטרית ל־n שחקנים הוא לפחות (4n–2)/(3n–1).

משפט 5.28 הוכח בהדרגה בכמה מאמרים על־ידי חוקרים שונים.[11] כל ההוכחות הידועות כיום הן ארוכות ומסובכות הרבה יותר משל משפט 5.26, ולא נראה אותן כאן. נראה רק דוגמה שיחס־הקירוב הוא הדוק. לשם פשטות, הדוגמה מתייחסת ל־n זוגי (ניתן לבנות דוגמה דומה ל־n איזוגי).

דוגמה 29.5

יש 3n–1 חפצים. הערכים של 2n החפצים הגדולים הם:

2n–1, 2n–1; 2n–2, 2n–2; ...; n, n.

הערכים של n–1 החפצים הקטנים כולם n. בהרצה של האלגוריתם החמדני, לאחר חלוקת 2n החפצים הגדולים, מתקבלת החלוקה הבאה:

- שני השחקנים הראשונים מקבלים: 2n–1, n.
- שני השחקנים הבאים מקבלים: 2n–2, n+1.
- ...
- שני השחקנים האחרונים מקבלים: 3n/2, 3n/2–1.

כעת סכום הערכים של כל השחקנים הוא שווה: 3n–1. האלגוריתם מחלק את n–1 החפצים הקטנים ל־n–1 שחקנים כלשהם, ושחקן אחד נשאר עם סל בשווי של 3n–1 בלבד.

אבל קיימת חלוקה טובה יותר:

- השחקן הראשון מקבל 2n–1, 2n–1.
- שני השחקנים הבאים מקבלים 2n–2, n, n.
- שני השחקנים הבאים מקבלים 2n–3, n+1, n.
- ...
- שני השחקנים הבאים מקבלים 3n/2, 3n/2–2, n.
- השחקן האחרון מקבל 3n/2–1, 3n/2–1, n.

בחלוקה זו הערך האגליטרי הוא 4n–2. לכן יחס־הקירוב הוא בדיוק (3n–1)/(4n–2).

שימו לב: בחלוקת חפצים (משפט 5.28) יחס הקירוב קטן מ־1, כי כשהערכים חיוביים, יחס־קירוב **גדול** יותר מבטיח ערך גדול יותר. לעומת זאת, בחלוקת מטלות (משפט 5.26) יחס הקירוב גדול מ־1, כי כשהערכים שליליים, יחס־קירוב **קטן** יותר מבטיח ערך גדול יותר. בשני המקרים, יחס הקירוב טוב יותר ככל שהוא קרוב יותר ל־1.

ישנם אלגוריתמים נוספים המשיגים קירוב גורם־קבוע. לדוגמה: אלגוריתם **ההפרש הגדול ביותר** (Largest Differencing Method), של נרנדרה קרמרקאר וריצ'ארד קארפ,[12] משיג יחס־קירוב דומה לזה של האלגוריתם החמדני. אלגוריתם בשם **MultiFit**, של קופמן גארי וג'ונסון,[13] משיג יחס־קירוב טוב יותר - 13/11 – בחלוקת מטלות (לא ידוע מה יחס־הקירוב שלו בחלוקת חפצים). בסעיף הבא נראה שאפשר להשיג יחס־קירוב טוב כרצוננו.

5.5.2 חלוקה אגליטרית – קירוב טוב כרצוננו

אנחנו רוצים לחזק את התוצאות של הסעיף הקודם בשתי דרכים. ראשית, אנחנו רוצים לטפל גם בחלוקה בין שחקנים עם הערכות שונות. שנית, אנחנו רוצים להשיג יחס־קירוב טוב יותר (קרוב יותר ל־1). אנחנו רוצים שיטה כללית ליצירת אלגוריתמי קירוב, שמקבלת קבוע חיובי כלשהו $\varepsilon>0$, ומייצרת אלגוריתם־קירוב עם יחס־קירוב לפחות $\varepsilon-1$ (עבור בעיית מינימום), או לכל היותר $\varepsilon+1$ (עבור בעיית מקסימום). שיטה כזאת נקראת באנגלית **Approximation Scheme**. אם אלגוריתמי־הקירוב שהשיטה מייצרת רצים בזמן פולינומיאלי בגודל הקלט, לכל ε בנפרד, אז השיטה נקראת Polynomial-Time Approximation Scheme ובקיצור PTAS. אם האלגוריתמים רצים בזמן שהוא פולינומיאלי גם בגודל הקלט וגם ב־$1/\varepsilon$, אז השיטה נקראת Fully Polynomial-Time Approximation Scheme, ובקיצור FPTAS.

בסעיף זה נראה FPTAS לבעיית החלוקה האגליטרית, עבור כל מספר קבוע n של שחקנים. הבסיס לאלגוריתם זה יהיה אלגוריתם 5.7 – חיפוש במרחב המצבים תוך גיזום מצבים זהים.[ז] נשתמש בשיטה כללית שהציג גרהארד ווגינגר,[14] המאפשרת להפוך כל אלגוריתם חיפוש הגוזם מצבים זהים, בדומה לאלגוריתם 5.7, ל־FPTAS. נציג כאן מקרה פרטי של השיטה שלו, המספיק לצורך המטרה המעניינת אותנו, שהיא מציאת חלוקה אגליטרית.

כזכור, זמן הריצה של אלגוריתם 5.7 תלוי במספר המצבים הכולל. מטרת השיטה של ווגינגר היא להקטין את מספר המצבים. בכל שלב, מחלקים את המצבים ל"תאים". בכל תא שיש בו יותר ממצב אחד, משאירים רק מצב אחד. כתוצאה מכך, מספר המצבים הוא לכל היותר כמספר התאים. מחיקת המצבים עלולה לגרום אי־דיוק, אבל גודל אי־דיוק זה הוא לכל היותר כגודלו של תא אחד.

גודל התאים תלוי באותו מספר חיובי ε, המייצג את רמת הקירוב הרצויה. נגדיר:

$$r := 1 + \frac{\varepsilon}{2m}.$$

כאשר m הוא מספר החפצים. ננרמל את הקלט כך שהערך החיובי הקטן ביותר שיכול להיות בוקטור המצב (הערך החיובי הקטן ביותר שמייחס שחקן כלשהו לחפץ כלשהו) הוא 1. נגדיר אינטרוואלים בין חזקות סמוכות של r:

$$[1,r),\ [r,r^2),\ [r^2,r^3),\ldots$$

מספר האינטרוואלים ייקבע כך שכל ערך שיכול להיות בווקטור־המצב נמצא באינטרוואל כלשהו. בפרט, אם הערך הגדול ביותר, שיכול להופיע בווקטור־המצב, הוא V, אז מספר האינטרוואלים יהיה הלוגריתם בבסיס r של V, מעוגל כלפי מעלה, ועוד אינטרוואל אחד עבור הערך 0. נסמן את מספר האינטרוואלים באות L. בקירוב, מתקיים:

$$L \approx \log_r(V) = \frac{\log(V)}{\log(r)}$$

תא הוא מכפלה קרטזית של n אינטרוואלים, ולכן כל מצב אפשרי נמצא בתא אחד בדיוק. מספר התאים הכולל הוא L^n ולכן מספר המצבים הכולל יהיה לכל היותר L^n. כעת נציג את ה־FPTAS. האלגוריתם מחזיק קבוצות־מצבים שנסמן באות S[t], כאשר t בין 0 ל־m.

אלגוריתם 30.5: חישוב חלוקה אגליטרית – FPTAS

א. אתחול: S[0] := קבוצה עם מצב אחד, שהוא וקטור עם n אפסים.

ב. לכל t בין 1 ל־m, חשב את קבוצת המצבים ה-S[t], t, באופן הבא:

* לכל מצב בקבוצה S[t-1], חשב את כל n המצבים הנובעים מנתינת חפץ t לשחקן כלשהו.

ז אלגוריתם־חיפוש הגוזם רק מצבים זהים נקרא בספרות גם **תיכנות דינאמי – Dynamic programming**.

* לכל מצב חדש s: אם הקבוצה S[t] מכילה מצב הנמצא באותו תא של המצב s, גזום את s. אחרת, הוסף את s לקבוצה S[t].

ג. בקבוצה S[m], מצא מצב עם הערך האגליטרי הגדול ביותר, והחזר את החלוקה המתאימה.

משפט 31.5

לכל n קבוע, אלגוריתם 5.30 הוא FPTAS לחישוב חלוקה אגליטרית בין n שחקנים.

הוכחה: עלינו לנתח את יחס־הקירוב של האלגוריתם, ואת זמן־הריצה שלו.

ניתוח יחס־הקירוב: בכל שלב, האלגוריתם משאיר רק מצב אחד בכל תא, וזורק מצבים אחרים הנמצאים באותו תא. לכן, בכל שלב נוצרת שגיאה כפלית של r לכל היותר, בכל קואורדינטה בווקטור. לאחר m שלבים, השגיאה הכפלית המצטברת היא לכל היותר:

$$r^m = \left(1 + \frac{\varepsilon}{2m}\right)^m \approx 1 + \frac{\varepsilon}{2}$$

מספר זה קטן גם מ־ $1 + \varepsilon$ וגם מ־ $\frac{1}{1 - \varepsilon}$. לכן, יחס־הקירוב בבעיית מקסימום (חלוקה אגליטרית של חפצים עם ערך חיובי) קטן מ־(ε+1), ויחס הקירוב בבעיית מינימום (חלוקה אגליטרית של מטלות עם ערך שלילי) גדול מ־ (1- ε), בהתאם להגדרה של FPTAS.

ניתוח זמן־הריצה: כמו באלגוריתם 5.7, גם כאן זמן הריצה נקבע ע"י מספר המצבים. מספר המצבים הוא לכל היותר כמספר התאים, שהוא L^n.

$$L \approx \log_r(V) = \frac{\log(V)}{\log(r)}$$

לפי הקירוב הידוע $\log(1 + x) \approx x$, והצבת ההגדרה של r, נקבל:

$$L \approx \frac{\log(V)}{\frac{\varepsilon}{2m}} \approx O\left(2m \cdot \frac{\log(V)}{\varepsilon}\right)$$

הביטוי $m \cdot \log(V)$ עומד ביחס ישר למספר הסיביות הדרושות לייצוג הקלט. לכן, עבור כל n קבוע, מספר המצבים L^n הוא פולינומיאלי בגודל הקלט וב־$\frac{1}{\varepsilon}$, בהתאם להגדרה של FPTAS.

מש"ל.

שימו לב: זמן הריצה של אלגוריתם 5.30 הוא **פולינומיאלי** במספר החפצים m וברמת הדיוק $\frac{1}{\varepsilon}$, אבל **מעריכי** במספר השחקנים n. לכן הוא FPTAS רק כאשר מספר השחקנים נחשב **פרמטר קבוע**, שאינו חלק מהקלט לבעיה. כשמספר השחקנים גדול, האלגוריתם אינו שימושי.

לעומת זאת, האלגוריתם החמדני (אלגוריתם 5.25) רץ בזמן פולינומיאלי גם כאשר מספר השחקנים הוא **חלק מהקלט**, והוא מהיר ושימושי גם כשמספר השחקנים גדול. החיסרון שלו הוא, שיחס־הקירוב שלו אינו יכול להיות טוב כרצוננו. כמו כן, הוא עובד רק על שחקנים עם הערכות זהות.

אלגוריתם שלישי פותח על-ידי נוגה אלון, יוסי אזר, טל ידיד, וגרהארד ווגינגר.[15] האלגוריתם פותר בעייה כללית; מקרה פרטי של בעייה זו הוא חלוקה אגליטרית (של חפצים או מטלות) בין שחקנים עם הערכות **זהות**. זמן הריצה של אלגוריתם זה הוא פולינומיאלי במספר השחקנים n ומספר החפצים m, אבל מעריכי ברמת הדיוק $\frac{1}{\varepsilon}$.

האם קיים אלגוריתם המשלב את היתרונות של כל האלגוריתמים הנ"ל – גם נותן קירוב טוב כרצוננו (FPTAS) וגם רץ בזמן פולינומיאלי בכל הפרמטרים של הבעיה $(n, m, \frac{1}{\varepsilon})$? בהנחה ש־ $P \neq NP$, התשובה היא לא: כאשר n הוא חלק מהקלט, בעיית החלוקה האגליטרית (אפילו במקרה הפרטי של שחקנים עם הערכות זהות) היא **NP־קשה במובן החזק**, ונובע מכך שלא קיים לה FPTAS. ההוכחה של עובדה זו היא מעבר להיקפו של ספר זה.

אם כך, כשיש הרבה שחקנים עם הערכות שונות, בעיית מציאת חלוקה אגליטרית היא קשה חישובית. מה בכל־זאת אפשר לעשות במצב זה? נראה בסעיף הבא.

5.5.3 אלגוריתמים נוספים **

המקרה הפרטי של חלוקה אגליטרית, שבו לכל השחקנים הערכות **זהות** וזכויות **שוות**, נחקר לעומק בספרות המדעית. המקרה הפרטי הזה נקרא multiway number partitioning – כיוון שמדובר בחלוקה של מספרים. כינוי נוסף הוא identical machine scheduling – כיוון שאחד היישומים הראשונים שלו הוא תיזמון תהליכים על מכונות עם קצב־ריצה זהה.

אחד האלגוריתמים המעשיים לפתרון מיטבי של בעיה זו נקרא **האלגוריתם החמדני השלם – complete greedy algorithm**.[16] זהו אלגוריתם חיפוש במרחב המצבים, הבונה את עץ־החלוקות **לעומק**, כאשר סדר הוספת החפצים הוא **מהגדול לקטן**. העלה הראשון שהאלגוריתם מגיע אליו הוא העלה המייצג את החלוקה שמחזיר האלגוריתם החמדני (אלגוריתם 5.25). בכל פעם שהאלגוריתם מגלה עלה חדש, הוא בודק אם הערך האגליטרי שלו גבוה יותר מהערך האגליטרי הגבוה ביותר שנמצא עד כה, ואם כן, הוא מעדכן ערך זה. האלגוריתם גוזם כל קודקוד, שהחלוקה האופטימית שלו אינה מובילה לערך אגליטרי גבוה יותר. גיזום זה מקטין באופן משמעותי את מרחב החיפוש.

תכונה שימושית ביותר של אלגוריתם זה היא, שניתן להקציב לו זמן מירבי לריצה (למשל: 30 שניות), והוא יחזיר לנו את החלוקה עם הערך האגליטרי הגבוה ביותר שהצליח למצוא עד אותו זמן. ככל שנקציב לו זמן רב יותר, כך נקבל חלוקה טובה יותר. אם נקציב לו זמן בלתי־מוגבל, נקבל את החלוקה המיטבית, אבל גם אם זמננו קצר, עדיין נקבל חלוקה טובה. החלוקה שתוחזר תהיה טובה לפחות כמו החלוקה שמחזיר האלגוריתם החמדני, אך ייתכן שתהיה טובה יותר. הדבר מאפשר לנו לקבל את החלוקה הטובה ביותר שאפשר למצוא במסגרת הזמן שברשותנו. אלגוריתם המקיים תכונה זו נקרא **אלגוריתם לכל עת – anytime algorithm**.

בשנים האחרונות פותחו אלגוריתמים רבים נוספים, הבונים את עץ־החלוקות בצורה שונה ומשתמשים בכללי־גיזום מתקדמים. הם תוארו ונחקרו לעומק על־ידי ריצ'ארד קורף, איתן שרייבר, ומיכאל מופיט.[17] האלגוריתמים המתקדמים ביותר המתוארים במאמרם מצליחים לפתור בזמן סביר (כ-2 דקות) בעיות חלוקה עם 60 חפצים, 12 שחקנים, ומספרים עם 48 סיביות.

שאלה פתוחה 5.32

א. איך אפשר להרחיב את האלגוריתמים הללו לשחקנים עם **זכויות** שונות?

ב. איך אפשר להרחיב את האלגוריתמים הללו לשחקנים עם **הערכות** שונות?

בסעיף זה עסקנו באלגוריתמי קירוב המשיגים קירוב **כפלי** – ערך הפתרון שנמצא שווה לגורם כלשהו כפול הערך המיטבי. סוג אחר של קירוב, הדומה לקירובים שלמדנו עליהם בפרק 4, הוא חלוקה אגליטרית **עד־כדי חפץ אחד**. חלוקה זו נותנת לכל שחקן את הערך האגליטרי האופטימלי פחות ערכו של חפץ אחד לכל היותר. קיימים אלגוריתמים

** ** תת-סעיף זה מומלץ למתקדמים בלבד.

המוצאים חלוקה כזאת לשחקנים עם הערכות כלשהן – זהות או שונות. האלגוריתמים אלה משתמשים בשיטות שנלמד בפרק 6 (סעיף 6.4), ולכן נתאר אותם שם.

בפרק 3 ראינו, שחלוקה אגליטרית היא לא תמיד יעילה־פארטו. כדי להשיג יעילות פארטו, צריך להשתמש בעקרון ה**לקסימין**. כשמחלקים משאבים רציפים, ישנו אלגוריתם יעיל המוצא חלוקה לקסימין־אגליטרית; האלגוריתם הזה אינו עובד כשהחפצים בדידים, כי מרחב החלוקות אינו מרחב קמור (ראו הגדרה בפרק 3). סילבן בוברט ומישל למיטר פיתחו מספר אלגוריתמי חיפוש במרחב המצבים לחישוב חלוקה לקסימין־אגליטרית של חפצים בדידים.[18]

5.6 סיכום פרק 5

בפרק זה ראינו הרבה אלגוריתמים לחלוקת חפצים בדידים – איך בוחרים באיזה אלגוריתם להשתמש?

ראשית, צריך להחליט מהו הקריטריון שאותו רוצים למטב: האם זה סכום הערכים – בהתאם לעקרון **האוטיליטרי**; או הערך הקטן ביותר – בהתאם לעקרון **האגליטרי**; או מכפלת הערכים – בהתאם לעקרון של נאש. השאלה הזו היא שאלה ערכית – אין לה פתרון אלגוריתמי או מתימטי. התובנה העיקרית שניתן להפיק מהמתמטיקה היא, שמיקסום מכפלת הערכים נותן חלוקה ללא־קנאה עד־כדי חפץ אחד (EF1); אולם אפשר להשיג תוצאה דומה על־ידי מיקסום סכום הערכים או הערך הקטן ביותר תחת אילוץ של EF1.

לאחר שבחרנו קריטריון למיטוב, יש לבחור אלגוריתם. ראינו כמה סוגים כלליים של אלגוריתמים, שניתן להתאים אותם לכל אחד מהקריטריונים. שני סוגי אלגוריתמים מאפשרים למצוא את הפתרון המיטבי במדוייק:

- **תיכנות ליניארי בשלמים**: היתרון העיקרי שלו הוא נוחות המימוש. בעזרת כלים קיימים לפתרון בעיות מיטוב. קל לנסח בעיות מיטוב שונות ומורכבות, ולפתור אותן בזמן קצר יחסית.
- **חיפוש במרחב המצבים**: דורש מעט יותר עבודה מתיכנות ליניארי בשלמים, בגלל הצורך לממש את החסמים לכל בעיה בנפרד. היתרון הוא, שאפשר למצוא את הפתרון המיטבי במהירות רבה יותר.

לשני סוגי האלגוריתמים הללו יש זמן־ריצה מעריכי במקרה הגרוע. לכן, אם אנחנו מנסים לפתור בעיה גדולה, עם הרבה שחקנים וחפצים, ייתכן שלא נצליח למצוא את הפתרון המדוייק בזמן סביר. כדי להתמודד עם בעיה זו, למדנו שני סוגים של אלגוריתמי קירוב:

- **אלגוריתם קירוב גורם־קבוע**: זהו אלגוריתם פשוט ומהיר ביותר, אולם הוא לא מתחייב למצוא את הפתרון המיטבי – הוא מתחייב רק שערך הפתרון שנמצא יהיה גורם קבוע מסויים כפול הפתרון המיטבי.
- **שיטת קירוב פולינומיאלית מלאה (FPTAS)**: מאפשרת לנו למצוא קירוב טוב כרצוננו, כאשר זמן הריצה הוא פולינומיאלי באיכות הקירוב. אפשר להשקיע זמן רב יותר – ולמצוא קירוב טוב יותר. החיסרון הוא, שהמימוש מסובך יותר, וזמן הריצה בפועל עשוי להיות איטי מאד, למרות שזמן הריצה התיאורטי הוא פולינומיאלי.

5.7 נספח: חלוקה מקסימין־הוגנת **

סעיף זה הוא נספח לפרק 4 העוסק בחלוקה הוגנת־בקירוב. הוא מובא כאן, בסוף פרק 5, כי הוא משתמש באלגוריתם 5.25 (האלגוריתם החמדני לחלוקת חפצים זהים) ובמשפטים הקשורים אליו.

בפרק 2 למדנו על **הוגנות־מקסימין** – תנאי הוגנות חליפי לפרופורציונליות, שניתן להשיג גם כאשר חלוקה פרופורציונלית אינה קיימת. נזכיר את ההגדרות כאן. נניח שיש לנו n שחקנים עם הערכות שונות. אנחנו מבקשים מכל אחד מהם למצוא חלוקה אגליטרית בין n שחקנים **שיש להם הערכות זהות לשלו**. החלוקה הזאת נקראת **חלוקת המקסימין של השחקן** (maximin partition) והערך האגליטרי (הערך הקטן ביותר של סל בחלוקה זו) נקרא **ערך המקסימין של השחקן** (maximin share, MMS). חלוקה שבה הערך של כל שחקן הוא לפחות ערך־המקסימין שלו נקראת חלוקה **מקסימין־הוגנת** (maximin fair).[19]

ניתן לראות בחלוקה מקסימין־הוגנת קירוב לחלוקה האגליטרית. ניתן גם לראות בה קירוב לחלוקה פרופורציונלית, כפי שמראה הלמה הבאה.

לֶמֶה 33.5

כשיש n שחקנים עם הערכות חיבוריות, ערך המקסימין של כל שחקן הוא לכל היותר n/1 מהערך הכולל של החפצים בעיניו.

הוכחה: נסמן ב־v את ההערכה של השחקן, ב־$X_1,\dots,X_n$ את חלוקת המקסימין של השחקן, וב־M את קבוצת כל החפצים. מתכונת החיבוריות נובע:

$$v(M) = v(X_1) + \dots + v(X_n)$$

נסמן ב־MMS את ערך המקסימין של השחקן. מהגדרת ערך המקסימין נובע, שערכו של כל סל בחלוקה הנ"ל הוא לפחות MMS. לכן:

$$v(M) \geq MMS + \dots + MMS = n \cdot MMS$$

$$MMS \leq \frac{v(M)}{n}$$

מש"ל.

5.7.1 קיום חלוקה מקסימין־הוגנת

מלֶמָה 5.33 נובע, שכל חלוקה פרופורציונלית (אם קיימת חלוקה כזאת) היא מקסימין־הוגנת. אבל חלוקה פרופורציונלית לא תמיד קיימת, אפילו כשיש שני שחקנים וחפץ אחד. האם תמיד קיימת חלוקה מקסימין־הוגנת? כשיש שני שחקנים, התשובה היא כן.

** ** תת-סעיף זה מומלץ למתקדמים בלבד.

משפט 34.5

לכל שני שחקנים עם הערכות חיבוריות, קיימת חלוקה מקסימין־הוגנת.

הוכחה: אלגוריתם "חתוך ובחר" מוצא חלוקה מקסימין־הוגנת. שחקן א מחלק את החפצים לשתי קבוצות, בהתאם לחלוקת המקסימין שלו. שחקן ב בוחר את הקבוצה עם הערך הגבוה יותר. מתכונת החיבוריות נובע, שהערך של קבוצה זו הוא לפחות 1/2 מהערך הכולל של שחקן ב; לפי לֶמָה 5.33, ערך זה הוא לפחות ערך המקסימין של שחקן ב. שחקן א לוקח את הסל הנשאר; לפי הגדרת חלוקת המקסימין, ערכו של סל זה הוא לפחות ערך המקסימין של שחקן א.

מש"ל.

עבור שלושה שחקנים או יותר, התשובה היא לא. זה התגלה כבר בשנת 2014 ע"י אריאל פרוקצ'יה וג'ונקסינג וואנג.[20] אנחנו נראה דוגמה קטנה יותר, שגילו אוריאל פייגה, אריאל ספיר ולליב טאובר.[21]

משפט 35.5

לשלושה שחקנים עם הערכות חיבוריות, ייתכן שאף חלוקה אינה מקסימין־הוגנת.

הוכחה: נראה דוגמה עם שלושה שחקנים ותשעה חפצים. לכל אחד מהשחקנים, סכום הערכים של כל החפצים הוא 120, ויש חלוקה לשלוש קבוצות שבה הערך של כל סל הוא בדיוק 40. לכן, ערך המקסימין של כל שחקן הוא בדיוק 40. נראה שבכל חלוקה, יש שחקן שערכו לכל היותר 39. החפצים מסודרים בריבוע של 3 על 3. הערכים של שחקן א הם:

1, 16, 23

26, 4, 10

12, 19, 9

בחלוקה לפי **שורות**, סכום הערכים בכל סל הוא 40. הערכים של שחקן ב הם:

1, 16, 22

26, 4, 9

13, 20, 9

בחלוקה לפי **עמודות**, סכום הערכים בכל סל הוא 40. הערכים של שחקן ג הם:

1, 15, 23

25, 4, 10

13, 20, 9

בחלוקה {15,25}, {4,13,23}, {1,9,10,20}, סכום הערכים בכל סל הוא 40.

נניח בשלילה, שקיימת חלוקה מקסימין־הוגנת. לכל חפץ, נסתכל על הערך הגדול ביותר שהוא נותן לשחקן כלשהו (המקסימום בין שלוש ההערכות של השחקנים):

1, 16, 23

26, 4, 10

13, 20, 9

בחלוקה מקסימין־הוגנת, הערך של כל שחקן הוא לפחות 40, ולכן הערך של כל סל לפי מטריצה זו הוא לפחות 40. כיוון שסכום הערכים במטריצה הוא 122, יש ערך "עודף" של 2 בלבד, ולכן הערך של כל סל הוא לכל היותר 42. נבדוק כמה חפצים יש בכל סל:

* אם סל מסויים מכיל חפץ אחד, אז הערך בסל הזה הוא לכל היותר 26 – סתירה.

* אם סל מסויים מכיל שני חפצים, אז החפצים האלה חייבים להיות ה־16 וה־26, כי זה הזוג היחיד שהסכום שלו בין 40 ל־42. הדרך היחידה לחלק את שבעת החפצים הנותרים לשני סלים עם ערך 40 היא: 4,13,23 ו: 1,9,10,20. אבל שני הסלים האלה שווים 39 עבור שחקנים א, ב. לכן, שחקן א או ב בהכרח מקבל לכל היותר 39 – סתירה.

* מכאן, שכל סל מכיל בדיוק שלושה חפצים. נסתכל על הסל שמכיל את ה־9. יש רק שתי דרכים להשלים אותו לסל שערכו בין 40 ל־42: או להוסיף לו את החפצים בשורה השלישית (9, 13, 20), או את החפצים בעמודה השלישית (9, 10, 23). בשני המקרים, סכום הערכים בסל זה הוא 42, ולכן יש לחלק את החפצים הנותרים לשני סלים עם ערך 40. הדרך היחידה לחלק את החפצים הנותרים לשני סלים עם ערך 40 היא לפי שורות (במקרה הראשון) או לפי עמודות (במקרה השני). בחלוקה לפי שורות, שתי השורות העליונות שוות 39 עבור שחקנים א, ב. בחלוקה לפי עמודות, שתי העמודות השמאליות שוות 39 עבור שחקנים ב, ג. לכן, שחקן אחד בהכרח מקבל לכל היותר 39 – סתירה.

מש"ל.

5.7.2 מציאת חלוקה מקסימין־הוגנת בקירוב

לאור משפט 5.35, אנחנו רוצים למצוא חלוקה הנותנת לכל שחקן ערך שהוא קרוב לערך המקסימין שלו. אנחנו נראה אלגוריתם אחד, של ברמן וקרישנמורתי.[22] האלגוריתם מעניין כי הוא מאד דומה לאלגוריתם החמדני (אלגוריתם 5.25), וגם משיג יחס־קירוב דומה.

כזכור, האלגוריתם החמדני עובר על החפצים (או המטלות) בסדר יורד של הערך שלהם (בערך מוחלט), ונותן את החפץ הבא לשחקן שסכום־הערכים הנוכחי שלו הוא קטן ביותר. שני אתגרים עומדים בפנינו, כשאנו באים להתאים את האלגוריתם הזה לשחקנים עם ערכים שונים:

- סדר הערכים עלול להיות שונה לשחקנים שונים. זה מציב בפנינו אתגר כבר בשלב הראשון של האלגוריתם – באיזה סדר נעבור על החפצים?
- גם אם סדר הערכים זהה לכל השחקנים, סכום הערכים בסל מסויים עלול להיות שונה לשחקנים שונים. זה מציב בפנינו אתגר בכל צעד – איך נחליט למי לתת את החפץ הבא?

ראשית, נתמודד עם אתגר הסדר. נניח שיש לנו אלגוריתם כלשהו, שעובד רק על **בעיה מסודרת** – בעיה שבה כל השחקנים מסכימים על סדר הערכים של החפצים (ראו בפרק 4, בסעיף על חלוקה ללא קנאה מלבד חפץ כלשהו). נניח שאנחנו מקבלים בעיה כללית - לא בהכרח מסודרת. נפעיל את האלגוריתם הקיים באופן הבא.[23]

אלגוריתם 36.5: שימוש באלגוריתם קיים, הפועל על בעיות מסודרות בלבד

אלג

א. בחר סדר כלשהו על החפצים ומספר אותם לפי הסדר: 1,...,m.

ב. הגדר לכל שחקן הערכה חדשה "מסודרת", באופן הבא:
* הערך החדש של חפץ 1 := הערך הישן הגבוה ביותר;
* הערך החדש של חפץ 2 := הערך הישן השני בגובהו;
...
* הערך החדש של חפץ k := הערך הישן ה־k בגובהו;
...
* הערך החדש של חפץ m := הערך הישן הנמוך ביותר.
(אם יש חפצים עם ערך שווה, הסדר ביניהם נקבע שרירותית).

ג. התקבלה בעיה מסודרת; הפעל עליה את האלגוריתם הקיים.

ד. בנה חלוקה חדשה עבור הבעיה המקורית, באופן הבא:
* השחקן שקיבל בשלב ג את חפץ 1, מקבל את החפץ הטוב ביותר בעיניו.
* השחקן שקיבל את חפץ 2, מקבל את החפץ הטוב ביותר בעיניו מאלה שנשארו
...
* השחקן שקיבל את חפץ k, מקבל את החפץ הטוב ביותר בעיניו מאלה שנשארו
...
* השחקן שקיבל בשלב ג את חפץ m, מקבל את החפץ האחרון שנשאר.

דוגמה.[24] יש שני שחקנים בשם A, B וארבע מטלות בשם w,x,y,z. ההערכות המקוריות הן:

- w, x, y, z
- A: -3, -5, -6, -1
- B: -2, -8, -4, -9

ההערכות המסודרות הן:

- 1, 2, 3, 4
- A: -1, -3, -5, -6
- B: -2, -4, -8, -9

נניח שהאלגוריתם הנתון, בשלב ג, נותן לשחקן A את מטלות 2, 4, ונותן לשחקן B את מטלות 1, 3. בחלוקה זו, הערך של שחקן A הוא -9 והערך של שחקן B הוא -10.

בשלב ד, שחקן B בוחר ראשון – כי הוא קיבל את 1. הוא בוחר את המטלה הטובה ביותר בעיניו, שהיא w. שחקן A בוחר שני – כי הוא קיבל את 2. הוא בוחר את המטלה הטובה ביותר בעיניו מאלה שנשארו, שהיא z. שחקן B בוחר שלישי ולוקח את y, ושחקן A מקבל את x.
בחלוקה הסופית, הסל של שחקן A הוא $\{z,x\}$ וערכו -6. הסל של שחקן B הוא $\{w,y\}$ וערכו -6. שני השחקנים קיבלו ערך גבוה יותר מהערך שהיה להם בחלוקה של שלב ג. זה נכון באופן כללי:

משפט 37.5

כאשר לכל השחקנים יש הערכות חיבוריות, הערך של כל שחקן בחלוקה שמחזיר אלגוריתם 5.36 גדול לפחות כמו בחלוקה של שלב ג.

הוכחה: נניח ששחקן כלשהו מקבל בשלב ג את חפץ מספר k. אז בשלב ד הוא יקבל חפץ בסיבוב ה־k. בסיבוב זה, לפחות אחד מבין החפצים k,...,1 עדיין זמין. לכן השחקן יקבל חפץ, השווה בעיניו **לפחות** כמו החפץ ה־k. באופן כללי, לכל חפץ שקיבל השחקן בשלב ג, הוא מקבל בשלב ד חפץ השווה בעיניו לפחות באותה מידה. מתכונת החיבוריות נובע, שסכום הערכים שקיבל השחקן בשלב ד גדול בעיניו לפחות כמו סכום הערכים שקיבל בשלב ג.

מש"ל.

ממשפט 5.37 נובע, שאם יש לנו אלגוריתם לבעיות מסודרות, המוצא חלוקה שבה כל שחקן מקבל ערך מעל סף כלשהו (למשל: אחוז כלשהו מערך המקסימין), אנחנו יכולים להשתמש בו כדי למצוא חלוקה בעלת תכונות דומות בבעיות כלליות. לכן נניח מעכשיו שיש לנו בעיה מסודרת. נסדר את החפצים בסדר יורד של הערך שלהם, ונתחיל להפעיל את האלגוריתם החמדני. האלגוריתם קובע, שיש לתת את החפץ הבא לשחקן שהערך שלו קטן ביותר; אבל שחקנים שונים יכולים לייחס ערכים שונים לאותו סל. איך נקבע למי לתת את החפץ?

כאן נשתמש ברעיון שכבר ראינו בפרק 4 (בסעיף על אלגוריתם גרף הקנאה). בכל שלב, נבנה את **גרף הקנאה** של החלוקה הנוכחית – גרף מכוון שבו הקודקודים הם השחקנים, וכל שחקן מצביע לשחקנים שהוא מקנא בהם. אם יש בו מעגלים – נסיר אותם ע"י החלפת סלים במעגל. לאחר שהוסרו המעגלים, בהכרח יש שחקן שאינו מקנא באף אחד אחר, ושחקן שאף אחד אחר אינו מקנא בו:

- בבעיית חלוקת חפצים, ניתן את החפץ הבא לשחקן שאף אחד אחר אינו מקנא בו;
- בבעיית חלוקת מטלות, ניתן את החפץ הבא לשחקן שאינו מקנא באף אחד אחר.

נסכם את האלגוריתם:

אלגוריתם 38.5: אלגוריתם ברמן-קרישנמורתי לבעייה מסודרת

1. אתחול:
* אתחל את החלוקה הנוכחית לחלוקה ריקה.
* סדר את החפצים בסדר יורד של הערך המוחלט שלהם.

2. חשב את גרף־הקנאה של החלוקה הנוכחית. אם יש מעגלים מכוונים, הסר אותם.

3. קח את החפץ הבא בסדרה:
* בבעיית חלוקת חפצים, תן אותו לשחקן שאף שחקן אחר אינו מקנא בו.
* בבעיית חלוקת מטלות, תן אותו לשחקן שאינו מקנא באף שחקן אחר.

4. אם נשארו חפצים – חזור לצעד 2.

משפט 39.5

בבעיית חלוקת מטלות בין n שחקנים, אלגוריתם 5.38 נותן לכל שחקן סל־מטלות שערכו לפחות $\frac{4}{3} - \frac{1}{3n}$ מערך־המקסימין שלו.

שימו לב שגורם־הקירוב של אלגוריתם ברמן-קרישנמורתי זהה לזה של האלגוריתם החמדני (משפט 5.26). הגורם הזה הוא הדוק אפילו כשלשחקנים יש הערכות זהות – ראו בחנו את עצמכם 5.27. גם הוכחת המשפט דומה להוכחה של משפט 5.26. כמו שם, גם כאן נתייחס לשם פשטות ל**עלויות** של המטלות שהן מספרים חיוביים (עלות = מינוס ערך).

הוכחת משפט 5.39: נתמקד בשחקן מסויים, וננרמל את העלויות שלו על־ידי חלוקתן בערך המקסימין שלו. אחרי הנירמול, ערך המקסימין של השחקן שווה 1. לכן סכום העלויות של כל המטלות קטן או שווה n, והעלות של כל מטלה לאחר הנירמול קטנה או שווה 1.

נחלק את המטלות ל**גדולות** (שעלותן עבור השחקן גדולה מ־ $\frac{1}{3}$) ו**קטנות** (שעלותן עבור השחקן לכל היותר $\frac{1}{3}$). הקבוצות הללו עשויות להיות שונות לשחקנים שונים, אבל זה לא משפיע על ההוכחה. כל סל בחלוקת־המקסימין של השחקן מכיל לכל היותר שתי מטלות גדולות, ובסך־הכל יש לכל היותר 2n מטלות גדולות.

אלגוריתם ברמן-קרישנמורתי פועל על בעייה מסודרת. הוא מחלק קודם את כל המטלות הגדולות, ואז את כל המטלות הקטנות. נוכיח שתי טענות־עזר: טענה א מתייחסת לחלוקת המטלות הגדולות, וטענה ב מתייחסת לחלוקת המטלות הקטנות.

טענה א: לאחר שהאלגוריתם סיים לחלק את המטלות שהן גדולות בעיני השחקן, סכום העלויות שבידי השחקן הוא לכל היותר 1.

הוכחה: אם השחקן שלנו קיבל מטלה גדולה אחת, אז כמובן העלות שלו היא לכל היותר 1.

נניח שהשחקן שלנו קיבל שתי מטלות גדולות. כשהוא מקבל את המטלה הגדולה השניה, הוא אינו מקנא באף שחקן אחר; לכן, בשלב זה, לכל שחקן אחר יש לפחות מטלה גדולה אחת, ולכן מספר המטלות הגדולות גדול מ־n. נסמן מספר זה ב־ n+t, עבור מספר שלם כלשהו t בין 1 ל־n. לכן בחלוקת המקסימין של השחקן ישנם t סלים עם שתי מטלות גדולות, ועוד n–t סלים עם מטלה גדולה אחת. נקרא לשתי מטלות גדולות

תואמות אם סכום עלויותיהן בעיני השחקן הוא לכל היותר 1. כמו בהוכחת משפט 5.26, יש לפחות t זוגות של מטלות תואמות. כל מטלה n–t+k תואמת למטלה n+t–k+1, לכל k בין 1 ל־t.

המטלה הראשונה שקיבל השחקן היתה מטלה כלשהי בין 1 ל־n; נסמן את מספרה ב־L. כשהשחקן מקבל את המטלה הגדולה השניה, הוא אינו מקנא באף שחקן אחר; לכן, בשלב זה, לשחקנים שקיבלו מטלות שמספרן בין L+1 לבין n, יש לפחות שתי מטלות גדולות. לכן מספר המטלות הגדולות הוא לפחות 2n–L+1. לפי ההגדרה, מספר המטלות הגדולות הוא n+t , ומכאן: L ≥ n–t+1. נניח ש: L=n–t+k, עבור מספר שלם כלשהו $k \geq 1$. השחקן שלנו יקבל מטלה נוספת, רק אחרי שכל השחקנים שהוא מקנא בהם (שקיבלו מטלות בין L+1 לבין n) יקבלו מטלה נוספת. לכן המטלה השניה שיקבל תהיה מטלה שמספרה לפחות n+t–k+1. כאמור למעלה, מטלה n–t+k תואמת למטלה n+t–k+1, ולכן סכום העלויות של השחקן קטן או שווה 1.

לסיום נוכיח, שהשחקן לעולם לא יקבל שלוש מטלות גדולות. סכום העלויות של כל שתי מטלות גדולות, בעיני השחקן, גדול מ־2/3. כל עוד לא סיימנו לחלק את n+t המטלות הגדולות, יש לפחות n–t+1 שחקנים שיש להם רק מטלה גדולה אחת, ולפחות אחת מהמטלות האלו היא מקבוצת המטלות ה"תואמות", ולכן העלות שלה קטנה מ־2/3. לכן, כל עוד לשחקן שלנו יש שתי מטלות גדולות, הוא מקנא בשחקן אחד לפחות, ולכן האלגוריתם לא ייתן לו מטלה גדולה שלישית.

טענה ב: כאשר האלגוריתם נותן לשחקן מטלה שהיא קטנה בעיניו, סכום העלויות החדש של השחקן (כולל המטלה שקיבל) קטן או שווה $\frac{4}{3} - \frac{1}{3n}$.

הוכחה: נסמן את עלות המטלה בעיני השחקן ב־x. סכום העלויות של המטלות שחולקו עד כה, בעיני השחקן, הוא לכל היותר n–x. השחקן שקיבל את המטלה אינו מקנא, ולכן סכום העלויות שלו, בעיני עצמו, הוא לכל היותר סכום העלויות של כל שחקן אחר. לכן, לפי כלל שובך־היונים, סכום העלויות של השחקן המקבל את המטלה הוא לכל היותר

$\frac{(n-x)}{n} = 1 - \frac{x}{n}$ לפני שקיבל את המטלה, ולכן לכל היותר $1 - \frac{x}{n} + x = 1 + x \cdot \left(1 - \frac{1}{n}\right)$

לאחר שקיבל את המטלה. כיוון שהמטלה קטנה, $x \leq 1/3$. לכן סכום העלויות החדש של השחקן הוא לכל היותר

$$1 + \frac{1 - \frac{1}{n}}{3} = \frac{4}{3} - \frac{1}{3n}.$$

בכך סיימנו את הוכחת טענה ב.

לסיום, נניח שהאלגוריתם, לאחר שנתן לשחקן שלנו מטלה כלשהי, מסיר מעגלי־קנאה על-ידי החלפת סלים במעגל. תהליך החלפת הסלים תמיד משפר את מצבו של כל אחד מהשחקנים המשתתפים במעגל, ואינו משנה את מצבם של השחקנים האחרים. לכן,

טענות א ו־ב נכונות גם לאחר הסרת מעגלי־קנאה. לכן, גם בסוף האלגוריתם, סכום

העלויות הכולל של השחקן הוא לכל היותר $\frac{4}{3} - \frac{1}{3n}$.

מש"ל.

בחנו את עצמכם 40.4

נ

תונים שלושה שחקנים ושמונה מטלות, עם העלויות הבאות:

	s,	t,	u,	v,	w,	x,	y,	z
A:	**39**,	33,	20,	11,	7,	5,	4,	**4**
B:	39,	**33**,	20,	14,	**7**,	5,	4,	4
C:	39,	33,	**20**,	**14**,	7,	**5**,	**4**,	4

א. הפעילו את אלגוריתם ברמן-קרישנמורתי, והראו שהחלוקה המתקבלת היא החלוקה המודגשת, שבה העלות של שחקן A הוא 43.

ב. הראו, שאם יש שלושה שחקנים עם הערכה זהה לזו של שחקן A, אז האלגוריתם החמדני מחזיר חלוקה שבה העלות הקטנה ביותר היא 42.

הוכחנו את יחס־הקירוב של אלגוריתם 5.38 על בעייה מסודרת. כעת, בהינתן בעייה כללית של חלוקת מטלות, אפשר להשתמש באלגוריתם 5.36, כאשר בשלב ג, האלגוריתם שיופעל יהיה אלגוריתם 5.38. לפי משפט 5.37, הסל שיקבל כל שחקן בסוף האלגוריתם יהיה טוב בעיניו לפחות באותה מידה, ולכן גורם הקירוב של משפט 5.39 נכון גם לבעייה כללית של חלוקת מטלות.

מה קורה בבעייה של חלוקת **חפצים**? ברמן וקרישנמורתי הוכיחו, שיחס־הקירוב של האלגוריתם שלהם בבעיית חלוקת חפצים הוא לפחות $2n/(3n-1)$. אבל זה פחות טוב מיחס־הקירוב של האלגוריתם החמדני לפי משפט 5.28, שהוא $(4n-2)/(3n-1)$. ניתן לשער, שיחס־הקירוב האמיתי של אלגוריתם ברמן-קרישנמורתי לחלוקת חפצים הוא $(4n-2)/(3n-1)$, אבל העניין עדיין פתוח.

5.7.3 אלגוריתמים נוספים **

ברמן וקרישנמורתי לקחו אלגוריתם קיים לפתרון בעיית תיזמון – אלגוריתם LPT, והתאימו אותו לבעיית חלוקה הוגנת. שין הואנג ופיניאן לו[25] השתמשו ברעיון דומה, רק שהם לקחו אלגוריתם אחר לבעיות תיזמון – אלגוריתם MultiFit.[11] יחס־הקירוב של אלגוריתם הואנג–לו בחלוקת מטלות הוא לכל היותר 11/9; לא ידוע מה יחס־הקירוב שלו בחלוקת חפצים.

ישנם עוד אלגוריתמים רבים המשיגים קירובים שונים להוגנות־מקסימין.[26] עדיין לא ידוע מה יחס־הקירוב הטוב ביותר שאפשר להשיג. בחלוקת חפצים, ידוע שאפשר להשיג יחס

** ** תת-סעיף זה מומלץ למתקדמים בלבד.

3/4,[27] ואי־אפשר להשיג יחס טוב יותר מ־ 39/40 (משפט 5.35). בחלוקת מטלות, ידוע שאפשר להשיג יחס 11/9,[23] ואי־אפשר להשיג יחס טוב יותר מ־44/43.[19]

שאלה פתוחה 5.41

מהו יחס־הקירוב הטוב ביותר האפשרי להוגנות־מקסימין בחלוקת **חפצים**? ובחלוקת **מטלות**?

ישנו סוג אחר של קירוב להוגנות־מקסימין. נניח שיש n שחקנים, ומבקשים מכל אחד מהם למצוא חלוקת־מקסימין בין **n+1** שחקנים עם הערכות זהות לשלו (במקום n). הערך המתקבל הוא בדרך־כלל, קטן יותר מערך־המקסימין המחושב ע"פ n שחקנים זהים. לכן ייתכן, תיאורטית, שאפשר להשיג חלוקה הנותנת לכל שחקן לפחות ערך זה. המצב הפוך כשמחלקים מטלות: הערך קטן יותר כשמחשבים את חלוקת־המקסימין בין **n–1** שחקנים עם הערכות זהות (יש פחות שחקנים, כל שחקן צריך לבצע יותר מטלות, ולכן הערך – השלילי – של כל שחקן הוא נמוך יותר).

שאלה פתוחה 5.42

נתונים n שחקנים עם הערכות חיבוריות. האם תמיד קיימת –

א. חלוקת **חפצים** שבה הערך של כל שחקן הוא לפחות ערך המקסימין (n+1) שלו?

ב. חלוקת **מטלות** שבה הערך של כל שחקן הוא לפחות ערך המקסימין (n–1) שלו?

עבור חלוקת חפצים, ידוע אלגוריתם הנותן לכל שחקן ערך־מקסימין (2n–2),[28] ואלגוריתם משופר הנותן לכל שחקן ערך־מקסימין (3n/2).[29] עבור חלוקת מטלות, ידוע אלגוריתם הנותן לכל שחקן ערך־מקסימין (2n/3), ואלגוריתם משופר הנותן לכל שחקן ערך־מקסימין (3n/4).[22]

הבעיה בכל האלגוריתמים הללו היא, שקשה להסביר אותם לאנשים שאינם מדעני־מחשב. כדי ששחקן יוכל להבין את הקירוב שמבטיחים לו, הוא צריך לחשב את ערך־המקסימין שלו, ולשם כך הוא צריך לפתור בעיית חלוקה שהיא קשה אפילו למחשב (בחנו את עצמכם 5.19 סעיף א), קל וחומר לבן־אדם.

5.7.4 שאלות נוספות לתירגול וחזרה

1. הוגנות־מקסימין לעומת הוגנות עד כדי חפץ אחד

א. תנו דוגמה לחלוקה מקסימין־הוגנת שאינה EF1.

ב. תנו דוגמה לחלוקה EFX שאינה מקסימין־הוגנת.

2. אלגוריתם גרף־הקנאה על בעיה לא מסודרת

בפרק 4 למדנו, שאלגוריתם גרף־הקנאה על בעיה מסודרת מחזיר תמיד חלוקה EFX. הראו (על־ידי דוגמה) שאלגוריתם 5.36 אינו שומר על תכונה זו. כלומר: אם לוקחים בעיה לא־מסודרת, מסדרים אותה, מריצים את אלגוריתם גרף־הקנאה בשלב ג, ואז מריצים את שלב ד של אלגוריתם 5.36, התוצאה אינה בהכרח EFX.

לחשוב על איור מסכם לפרק

מושגים בפרק לפי סדר א"ב

- **בעיית חלוקת המספרים – Partition Problem – Number Partitioning Problem**: בעיה חישובית, שהקלט שלה הוא רשימת מספרים, והפלט הוא "כן" אם ורק אם קיימת חלוקה של המספרים לשתי תת־קבוצות עם סכום זהה.
- **בעיית תיזמון – Scheduling problem**: בעייה חישובית, שהקלט שלה הוא רשימת תהליכים חישוביים ואוסף של מעבדים, והפלט שלה הוא חלוקה של התהליכים בין המעבדים שמטרתה לבצע את כל התהליכים בזמן הקצר ביותר האפשרי.
- **גיזום – pruning**: ביטול החיפוש בתת־עץ מסויים של עץ הפתרונות, לאחר שנוכחים לדעת שתת־העץ הזה אינו חיוני למציאת פתרון מיטבי.
- **חיפוש במרחב המצבים – State-space search**: שיטה לפתרון בעיות־מיטוב, שבה עוברים על מרחב הפתרונות האפשריים, תוך *גיזום* חלקים ממנו אשר בוודאות אינם כוללים פתרון מיטבי.
- **חלוקה אופטימלית־נאש – Nash-optimal allocation** (בהקשר של חפצים בדידים): חלוקה הממקסמת את מספר השחקנים המקבלים ערך חיובי, ובכפוף לזה, את המכפלה של ערכי השחקנים המקבלים ערך חיובי.
- **חסם אופטימי**: בהינתן פתרון חלקי X לבעייה כלשהי (למשל: חלוקה של חלק מהחפצים), חסם אופטימי הוא ערך כלשהו V, כך שלא ניתן להשיג ערך טוב יותר מ־V כשמתחילים מ־X.
- **חסם פסימי**: בהינתן קבוצה חלקית כלשהי של פתרונות לבעייה (למשל: אוסף חלוקות של חפצים), חסם פסימי הוא ערך הפתרון הטוב ביותר שנמצא עד כה.
- **יחס־הקירוב – Approximation ratio**: היחס בין ערך הפתרון המיטבי, לבין ערך הפתרון שהוחזר על־ידי אלגוריתם כלשהו.
- **סיעוף וחסימה – Branch and bound**: סוג של אלגוריתם *חיפוש במרחב המצבים*, המבצע *גיזום* לכל ענף עם *חסם אופטימי* שאינו טוב יותר מהפתרון הטוב ביותר שנתגלה עד כה.
- **קירוב גורם־קבוע – Constant-factor approximation**: אלגוריתם קירוב המחזיר פתרון, שערכו לפחות r כפול הערך המקסימלי (לבעיית מקסימום), או לכל היותר r כפול הערך המינימלי (לבעיית מינימום), כאשר r הוא מספר קבוע שאינו תלוי בקלט.
- **שיטת קירוב – Approximation Scheme**: שיטה ליצירת אלגוריתמי־קירוב עם יחס־קירוב טוב כרצוננו.
- **שיטת קירוב בזמן פולינומיאלי – Polynomial-Time Approximation Scheme – PTAS**: שיטת קירוב המייצרת אלגוריתם קירוב עם זמן ריצה פולינומיאלי בגודל הקלט, לכל יחס־קירוב בנפרד.
- **שיטת קירוב בזמן פולינומיאלי מלא – Fully Polynomial-Time Approximation Scheme – FPTAS**: שיטת קירוב המייצרת אלגוריתם קירוב עם זמן ריצה פולינומיאלי בגודל הקלט וביחס־הקירוב.
- **תיכנות ליניארי בשלמים – Integer Linear Programming**: תוכנית ליניארית עם אילוץ הקובע, שכל המשתנים צריכים להיות מספרים שלמים.

ביבליוגרפיה לפרק

[1] Richard Karp (1972): "Reducibility Among Combinatorial Problems". In R. E. Miller; J. W. Thatcher; J.D. Bohlinger (editors): “Complexity of Computer Computations”. New York: Plenum. Pages 85–103.

[2] Haris Aziz, Serge Gaspers, Simon Mackenzie, Toby Walsh (2015): “Fair assignment of indivisible objects under ordinal preferences”. Artificial Intelligence 227: 71–92.

[3] Sylvain Bouveret, Ulle Endriss, Jérôme Lang (2010): "Fair division under ordinal preferences: Computing envy-free allocations of indivisible goods." Proceedings of ECAI’10.

[4] Haris Aziz, Xin Huang, Nicholas Mattei, Erel Segal-Halevi (2022): “Computing Welfare-Maximizing Fair Allocations of Indivisible Goods”. arXiv preprint 2012.03979.

[5] Ioannis Caragiannis, David Kurokawa, Hervé Moulin, Ariel D. Procaccia, Nisarg Shah, Junxing Wang (2019): "The Unreasonable Fairness of Maximum Nash Welfare". ACM Transactions on Economics and Computation. 7.3: 12:1–12:32.

[6] Mithun Chakraborty, Erel Segal-Halevi, Warut Suksompong (2022): "Weighted Fairness Notions for Indivisible Items Revisited". Proceedings of AAAI 2022.

[7] Richard Cole and Vasilis Gkatzelis (2015): "Approximating the Nash Social Welfare with Indivisible Items". Proceedings of STOC 2015, 371–380.
Nima Anari, Shayan Oveis Gharan, Amin Saberi, Mohit Singh (2017): "Nash Social Welfare, Matrix Permanent, and Stable Polynomials". Proceedings of ITCS 2017, 36:1–36:12.
Richard Cole, Nikhil R. Devanur, Vasilis Gkatzelis, Kamal Jain, Tung Mai,; Vijay V. Vazirani, Sadra Yazdanbod (2017): "Convex Program Duality, Fisher Markets, and Nash Social Welfare". Proceedings of EC 2017.
Wikipedia page: "Efficient approximately-fair item allocation".

[8] Georgios Amanatidis, Georgios Birmpas, Aris Filos-Ratsikas, Alexandros Hollender, Alexandros A. Voudouris (2021): "Maximum Nash welfare and other stories about EFX". Theoretical Computer Science, 863:69–85.
Daniel Halpern, Ariel D. Procaccia, Alexandros Psomas, Nisarg Shah (2020): "Fair Division with Binary Valuations: One Rule to Rule Them All". Web and Internet Economics 12495: 370–383.

[9] Ronald L. Graham (1969): "Bounds on multiprocessing timing anomalies". SIAM journal on Applied Mathematics 17.2, 416-429.

[10] בהשראת הוכחה של שין שייאו:
Xin Xiao (2017): "A Direct Proof of the 4/3 Bound of LPT Scheduling Rule". Proceedings of FMSMT 2017, Atlantis Press, 486–489.
Wikipedia page: "Longest-processing-time-first scheduling".

[11] Bryan L. Deuermeyer, Donald K. Friesen, Michael A. Langston (1982): "Scheduling to Maximize the Minimum Processor Finish Time in a Multiprocessor System". SIAM Journal on Algebraic and Discrete Methods, 3.2, 190–196.
János Csirik, Hans Kellerer, Gerhard Woeginger (1992): "The exact LPT-bound for maximizing the minimum completion time". Operations Research Letters. 11.5: 281–287.
Bang Ye Wu (2005): "An analysis of the LPT algorithm for the max–min and the min–ratio partition problems". Theoretical Computer Science. 349.3: 407–419.

[12] Narendra Karmarkar and Richard M. Karp (1982): "The differencing method of set partitioning", Tech report UCB/CSD 82/113, Computer science division, University of California, Berkeley, 1982.

[13] E. G. Coffman Jr., M. R. Garey, D. S. Johnson (1978): "An Application of Bin-Packing to Multiprocessor Scheduling". SIAM Journal on Computing. 7.1: 1–17.
Wikipedia page: "Multifit algorithm".

[14] Gerhard J. Woeginger (2000): "When does a dynamic programming formulation guarantee the existence of a fully polynomial time approximation scheme (FPTAS)?". INFORMS Journal on Computing 12.1, 57-74.
Wikipedia page: "FPTAS".

[15] Noga Alon, Yossi Azar, Gerhard J. Woeginger, Tal Yadid (1998): "Approximation schemes for scheduling on parallel machines". Journal of Scheduling 1.1: 55-66.

[16] Richard E. Korf (1995): "From approximate to optimal solutions: A case study of number partitioning". In Proceedings of IJCAI 1995, pages 266–272.

[17] Ethan L. Schreiber, Richard E. Korf, Michael D. Moffitt (2018): "Optimal multi-way number partitioning." Journal of the ACM (JACM) 65.4, 1-61.

[18] Sylvain Bouveret, Michel Lemaître (2009): "Computing leximin-optimal solutions in constraint networks". Artificial Intelligence. 173.2: 343–364.

[19] Eric Budish (2011): "The combinatorial assignment problem: Approximate competitive equilibrium from equal incomes." Journal of Political Economy 119.6: 1061-1103.

[20] Ariel D. Procaccia, Junxing Wang (2014): "Fair enough: guaranteeing approximate maximin shares". Proceedings of EC 2014:675–692.

[21] Uriel Feige, Ariel Sapir and Laliv Tauber (2021): "A tight negative example for MMS fair allocations." arXiv preprint 2104.04977.

[22] Siddharth Barman, Sanath Kumar Krishnamurthy (2020): "Approximation Algorithms for Maximin Fair Division". ACM Transactions on Economics and Computation. 8.1: 5:1–5:28.

[23] Sylvain Bouveret and Michel Lemaitre (2016): "Characterizing conflicts in fair division of indivisible goods using a scale of criteria". Autonomous Agents and Multi-Agent Systems 30.2, 259-290.

[24] Hadi Hosseini, Andrew Searns, Erel Segal-Halevi (2022): "Ordinal Maximin Share Approximation for Chores". Proceedings of AAMAS 2022. arXiv preprint 2201.07424.

[25] Xin Huang, Pinyan Lu (2021): "An Algorithmic Framework for Approximating Maximin Share Allocation of Chores". Proceedings of EC 2021, 630-631.

[26] Georgios Amanatidis, Evangelos Markakis, Afshin Nikzad,; Amin Saberi (2017): "Approximation Algorithms for Computing Maximin Share Allocations". ACM Transactions on Algorithms. 13.4: 1–28.

Mohammad Ghodsi, Mohammadtaghi Hajiaghayi, Masoud Seddighin, Saeed Seddighin, Hadi Yami (2018): "Fair Allocation of Indivisible Goods: Improvements and Generalizations". Proceedings of EC 2018: 539–556.

Jugal Garg, Peter McGlaughlin, Setareh Taki (2019): "Approximating Maximin Share Allocations". Proceedings of the 2nd Symposium on Simplicity in Algorithms (SOSA 2019).

[27] Jugal Garg, Setareh Taki (2020): "An Improved Approximation Algorithm for Maximin Shares". Proceedings of EC 2020: 379–380.

[28] Elad Aigner-Horev, Erel Segal-Halevi (2022): "Envy-free matchings in bipartite graphs and their applications to fair division". Information Sciences. 587: 164–187

[29] Hadi Hosseini, Andrew Searns (2021): "Guaranteeing Maximin Shares: Some Agents Left Behind". Proceedings of IJCAI 2021.

Hadi Hosseini, Andrew Searns and Erel Segal-Halevi (2022): "Ordinal Maximin Share Approximation for Goods." JAIR. arXiv preprint 2109.01925.