אלגו מתקדם - 67824

11 ביוני 2012

מרצה: יאיר בר־טל
בודק: אורן בקר

איני לוקחת אחריות על מה שכתוב כאן, so tread lightly
אין המרצה קשור לסיכום זה בשום דרך.
הערות יתקבלו בברכה - *noga.rotman@gmail.com*. אהבתם? יש עוד!
$www.cs.huji.ac.il/\sim nogar02$

תוכן עניינים

רשימת אלגוריתמים

0.1 הקדמה

5/3/2012

המחקר במדעי המחשב עוסק בשני נושאים עיקריים:

- סיבוכיות
- אלגוריתמים

אנו נעסוק באלגוריתמים מתקדמים.

הנושאים שנעסוק בהם הם:

- אלגוריתמי אפרוקסימציה (Approximation Algorithms).
- אלגוריתמים מקוונים (Online Algorithms) - בעיות בהן המידע לא נתון מראש, אלא בשלבים (לא קשור לאינטרנט).
- נושאים נוספים:
  - שיטות מטריות.

הסיבה לכך היא שכמות גדולה מהבעיות הבסיסיות בעולם הם $NP-complete$ - בעיות כמו תזמון, מציאת מסלול וכו'. אי אפשר לפתור אותן בזמן פולינומיאלי. צריך להתמודד איתן בצורה אחרת.

ישנן כמה דרכים לנסות להתמודד איתן. הגישה הכללית בתיאוריה של מדעי המחשב היא לנסות להגיד דברים יותר ריגורוזיים על מה שאנחנו עושים. כאן יש פער בין המעשיות לבין תיאוריה:

1. יוריסטיקות - שיטות מעשיות. כלומר, לא מנסים לנתח לעומק (רק ניתוח בסיסי) אבל אי אפשר לקבל הבטחה לגבי האלגוריתמים המתקבלים (לא מסמנים בזמן סביר במקרים מסויימים, לא תמיד מסיימים), אבל ברב המקרים מסתיימים בצורה טובה.
2. ניתוח המקרה הממוצע (לאו דווקא באופן המוחלט)\מודלים הסתברותיים על הקלט. כאן יש חיסרון - במידה והמציאות היא לא כמו המודל, הניתוח אינו רלוונטי. בגישה זו פחות ניגע בקורס הזה.
3. ניתוח המקרה הגרוע - ברגע שיש הבטחה מהסוג הזה אז ההבטחה נכונה לכל קלט שנקבל. עם זאת, המציאות יכולה להיות הרבה יותר טובה מהמקרה הזה, ולכן בשאר המקרים אולי אנחנו נוקטים בדרך לא נכונה. עם זאת, בגישה הזו אפשר להגיד דברים מאוד מעניינים.

בגישה השניה נכנס הנושא של אלגוריתמי אפרוקסימציה - ננסה למצוא קירוב עד כדי קבוע מסויים. זוהי בעיית $offline$ - כל הקלט נתון מראש (למשל, תזמון של עבודות במעבד).

אלגוריתמים מקוונים לעומת זאת דנים בבעיות בהן הקלט לא ניתן כולו בבת אחת, אלא תוך כדי עבודה. עוד דרך להתמודד עם בעיות שכאלו - Competitive Analysis.

0.1.1 ספרים

Approximation Algorithms - Vijay Vazirani.

The Design of Approximation Algorithms - D. Williamson, D. Shmous.

0.1.2 הרכב הציון

סיכומי שיעור יהיו חלק מהציון (לא ברור מה חלקו בשלב זה).
רב הציון יהיה מבוסס על הגשת תרגילים, בערך תרגיל כל שבועיים. לא יהיה מבחן בית.
יהיה ראיון בסוף הקורס על התרגילים - $20\% - 25\%$.

1 אלגוריתמי אפרוקסימציה

1.1 הגדרות

נזכר במושגים הבסיסיים:

הגדרה 1.1 מחלקה P היא מחלקת הבעיות שניתן לפתור בזמן פולינומי.

הגדרה 1.2 מחלקת NP היא מחלק הבעיות שניתן לפתור בזמן פולינומי עם עד, או שניתן לבדוק אותן בזמן פולינומי (ההגדרות שקולות).

הגדרה 1.3 $NP-hard$ - בעיה שאפשר לעשות עליה רדוקציה פולינומיאלית מכל בעיה אחרת ב-NP.

הגדרה 1.4 $NP-complete$ - אם הבעיה היא $NP-hard$ וגם ב-NP.

1.1.1 אלגוריתמי אופטימיזציה

סוג הבעיות שאנו מתעסקים בהן הוא בעיות אופטימיזציה (בהן הקלט נתון מראש):
נתון קלט I. לכל פתרון אפשרי מוגדרת:

- בבעיות מינימיזציה - עלות (מחיר).
- בבעיות מקסימיזציה - תועלת (רווח).

בהנתן אלגוריתם A, נסמן ב-$A(I)$ את המחיר/רווח שלו.
האלגוריתמים האופטימלי OPT עם מחיר/רווח מסומן ב-$OPT(I)$:

- במינימיזציה - יש לו מחיר מינימלי.
- במקסימיזציה - יש לו רווח מקסימלי.

דוגמאות בסיסיות לבעיות מינימיזציה:

- בעיית תזמון - Scheduling (בדר"כ חושבים עליה כבעיית מינימיזציה).
- TSP - נתון גרף, ונרצה למצוא מסלול שעובר דרך כל הנקודות בעל עלות מינימלי, שחוזר למקום שבו התחלנו בו (מעגל).

דוגמאות לבעיות מקסימיזציה:

- Max SAT.
- Max Cut.
- Max Click - מציאת הקליק הגדול ביותר בגרף.

מהי המשמעות לכך שבעיית אופטימיזציה היא NP-קשה (שכן בהגדרה ההתייחסות היא לבעיות הכרעה)?
הופכים את בעית האופטימיזציה לבעיית הכרעה:
בהנתן k, האם ערך הפתרון האופטימלי $\geq k$ למינימיזציה, או $\leq k$ למקסימיזציה.
מאחר והבעיות הללו הן קשות, אז גם בעיות האופטימיזציה הן קשות.

1.1.2 אלגוריתמי קירוב

הגדרה 1.5 אם מדובר בבעיית מינימיזציה, אומרים שלאלגוריתם A יש יחס קירוב α (עבור $\alpha > 1$) אם A רץ בזמן פולינומיאלי, ולכל קלט I מתקיים:

$$A(I) \leq \alpha \cdot OPT(I)$$

עבור בעיית מקסימיזציה, אומרים שלאלגוריתם A יש יחס קירוב α (עבור $\alpha < 1$) אם A רץ בזמן פולינומיאלי, ולכל קלט I מתקיים:

$$A(I) \geq \alpha \cdot OPT(I)$$

הגדרה 1.6 יחס הקירוב של אלגוריתם A:
בבעיית מינימיזציה - ה-α המינימלי שעבורו לאלגוריתם A יש יחס קירוב α.

הגדרה 1.7 יחס הקירוב של הבעיה עצמה:
יחס הקירוב הטוב ביותר (מינימיזציה - מינימלי, מקסימיזציה - מקסימלי) שאלגוריתם (פולינומיאלי) כלשהוא יכול להשיג.

נשים לב כי α יכול להיות פונקציה של גודל הקלט.

נתחיל עם דוגמא שאנחנו כבר למעשה מכירים:

בעיית Vertex Cover

נתון גרף $G := (V, E)$.
אומרים שקבוצה $C \subseteq V$ היא כיסוי של G, אם $\forall e \in E$ קיים קודקוד בכיסוי $v \in C$ כך ש-$v \in e$.
רוצים למצוא כיסוי C כך ש-$|C|$ מינימלי.

Weighted VC

נתונה בנוסף פונקציית משקל $w : V \to \mathbb{R}^+$.
רוצים למצוא כיסוי C כך ש-$\sum_{v \in C} w(v)$ מינימלי.

זוהי בעיה NP-קשה. כיצד אפשר לעשות את הניתוח כאשר לא נוכל להציג את הפתרון האופטימלי?
כאשר מדובר בבעיית מינימיזציה, מציגים חסם תחתון למחיר האופטימלי (בבעיית מקסימיזציה - חסם עליון לרווח האופטימלי). בהנתן קלט I, נסמן חסם תחתון זה ב-$LB(I)$ (או $UB(I)$ לבעיית מקסימיזציה). במקום לחסום את היחס בין המחיר של האלגוריתם לאופטימלי, נחסום את היחס בין המחיר של האלגוריתם לחסם התחתון:

$$\forall I \; OPT(I) \leq LB(I) \Rightarrow \frac{A(I)}{OPT(I)} \leq \frac{A(I)}{LB(I)} \leq \alpha$$

הגדרה 1.8 זיווג בגרף $G := (V, E)$ הוא תת קבוצה $M \subseteq E$ שזרות זו לזו, כלומר שאין ב־M זוג קשתות בעלות קודקוד משותף.

הגדרה 1.9 זיווג מקסימלי (maximal matching)[1]הוא זיווג $M \subseteq E$ שלא ניתן להוסיף לו קשת נוספת.

אלגוריתם 1 אלגוריתם ל־VC

מצא זיווג מקסימלי M, והכנס לכיסוי C את כל קודקודי הקשתות ב־M.

משפט 1.10 לאלגוריתם יש יחס קירוב 2.

הוכחה: נגדיר לכל קלט I ־ $M(I)$ (אפשר לעשות בצורה גרידית, אבל גם את זה אפשר בצורות שונות).

טענה 1.11

$$\forall I \; OPT(I) \geq |M(I)| = LB(I)$$

הוכחה: נתבונן בזיווג המקסימלי. לכל קשת ב־$M(I)$ חייב להיות לפחות קודקוד אחד בכיסוי (כי הקשתות זרות). ■

נוכיח את המשפט:

- נוכיח כי קיבלנו כיסוי חוקי:
  נניח בשלילה שהכיסוי לא חוקי. אזי קיימת קשת כך שאף אחד מקודקודיה אינו בכיסוי C (ולכן לא ב־M). אזי ניתן להוסיף אותה ל־M, בסתירה למקסימליות M.
- ננתח את העלות:

$$A(I) = 2|M(I)| \leq 2 \cdot OPT(I)$$

■

לכל בעיית אפרוקסימציה יש תמיד שאלות טבעיות שעולות:

1. האם יחס הקירוב שהוכחנו הוא הטוב ביותר עבור האלגוריתם? במקרה שלנו נראה שכן, כלומר שלא ניתן לשפר את יחס הקירוב.
2. "פער השלמות"[2]: האם יש אלגוריתם אחר B שיכול להשיג יחס קירוב טוב יותר ביחס לאותו חסם תחתון? במקרה שלנו נראה שאין אלגוריתם יותר טוב לחסם התחתון הזה.

[1] בניגוד ל־maximum matching, בו מספר הצלעות בזיווג הוא מקסימלי. אלו הגדרות שלא חופפות ־ למשל ניתן להסתכל בגרף בצורת האות Z. קל הרבה יותר למצוא maximal (בצורה גרידית ־ מוסיפים קשתות כל עוד אפשר), והוא מספיק להמשך הדיון לכן אנו דנים בו.

[2] נגיע למושג הזה בהמשך, כרגע לא נדון בו.

3. האם יחס הקירוב לבעיה הוא מה שהוכחנו (במקרה שלנו 2)? האם ניתן לשפר את החסם התחתון?

נדון בשאלות, כאשר נתייחס לחסם התחתון שמצאנו 2:
על השאלה הראשונה נוכל להגיד שהתשובה היא כן, במידה ונוכל למצוא קלט I עבורו $A(I) \geq 2 \cdot OPT(I)$. כמובן שלא תמיד ניתן למצוא כזה, וניתן לנסח את השאלה בצורה הבאה:
האם לכל $\varepsilon > 0$ קיים קלט I כך ש- $A(I) > (2-\varepsilon) \cdot OPT(I)$?
לגבי השאלה השניה:
מספיק שנראה כי קיים I כך ש:

$$\frac{OPT(I)}{LB(I)} \geq 2$$

וזאת כי לכל B, $\frac{B(I)}{LB(I)} \geq \frac{OPT(I)}{LB(I)}$.
למעשה, נספיק שנראה כי לכל $\varepsilon > 0$ קיים קלט I כך ש:

$$\frac{OPT(I)}{LB(I)} \geq 2 - \varepsilon$$

נענה על השאלות:

1. יחס הקירוב שהוכחנו הוא אכן הטוב ביותר עבור האלגוריתם - לדוגמא, בהנתן קלט בגודל n בו כל הצלעות זרות:

$$OPT(I) = n,\ A(I) = 2n$$
$$\Rightarrow \frac{A(I)}{OPT(I)} = 2$$

כלומר מצאנו קלט עבורו יחס הקירוב שהוכחנו מתקיים, כנדרש.
מספיק במקרה זה לקחת קשת יחידה (אנו דנים במקרה כללי כי אפשר לטעון כי אולי במקרים של גרפים מאוד גדולים קיים יחס קירוב טוב יותר).

2. $LB(I) = |M(I)|$. נרצה להראות כי:

$$\frac{OPT(I)}{LB(I)} \geq 2$$

בהנתן קליק בגודל n (אי זוגי), $OPT(I) = n-1$ (אם ניקח פחות, ישנם שני קודקודים שיש ביניהם צלע (כי זה קליק) והיא לא מכוסה). כמו כן:

$$LB(I) = \frac{n-1}{2} \Rightarrow \frac{OPT(I)}{LB(I)} = 2$$

כאן מספיק לקחת משולש.

3. אפשר לשפר "בטיפה" - ידוע יחס קירוב של $2-\frac{1}{\sqrt{\log n}}$ באלגוריתם של Karalocostas, '04.
איך אפשר להגיד שלא ניתן לשפר את החסם התחתון? זה לא נושא קל. יש תיאוריה

שלמה העוסקת בנושא זה - "קושי של קירוב". זה לא הנושא של קורס זה. חוקרים הראו (פריצת דרך בתיאוריה של מדעי המחשב לפני עשרים שנה) שיש מחלקות של בעיות שלא ניתן לקרב. לעיתים כדי להראות קושי של קירוב מספיק להראות רדוקציה, אבל גם זה לא נושא קל.
על הבעיה שלנו, בהנחה ש-$P \neq NP$, לא ניתן לקרב את הבעיה ביחס קטן מ-1.36... - תוצאה של Dinur, Safra '05.

1.2 תכנות לינארי - Linear Programming

מסתבר שישנן הרבה בעיות אופטימיזציה שאפשר לבטא אותן בעזרת פונקציית מטרה לינארית.
למשל, נעסוק בבעיה הממושקלת של VC. ניתן להציג אותה באופן הבא:
לכל קודקוד $v \in V$ בגרף ניחס מספר/אינדיקטור $\chi_v \in \{0,1\}$: $\chi_v = 1 \Leftrightarrow v \in C$.
נגדיר את פונקציית המטרה להיות:

$$min\left(\sum_{v\in V} w(v)\cdot\chi_v\right)$$

נרצה שלכל צלע (u,v) לפחות עבור אחד מהם האינדיקטור יהיה אחד. אפשר לבטא אילוץ זה בצורה לינארית:

$$\forall (u,v)\in E:\ \chi_v+\chi_u\geq 1$$

כמו כן, ישנו אילוץ נוסף:

$$\forall v\in V:\ \chi_v\in\{0,1\}$$

מערכת זו נקראת ILP.
אם נוריד את האילוץ השני, נקבל בעיה שאפשר לפתור בזמן פולינומי:
נבנה תוכנית לינארית - LP, והתוצאות שנקבל בתוכנית הקודמת יתקבלו גם פה:

$$\begin{aligned} &min\left(\sum_{v\in V} w(v)\cdot\chi_v\right)\\ &\forall (u,v)\in E:\ \chi_v+\chi_u\geq 1\\ &\forall v\in V:\ \chi_v\geq 0\\ &\forall v\in V:\ \chi_v\leq 1 \end{aligned}$$

כשנפתור את התוכנית החדשה, נקבל תוצאות שהן שבריות. זה לא עונה לנו על השאלה של הכיסוי - אלא נותן לנו מעין "כיסוי שברי". הרעיון - נשתמש ב"כיסוי השברי" אותו אנו יודעים למצוא, על מנת לפתור את הבעיה הממושקלת.
מדוע קל יותר לפתור את הבעיה השניה? בעוד שהמרחב המקורי הוא מרחב בדיד, התוכנית השניה נותנת מרחב קמור, ובמרחב שכזה ניתן לעשות אופטימיזציה בצורה קלה יותר.
כיצד נשתמש בפתרון הבעיה השניה?

$$\forall I: ILP(I) = OPT(I)$$
$$OPT(I) = ILP(I) \geq LP(I) := LB(I)$$

מדוע קיבלנו ש־$LP(I)$ הוא חסם תחתון? מהכלה! המקרים בהם χ_v הוא אפס או אחד מוכלים בבעיה השניה, ולכן כל פיתרון של ILP הוא פתרון של LP, ולכן $LP(I)$ הוא חסם תחתון של $ILP(I)$.

נאמר כמה מילים על ההגדרה הכללית של תוכנית לינארית LP:
יהיו משתנים $x_1, ..., x_n \in \mathbb{R}$, מקדמים $c_1, ..., c_n \in \mathbb{R}$, ופונקציית המטרה היא:

$$min/max \left\{ \sum_{1 \leq i \leq n} c_i x_i \right\}$$

וכמו כן יהיו מקדמים $b_1, ..., b_m \in \mathbb{R}$, $a_{ij} \in \mathbb{R},\ 1 \leq i \leq n,\ 1 \leq j \leq m$, $A = (a_{ij})$, ואילוצים:

$$\forall 1 \leq j \leq m_i \ \sum_i a_{ij} x_j \geq \backslash \leq \backslash = b_j$$

אזי את התוכנית הלינארית הנ"ל ניתן לפתור בזמן פולינומיאלי. למשל, בעזרת אלגוריתם simplex (אין עליו ניתוח רוגרוזי אבל בפועל הוא עובד טוב). יש אלגוריתמים נוספים שאפשר עליהם להוכיח שהם אכן פועלים בזמן פולינומיאלי, למשל אלגוריתם אליפסואיד.
במקרה שלנו, התוכנית הלינארית היא:

$$min \left(\sum_{v \in V} w(v) \cdot x_v \right)$$
$$\forall (u,v) \in E: \ x_v + x_u \geq 1$$
$$\forall v \in V: \ x_v \geq 0$$
$$\forall v \in V: \ x_v \leq 1$$

מכיוון ועושים מינימיזציה, אין למעשה צורך באילוץ האחרון ־ אם יש פתרון שבו $\chi_v > 1$, אז קיים פתרון בו $\chi_v \leq 1$ והוא לא פחות טוב ממנו (מבחינת המשקלות).

12/03/2012

כיצד משתמשים בזה כדי להראות את האפרוקסימציה?
ראינו כי הפתרון האופטימלי ל־LP (הפתרון האופטימלי השברי) מהווה חסם תחתון (לבעיית מינימיזציה) למחיר הפתרון האופטימלי (של ה־ILP).

אלגוריתם 2 אלגוריתם ל־WVC
נחשב את הפתרון האופטימלי ל־LP.
לכל i, אם $x_i \geq \frac{1}{2}$, קבע $\hat{x}_i = 1$, אחרת $\hat{x}_i = 0$.

משפט 1.12 לאלגוריתם יש יחס קירוב 2.

הוכחה: תחילה, נראה שקיבלנו כיסוי:
נסתכל על קשת $\{i,j\} \in E$, $x_i + x_j \geq 1$, לכן $x_i \geq \frac{1}{2}$ או $x_j \geq \frac{1}{2}$. לכן $\hat{x}_i = 1$ או $\hat{x}_j = 1$ (כלומר זהו כיסוי), ולכן $\hat{x}_i + \hat{x}_j \geq 1$.
כעת, נראה כי זהו קירוב 2:

$$ALG\left(I\right) = w\left(C\right) = \sum_{i \in V} w_i \hat{x}_i = \sum_{i:x_i \geq \frac{1}{2}} w_i \leq 2 \sum_{i:x_i \in V} w_i x_i = 2 \cdot OPT_F\left(I\right) \leq 2 \cdot OPT\left(I\right)$$

■

נחזור ונשאל את שלוש השאלות ששאלנו בשיעור הקודם:

1. נמצא חסם תחתון לאלגוריתם שמצאנו - נביט למשל בגרף בו יש k קשתות, כאשר אף קשת אינו מחוברת לאחרת, והמשקלות אחידים. אזי:

$$OPT\left(I\right) = k$$

אם הפתרון השברי כזה שכל ה־$x_i = \frac{1}{2}$, אזי $ALG\left(I\right) = 2k$. זה מספיק בשביל להגיד שלאלגוריתם יש חסם תחתון (אפשר למצוא דוגמא יותר חזקה, אבל נפסח על זה).

2. נראה שלא קיים אלגוריתם אחר B שאפשר לקבל ממנו יחס קירוב יותר טוב על החסם התחתון שמצאנו - נראה שאפשר למצוא חסם תחתון על פער השלמות (פער השלמות מראה שאין אלגוריתם קירוב עם יחס פחות מ־α לעומת ה־OPT_F):

$$\left(\frac{B\left(I\right)}{OPT_F\left(I\right)} \geq\right) \frac{OPT\left(I\right)}{OPT_F\left(I\right)} \geq 2 - \frac{2}{n}$$

נראה זאת ע"י דוגמא נגדית: ניקח את I להיות קליקה בגודל n. פתרון לבעיה זו (לאו דוקא האופטימלי) הוא הפתרון בו $x_i = \frac{1}{2}$ לכל i. לפתרון זה העלות היא $\frac{n}{2}$. לכן עלות הפתרון האופטימלי הוא לא יותר מ־$\frac{n}{2}$, ולכן נקבל:

$$\begin{aligned} &OPT_F\left(I\right) \leq \frac{n}{2} \\ &OPT\left(I\right) = n - 1 \\ &\Rightarrow \frac{OPT\left(I\right)}{OPT_F\left(I\right)} \geq \frac{n-1}{n/2} = 2 - \frac{n}{2} \end{aligned}$$

3. האם יש חסם תחתון טוב יותר לבעיה עצמה? דיברנו על זה כבר בשבוע שעבר (משתמשים בהנחה $P \neq NP$).

מאוחר יותר נראה עוד דוגמא ל־VC, וגם לו יחס קירוב 2. החסרון היחיד במקרה שלנו הוא שצריך לפתור את ה־LP. בדוגמא בהמשך לא נצטרך לפתור את ה־LP, אך יהיה בו שימוש בניתוח (זהו ניתוח פשוט יותר, קיים גם ניתוח אחר שלא משתמש כלל ב־LP).

1.2.1 מחלקות נוספות

הגדרה 1.13 בעיה $L \in APX$ אם קיים קבוע α כך ש־L ניתנת לקירוב ביחס α (קיים אלגוריתם פולינומי עם יחס קירוב α).

למשל, $VC, WVC, TSP \in APX$, תזמון משימות.

הגדרה 1.14 בעיה $L \in PTAS$ (Poly-time approximation scheme) אם לכל $\varepsilon > 0$ קיים אלגוריתם פולינומי עם יחס קירוב $\varepsilon + 1$ ל־L.

הגדרה 1.15 בעיה $L \in FPTAS$ (Fully poly-time approximation scheme) אם $L \in PTAS$, כלומר אם לכל $\varepsilon > 0$ קיים אלגוריתם עם יחס קירוב $\varepsilon + 1$, וגם זמן ריצה פולינומי בגודל הקלט, וב־$\frac{1}{\varepsilon}$.

למשל, אלגוריתם שרץ ב־$n^{\frac{1}{\varepsilon}}$ טוב ל־$PTAS$ (כי לכל ε קבוע הוא פולינומיאלי) אבל לא טוב ל־$FPTAS$. דוגמא לזמן ריצה שטוב ל־$FPTAS$ - $\left(\frac{n}{\varepsilon}\right)^2$.
מתקיים:

$$P \subseteq FPTAS \subseteq PTAS \subseteq APX \subseteq NP$$

אם $P \neq NP$, אזי $PTAS \neq APX$, כי $VC \in APX$ אב תחת ההנחה $VC \notin PTAS$.

1.2.2 על אלגוריתם ה־$simplex$

ראינו את המבנה של LP:

$$min/max \ \sum_i c_i x_i$$
$$\forall j \ \sum_i a_{ij} x_i \geq / \leq / = b_j$$

דרך אחרת לכתוב את זה היא בעזרת וקטורים:

$$x = (x_1, x_2, ..., x_n)^T, \ c = (c_1, ..., c_n)^T, \ b = (b_1, ..., b_m)^T$$
$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots \\ & \ddots & \\ & & a_{nn} \end{pmatrix}, \ \sum c_i x_i = c^T x$$

נראה דוגמא פשוטה של LP בשני מימדים (שני משתנים):
נביט באילוצים הבאים:

$$min \ x_1 - 2x_2$$
$$x_1 + x_2 \leq 1$$
$$x_1 \geq 0$$
$$x_2 \geq 0$$

אם נסתכל על מערכת צירים (ציר ה־x מציין את x_1, ציר y מציין את x_2), אפשר לצייר את הטווח שלנו - אנו מסתכלים על הרביע הראשון, וישנו קו ישר העובר בנקודות $(0,1), (1,0)$

שבינו לבין ראשית הצירים נמצאים המשתנים שלנו. ניקח את פונקציית המטרה ונבחר c כך ש$x_1 - 2x_2 = c$, אז נקבל קו. כאשר "נשחק" עם ה־c נקבל קווים שונים מקבילים זה לזה.

השאלה הנשאלת היא, מהו ה־c הכי קטן בו הפונקציה עדיין בתחום של המשולש. אפשר להוכיח עבור המקרה שלנו כי המינימום והמקסימום יתקבלו בקודקודי המשולש (בשניים שאינם ראשית הצירים), או במקרה מסויים בצלע של המשולש (אם הקו מקביל בדיוק לצלע ־ עבור פונקציית מטרה אחרת ־ ואז ממילא מתקבל בזווית).

במקרה שלנו $c = -2$ הוא המינימלי (אפשר לבדוק את שני הקודקודים ולמצוא מיהו המינימום ומיהו המקסימום).

לכן, מספיק לבדוק מה קורה בקודקודים. אלגוריתם ה־$simplex$ (פתרון ל־LP) עושה בדיוק את זה! אמרנו שאלגוריתם זה הוא לעיתים אקספוננציאלי ־ זה קורה כאשר מספר הקודקודים הוא אקספוננציאלי. עדיין אלגוריתם זה עובד בזמן טוב, וזאת כי הוא עובד בצורה לוקלית (אינטואיטיבית הוא עובר מקודקוד לקודקוד שכן) , ובפועל זה עובד טוב (אין לכך הוכחה ריגורוזית).

1.2.3 LP בהצגה קנונית / סטנדרטית

הצורה הקנונית (מקרה פרטי של LP כללי):

$$min\ c^T x,\ Ax \geq b$$

הצורה הסטנדרטית ־ הופכים את אי השיוויונים לשיוויונים:

$$\begin{aligned} &min\ c^T x \\ &Ax = b \\ &x \geq 0 \end{aligned}$$

כל תוכנית LP כללית אפשר להפוך לצורה הסטנדרטית ־ לא נדון בזה.

כמו כן, כל תוכנית LP אפשר להפוך לצורה הקנונית ־ בטענה זו נדון.

צורה כללית $\leftarrow$ צורה קנונית

1. אם הבעיה היא בעיית מקסימום, כלומר $max\ c^T x \Leftarrow min\ -c^T x$.
2. אם יש שיוויון $\sum_i a_{ij} x_i = b_j \Leftarrow$ מפרקים לשני אי שיוויונות $\sum_i a_{ij} x_i \leq b_j,\ \sum_i a_{ij} x_i \geq b_j$.
3. אם יש אי שיוויון $\sum_i a_{ij} x_i \leq b_j \Leftarrow \sum_i (-a_{ij})\, x_i \geq -b_j$

1.2.4 דואליות של תכנון לינארי ־ LP Duality

נביט בתוכנית בצורה סטנדרטית (primal) P:

$$\begin{aligned} &P* = min\ x_1 + 2x_2 + 4x_3 \\ &x_1 + x_2 + 2x_3 = 5 \\ &2x_1 + x_2 + 3x_3 = 8 \\ &x_1 \geq 0,\ x_2 \geq 0,\ x_3 \geq 0 \end{aligned}$$

נרצה להעריך את הפתרון האופטימלי, כלומר נרצה להגיד ביטויים מהצורה $P* \geq something$. אזי, נרצה למצוא צ"ל של שני האילוצים שיתן לנו את פונקציית המטרה. אם נמצא y_1, y_2 כך ש:

$$P* = x_1 + 2x_2 + 4x_3 \geq y_1 \cdot (x_1 + x_2 + 2x_3) + y_2 \cdot (2x_1 + x_2 + 3x_3)$$

נקבל כי הביטוי גדול מ־

$$= 5y_1 + 8y_2$$

בדוגמא זו אפשר למצוא כאלו ־ $y_1 = 1,\ y_2 = 0$, ואז נקבל כי הפתרון האופטימלי לא יכול להיות קטן יותר מ־5. עוד דוגמא ־ $y_1 = 3,\ y_2 = -1$, ואז נקבל כי הפתרון האופטימלי לא יכול להיות קטן מ־7, כלומר הוא לפחות 7. נטען שזה הכי טוב שאפשר, כלומר קיים פתרון שנותן 7. נראה שקיים כזה:

$$x = \begin{pmatrix} 3 \\ 2 \\ 0 \end{pmatrix}$$

אם נציב נקבל כי פתרון זה אכן נותן 7, כנדרש.
כך אפשר למצוא מהו הפתרון האופטימלי ללא פתרון ממש של הבעיה.

הדואליות נובעת מנסיון לעשות פורמליזציה לתהליך מעלה:

$$x_1 + 2x_2 + 4x_3 = P* \geq 5y_1 + 8y_2 = (x_1 + x_2 + 2x_3) \cdot y_1 + (2x_1 + x_2 + 3x_3) \cdot y_2$$
$$(y_1 + 2y_2) \cdot x_1 + (y_1 + y_2) \cdot x_2 + (2y_1 + 3y_2) \cdot x_3$$

בהנחה ש־$x_i \geq 0$, נקבל כי כל פתרון המקיים את התנאים:

$$y_1 + 2y_2 \leq 1$$
$$y_1 + y_2 \leq 2$$
$$2y_1 + 3y_2 \leq 4$$

יקיים את המטרה, ונרצה למצוא את החסם התחתון הכי גדול מבין הפתרונות הללו, כלומר מצאנו כי אנו רוצים למצוא את הפתרון לתוכנית לינארית חדשה:

$$q* = max\ 5y_1 + 8y_2$$
$$y_1 + 2y_2 \leq 1$$
$$y_1 + y_2 \leq 2$$
$$2y_1 + 3y_2 \leq 4$$

זוהי נקראת תוכנית לינארית דואלית ־ נסמנה ב־D (dual).
דהיינו, כל פיתרון אפשרי ל־D נותן ערך $\geq P*$ לפתרון אפשרי ל־P.
בדוגמא הספציפית שלנו, ראינו כי $P* \leq 7$, וכן ראינו הצבה עבורה $q* \geq 7$, ולכן:

$$7 \geq P* \leq q* \geq 7$$

למה זה נכון תמיד? לא נוכיח זאת.

$$OPT_{DLP}(I) = OPT_{LP}(I) \leq OPT(I)$$
$$q \leq P* \leq OPT(I)$$

לכל פתרון אפשרי q של D:

$$ALG(I) \leq \alpha \cdot q$$

נצטרך להגדיר קצת יותר טוב איך נראית הדואליות.
לתוכנית בצורה סטנדרטית:

$$min\ c^T x$$
$$Ax = b$$
$$x \geq 0$$

התוכנית הדואלית תהיה:

$$max\ b^T y$$
$$A^T y \leq c$$

משפט הדואליות החלשה

משפט 1.16 אם x הוא פתרון ל־P, ו־y הוא פתרון ל־D, אז:

$$c^T x \geq b^T y$$

כלומר, ערך כל פתרון פרימלי הוא לפחות ערך הפתרון הדואלי.

הוכחה:

$$c^T x \geq \left(A^T y\right)^T x = y^T Ax = y^T b = b^T y$$

■

בדרך כלל משפט זה יספיק לנו. עם זאת, נראה גם את המשפט החזק:

משפט הדואליות החזקה

משפט 1.17 אם $x*$ פתרון אופטימלי ל־P (כלומר קיים פתרון אופטימלי ל־P), אז קיים פתרון אופטימלי $y*$ ל־D, ומתקיים:

$$c^T x* = b^T y^*$$

ההוכחה של משפט זה לא פשוטה. כמו כן משפט זה גם פועל הפוך.

תרגיל קל - התוכנית הדואלית של D היא P.

ישנה צורה קנונית מיוחדת נוספת:

$$\begin{aligned} &min\ c^T x \\ &Ax \geq b \\ &x \geq 0 \end{aligned}$$

כאשר הפתרון הוא מהצורה:

$$\sum_{1 \leq i \leq n} a_{ij} x_i \leq b_j$$

אם היינו מתחילים מצורה זו בדוגמא יכולנו לעשות את אותו הדבר. במקרה זה היינו מקבלים כי התוכנית הדואלית היא:

$$\begin{aligned} &max\ b^T y \\ &A^T y \leq c \\ &y \geq 0 \end{aligned}$$

כאשר הפתרון הוא מהצורה:

$$\sum_{1 \leq j \leq m} a_{ij} y_j \leq c_i$$

19/03/2012

יכול להיות שבמעבר לתוכנית הדואלית נקבל פחות משתנים מהתוכנית המקורית. עם זאת, הסיבה העיקרית במעבר לתוכנית הדואלית היא שפעולה זו נותנת לנו מבנה לבעיה.

הגדרה 1.18 P אפשרית אם קיים לה פתרון כלשהוא (לאו דווקא אופטימלי).

הגדרה 1.19 P לא חסומה אם לכל $t > 0$ קיים פתרון אפשרי x כך ש-$c^t x \leq -t$, כלומר אם הפתרון $\leftarrow -\infty$.

הגדרה 1.20 D לא חסומה אם לכל $t > 0$ קיים פתרון אפשרי y כך ש-$b^T y \geq t$.

מסקנה 1.21 (ממשפטי הדואליות) ייתכן בדיוק אחד מ-4 מצבים אפשריים:

1. P אפשרית ולא חסומה $\Leftarrow$ D לא אפשרית.
2. D אפשרית ולא חסומה $\Leftarrow$ P לא אפשרית.
3. אם P אפשרית וחסומה אז D אפשרית וחסומה, ולהפך.
4. D, P לא אפשריות.

כפי שאמרנו, הדרך להשתמש בניתוח זה לצורך אפרוקסימציה היא בעצם קבלת חסם תחתון נוסף למחיר אופטימלי.

$$ILP = OPT\left(I\right) \geq \left(LP =\right) OPT_F\left(I\right) = c^T x*$$

משפט הדואליות החלשה נותן לנו כי:

$$c^T x* \geq b^T y$$

כלומר, במקום לפתור את התוכנית הלינארית ולעשות עיגול, אפשר ליצור את התוכנית הדואלית ובעזרתה ליצור חסמים תחתונים למחיר האופטימלי ע"י בחירת פתרון כלשהוא של התוכנית הדואלית - כל אחד מהם יהווה חסם כנדרש (כמובן שצריך למצוא אחד כזה "בצורה חכמה" - אם נמצא פתרון שהוא קטן מידי, זה לא חסם הדוק, ולכן זה לא ממש יעזור לנו).

נרצה להוכיח שהמחיר של האלגוריתם שפותר את הבעיה הראשית מקיים:

$$ALG\left(I\right) \leq \alpha \cdot b^T y \left(\leq \alpha \cdot c^T x* = \alpha \cdot b^T y* \leq \alpha \cdot OPT\left(I\right)\right)$$

מסתבר שבמקרים מסויימים זה אכן יותר פשוט (החלק בסוגריים כבר ידוע לנו ממשפטי הדואליות).

כדי להשתמש באסטרטגיה זו בבעיית ה-VC, יש צורך בכלי נוסף. המשפט הבא נותן איפיון נוסף יותר שימושי ליחס בין הפתרונות האופטימליים.

עקרון העודף המשלים - Complementary Slackness Principle

משפט 1.22 יהיו x, y פתרונות אפשריים ל-P, D בהתאמה (בהצגה הקנונית המיוחדת - זו שראינו בסוף השיעור הקודם).

x, y הם פתרונות אופטימליים אמ"מ מתקיימים שני התנאים הבאים:

1. לכל $1 \leq i \leq n$, או $x_i = 0$ או $\sum_{1 \leq j \leq m} a_{ij} y_j = c_i$.
2. לכל $1 \leq j \leq m$, או $y_j = 0$ או $\sum_{1 \leq i \leq n} a_{ij} x_i = b_j$.

הוכחה: $\Rightarrow$ נניח ששני התנאים מתקיימים.

$$c^T x = \sum_{1 \leq i \leq n} c_i x_i = \sum_i \left(\sum_j a_{ij} y_j \right) x_i = \sum_j \left(\sum_i a_{ij} x_i \right) y_j = \sum_j b_j y_i = b^T y$$

קיבלנו שיוויון - נזכור כי השיוויון אומר כי הפתרונות אופטימליים (כי ממשפט הדואליות החלש $c^T x \geq b^T y$. המעברים נובעים ישירות מהתנאים - במקרים בהם $x_i = 0$ או $y_j = 0$ מתאפסים בשני המקומות).

$\Leftarrow$ נניח כי x, y הם פתרונות אופטימליים.

1. דומה לקודם, רק כאן נצטרך להפעיל יותר את הראש:

$$0 = c^T x - b^T y \geq c^T x - \left(Ax\right)^T y = x^T c - x^T A^T y = x^T \left(c - A^T y\right) \overset{A^T y \leq c,\ x_i \geq 0}{\geq} 0$$

לכן חייב להיות שיוויון לאורך כל הדרך, כלומר $x^T\left(c-A^Ty\right)=0$, ואת הביטוי הזה אפשר לכתוב בתור:

$$\sum_i x_i\left(c_i-\sum_j a_{ij}y_j\right)=0$$

x_i אי שליליים, וכן גם $\sum_j a_{ij}y_j$. לכן כדי שכל הביטוי יהיה שווה לאפס, או $x_i=0$ או $\left(c_i-\sum_j a_{ij}y_j\right)=0$ כלומר $c_i=\sum_j a_{ij}y_j$, כנדרש.

2. אותו דבר בצורה אנלוגית:

$$0=c^Tx-b^Ty\geq\left(A^Ty\right)x-b^Ty=y^TAx-y^Tb=y^T\left(Ax-b\right)\geq 0$$

וקיבלנו $y^T\left(Ax-b\right)=0$, או בכתיב מלא:

$$\sum_j y_j\left(\sum_i a_{ij}x_i-b_j\right)=0$$

שוב הגורמים אי שליליים, ולכן או $y_j=0$, או $\sum_i a_{ij}x_i-b_j=0$, כלומר $\sum_i a_{ij}x_i=b_j$.

■

יש שימוש בתוכניות לינאריות גם לפתרון של בעיות לינאריות (לא אפרוקסימציה). למשל בעיות $flow$ אפשר לפתור כך, ועוד תוכניות כאלו. אז העקרון הנ"ל הוא כלי שימושי להוכחה שהפתרון שהתקבל הוא אופטימלי. לנו זה פחות שימושי, כי אנחנו מחפשים אפרוקסימציה, ולא את הפתרון האופטימלי. לכן בפועל נשתמש בהרחבה של הפתרון המשלים, שיתייחס רק לכיוון הקל של העקרון הנ"ל:

עקרון העודף המשלים המורחב

משפט 1.23 יהיו x,y פתרונות ל־P,D בהתאמה (בצורה קנונית מיוחדת) המקיימים את התנאים הבאים:

1. לכל $1\leq i\leq n$, או ש־$x_i=0$ או $\sum_j a_{ij}y_j\geq\frac{c_i}{\alpha}$.
2. לכל $1\leq j\leq m$, או ש־$y_j=0$ או $\sum_j a_{ij}x_i\leq\beta\cdot b_j$.

כאשר $\alpha,\beta\geq 1$.
אזי:

$$c^Tx\leq\alpha\beta\cdot b^Ty\left(\leq\alpha\beta\cdot b^Ty*=\alpha\beta\cdot OPT_F\left(I\right)\right)$$

נזכור כי ממשפט הדואליות החלשה, $c^Tx\geq b^Ty$. אזי, אם נמצא פתרונות שמקיימים את התנאים בעקרון הנ"ל, נקבל כי למעשה הפתרון של האלגוריתם חסום ע"י הפתרון הדואלי,

מלמטה על ידו, ומלמעלה ע"י הפתרון הדואלי כפול $\alpha \cdot \beta$ כלשהוא. עם זאת, לפני השימוש, עלינו להוכיח את נכונות העקרון: **הוכחה:**

$$c^T x = \sum_i c_i x_i \overset{1}{\leq} \sum_i \left(\alpha \sum_j a_{ij} y_j \right) x_i = \alpha \sum_j \left(\sum_i a_{ij} x_i \right) y_j \overset{2}{\leq} \alpha\beta \sum_j b_j y_j$$

■

אנחנו כאילו מחפשים שני פתרונות - גם פתרון דואלי וגם פתרון פרימלי. אם הם יקיימו את התנאים של עקרון העודף המשלים המורחב, נדע שהם קשורים זה לזה באי השיוויון $c^T x \leq \alpha\beta \cdot b^T y$, ולכן אפשר להשתמש בהם כדי לחסום את הפתרון הרצוי. השימוש הזה נקרא Primal-Dual approach, וזו שיטה מאוד ידועה ומקובלת. לא תמיד זה קל, אבל במקרה שלנו זה יהיה קל:
נחזור לבעיית ה-WVC שלנו:

$$\begin{aligned}
&LP\ \ (P)\\
&min \sum_{i \in V} w_i x_i\\
&\forall \{i,l\} \in E\ x_i + x_l \geq 1\\
&\forall i \in V\ x_i \geq 0
\end{aligned}$$

התוכנית הדואלית D שתתאים לבעיה זו היא (מתקבל משתנה דואלי לכל קשת $y_{i,l} = j$):

$$\begin{aligned}
&max \sum_{\{i,l\} \in E} y_{il}\\
&\forall i \in V \sum_{l:\{i,l\} \in E} y_{il} \leq w_i\\
&\forall \{i,l\} \in E\ y_{il} \geq 0
\end{aligned}$$

לאלגוריתם ל-WVC ישנן שתי הצגות - אחת מאוד פשוטה ואינה קשורה ל-LP, והשניה יותר מסובכת אבל קשורה ל-LP. נראה תחילה את ההצגה השניה:

אלגוריתם 3 WVC - Primal-Dual

נמצא פתרון שלם x ופתרון דואלי y (לא שלם) (שיקיימו $c^T x \leq \alpha\beta \cdot b^T y$ עבור $\alpha, \beta \geq 1$):
נתחיל מפתרונות התחלתיים $\forall i \in V\ x_i = 0$, $\forall \{i,l\} \in E\ y_{il} = 0$.

- בחר קשת $\{i,l\}$ שאינה מכוסה עדיין (כלומר $x_i = x_l = 0$) - אם אין כזו, סיימנו.
- הגדל את $y_{i,l}$ עד שאחד מאי השיוויונים (הדואליים) ב-D הופך להדוק (יש שניים כאלו - אחד של i ואחד של l, כלומר אי השיוויון עובר לשיוויון - נבחר את זה הקרוב ביותר לערך הנדרש).
- אם אי השיוויון של D המתאים ל-i נהיה הדוק, אז נקבע $x_i = 1$. אם אי השיוויון של D המתאים ל-l נהיה הדוק, נקבע $x_l = 1$.

ניתן תיאור אלטרנטיבי לאלגוריתם שאינו קשור ל-LP:

- בחר קשת לא מכוסה $\{i,l\}$, בחר קודקוד i (בה"כ) שמשקלו מינימלי.
- הורד את משקל w_i ממשקל קודקוד l, והכנס את i לכיסוי. הכנס את l לכיסוי גם כן אם המשקל שלו הופל ל־0.
- חזור על התהליך עד שאין קשת שאינה מכוסה.

נשים לב שקיבלנו הרחבה מאוד טבעית של האלגוריתם שראינו בשיעור הראשון לבעיה הלא ממושקלת VC.

משפט 1.24 לאלגוריתם P-D ל־WVC יש יחס קירוב 2.

הוכחה: למעשה, צריך להראות כי הפתרון מקיים את עקרון העודף המשלים המורחב.
תנאי 1: עבור $\alpha = 1$ ברור מהגדרת האלגוריתם, שכן ברגע שיש אילוץ ב־D שמתקים ל־i הדוק, אז קובעים את $x_i = 1$.
תנאי 2: $x_i + x_j \leq 2 \Leftarrow x_i, x_l \leq 1$ (אם $y_{i,l} \neq 0$, אז במקרה הגרוע גם $x_i = 1$ וגם $x_l = 1$), ולכן התנאי מתקיים עבור $\beta = 2$.
לכן, מעקרון העודף המשלים המורחב:

$$w(C) = \sum_i w_i x_i \leq 2 \cdot OPT_F(I) \leq 2 \cdot OPT(I)$$

כנדרש. ■

1.2.5 Weighted Set Cover

נתון עולם $U = \{e_1, ..., e_n\}$ בגודל n, ואוסף של תת קבוצות $\mathcal{S}$ כך ש:

- $\bigcup_{S \in \mathcal{S}} S = U$.
- לכל $S \in \mathcal{S}$, $S \subset U$.

ונתונה פונקציית משקל:

$$w : \mathcal{S} \to \mathbb{R}^+$$

רוצים למצוא כיסוי $C \subset \mathcal{S}$ (קבוצה של קבוצות) כך ש: $\bigcup_{S \in C} S = U$ בעל משקל מינימלי $w(C) = \sum_{S \in C} w(S)$. כלומר:

$$min \sum_{S \in \mathcal{S}} w(S) \cdot x_S \ \ (ILP: \ x_S = 1 \Leftrightarrow S \in C)$$
$$\forall e \in U \sum_{\mathcal{S} \ni S | e \in S} x_S \geq 1$$
$$\forall s \in S \ x_S \geq 0 \ \ (ILP: \ x_S \in \{\{0,1\}\})$$

קיבלנו תוכנית בצורה הקנונית המיוחדת. התוכנית הדואלית המתאימה אם כך תהיה:

$$max \sum_{e \in U} y_e$$
$$\forall s \in S \sum_{e \in S} y_e \leq w(S)$$
$$\forall e \in U \; y_e \geq 0$$

ננסה למצוא אינטרפטציה טבעית לבעיות מעלה בשביל אינטואיציה:
נביט על התוכנית המקורית - היא בעיית כיסוי. התוכנית הדואלית עם זאת היא "בעיית אריזה" $(knapsack)$ - מנסים "לארוז" משקל מקסימלי תחת אילוצים (וקטור שמגביל אותנו באיזושהיא צורה).
באופן כללי, ניקח את המבנה שדיברנו עליו של התוכנית הקנונית המיוחדת - אם כל המקדמים הם אי שליליים, אז זוהי בעיית כיסוי.

אלגוריתם 4 האלגוריתם החמדן ל-WSC

נסמן:

- $A \subseteq U$ אוסף האיברים שכבר כוסו עד לשלב הנוכחי בריצת האלגוריתם. מספר האיברים שעדיין לא כוסו - $|S \backslash A|$.
- $\gamma(S) = \frac{w(S)}{|S \backslash A|}$ יהיה יחס תועלת העלות. כאשר $S \backslash A = \phi$, $\gamma(S) = \infty$.
- $\forall e \in S \backslash A \; price(e) = \gamma(S)$.

האלגוריתם:

- $\forall s \in S \; x_s := 0$
- $A \Leftarrow \phi$.
- אם $A \neq U$ אז:
  - $S := arg \min_{S \in \mathcal{S}, \; x_S \neq 1} \gamma(S)$.
  - $\forall e \in S/A \; price(e) = \gamma(S)$ (צעד זה לא הכרחי לאלגוריתם, הוא רק לצורך הניתוח).
  - $A := A \cup S$.
  - $x_S := 1$.

נשים לב - $Greedy(I) = \sum_{e \in U} price(e)$.

משפט 1.25 לאלגוריתם ה-$Greedy$ יש יחס קירוב של $H_n \leq ln(n) + 1$.

26/03/2012

$H_n = 1 + \frac{1}{2} + \frac{1}{3} + ... + \frac{1}{n}$ הטור ההרמוני, וידוע כי $ln(n) < H_n < ln(n) + 1$. **הוכחה:** רוצים להגדיר פתרון דואלי y.

$$y_e := \frac{price(e)}{H_n}$$

אם נראה שזהו פתרון חוקי, אז הוכחנו את יחס הקירוב הדרוש ממשפט הדואליות החלש, שכן:

$$Greedy(I) = \sum_{e \in U} price(e) = H_n \cdot \sum_{e \in U} y_e \leq H_n \cdot OPT_F(I) \leq H_n \cdot OPT(I)$$

כאשר הביטוי השלישי משמאל הוא הפתרון הדואלי, ולכן הוא תמיד קטן מהערך האופטימלי של התוכנית הראשית.

נוכיח אם כך את חוקיות הפתרון:

נתבונן בקבוצה כלשהיא $S \in \mathcal{S}$. נסתכל על האיברים של S, ונסדר אותם לפי סדר הכיסוי שלהם.

יהי e_i האיבר ה־i בסידור זה. נשים לב כי $\gamma(S)$ משתנה בכל שלב לפי מספר האיברים שכבר כוסו (ישירות מההגדרה). נתייחס מטה לערך של $\gamma(S)$ בשלב הנוכחי. בשלב זה $|S \backslash A| \geq k - i + 1$, בהנתן $|S| = k$, ואז $|S \cap A| \leq i - 1$. אזי:

$$price(e_i) \leq \frac{w(S)}{k - i + 1}$$

$$\sum_{e \in S} y_e \overset{def}{=} \sum_{e \in S} \frac{price(e_i)}{H_n} \leq \frac{w(S)}{H_n} \left(\frac{1}{k} + \frac{1}{k-1} + ... + 1 \right) = \frac{H_k}{H_n} w(S) \leq w(S)$$

ולכן זהו פתרון חוקי, כנדרש. ■

ננתח את הפתרון לפי השאלות הרגילות שלנו:

1. נשאל, האם לאלגוריתם הגרידי יש יחס קירוב הדוק יותר מזה שמצאנו? לא! נראה זאת ע"י דוגמא, כלומר, מחפשים דוגמא בה המחיר האופטימלי הוא נמוך, כלומר יש כיסוי בגודל קטן (מותר להשתמש במשקלות), ולמרות זאת המחיר של האלגוריתם יהיה גדול.
הדוגמא די פשוטה[3]: יש n איברים, לכל איבר יש קבוצה, לקבוצה שכוללת את כל האיברים יש משקל של $1 + \epsilon$. לקבוצה הראשונה יש משקל של $\frac{1}{n}$, לבאה $\frac{1}{n-1}$ (אחרי שכבר בחרנו את האלמנט הראשון), לבאה $\frac{1}{n-2}$, וכו' עד 1.
המחיר של האלגוריתם הגרידי למקרה הזה הוא H_n (בוחר את כל הקבוצות הבודדות), והמחיר של האופטימלי הוא $1 + \epsilon$.

2. "פער השלמות" - או שזה המקרה הכי טוב אבסולוטית, או שיש דרך אחרת (ניגע בהן בהמשך). נרצה להוכיח:

$$\frac{OPT(I)}{OPT_F(I)} \geq \frac{log(n+1)}{2}$$

[3] זוהי דוגמא עם משקלות, אבל אפשר לעשות זאת ללא - תרגיל לתלמיד השקדן.

מדוע? בהנתן $U = \{e_1, ..., e_2\}$, נסתכל על האינדקסים $1 \leq i \leq n$ כמספרים בעלי k ביטים, כלומר $n = 2^k - 1$, ונחשוב על האינדקסים הללו כעל וקטורים בשדה $\mathbb{Z}_2$ - למשל, אפשר להכפיל את המספרים בצורה הבאה[4]:

$$i \cdot j := \left(\sum_l i_l j_l \right) mod 2$$

החיבור $mod 2$ הוא xor. נגדיר את הקבוצות S_i בצורה הבאה:

$$S_i = \{e_j | i \cdot j = 1\}, \ \mathcal{S} = \{S_1, S_2, ..., S_n\}$$

מה הגודל של S_i? בדיוק חצי מהוקטורים הם אלו שנותנים 1, וחצי הם אלו שנותנים 0, כלומר $|S_i| = 2^{k-1} = \frac{n+1}{2}$. כמו כן, לכל איבר, מספר הקבוצות שמכילות אותו הוא $2^{k-1} = \frac{n+1}{2}$ (שהרי כל איבר בפעולת הכפל מעלה עם מחצית מהאיברים יתן 1).
בהנתן שהמשקלות הן 1 לכל האיברים, הפתרון האופטימלי הוא:

$$\forall i \ x_{S_i} = \frac{2}{n+1}$$

כלומר, זהו מחיר לכל קבוצה שמכילה את האיבר הזה. אז, בסכימה של האיבר על הקבוצות שמכילה אותו, נקבל 1, כלומר הוא מוכל. אזי (זהו פתרון אופטימלי אך לא נראה זאת, ולכן המעבר הראשון הוא לא שיוויון):

$$OPT_F(I) \leq n \cdot \frac{2}{n+1} < 2$$

נניח בשלילה שהאלגוריתם האופטימלי בוחר $k > p$ קבוצות לכיסוי.

$$\frac{OPT(I)}{OPT_F(I)} \geq \frac{k}{2} = \frac{log(n+1)}{2}$$

נסמן $i_1, ..., i_p$ האינדקסים של הקבוצות. נתבונן במטריצה:

$$B = \begin{pmatrix} \cdots & \cdots & i_1 & \cdots & \cdots \\ & & i_2 & & \\ & & \vdots & & \\ & & \vdots & & \\ & & i_p & & \end{pmatrix} \ p \times k$$

לפי ההגדרה, דרגתה $k > p \geq$ (כלומר היא לא מדרגה מלאה). אזי, מרחב האפס (הקרנל) של B לא מכיל רק את 0, כלומר $\exists j \neq 0 \ B \cdot j = 0$, ולכן e_j לא מכוסה (שכן לא קיים $1 \leq l \leq p$ כך ש-$i_l \cdot j = 1$), בסתירה לבניה שלנו. על כן, החסם שלנו מתקיים, כנדרש.

3. האם אפשר לשפר את האפורוקסימציה (בעזרת אלגוריתם אחר)? במידה ו-$P \neq NP$ אז יחס הקירוב לבעיה הוא לפחות $ln(n)$ (תוצאה של Feige).

[4] יש כאן טענה שזוהי מכפלה פנימית, אך הדרישה של אפס אמ"מ זהו וקטור האפס לא מתקיימת.

ניתן פתרון נוסף לבעיה:

$$min \sum_{S \in \mathcal{S}} w(S) \cdot x_S$$
$$\forall e \in U \sum_{S:S \ni e} X_S \geq 1$$
$$\forall S \in \mathcal{S}\ x_S \geq 0$$

קיים פתרון אופטימלי בו $0 \leq x_S \leq 1$.

אלגוריתם 5 עיגול רנדומלי - Randomized Rounding

- נפתור את תוכנית ה־LP.
- נבחר את S להשתתף בכיסוי בהסתברות x_S.
- כל t איטרציות שכאלו יהיו כיסוי אחד.

נבחר r כיסויים כאלו, ונבחר את הטוב מבין החוקיים.
הבעיה: זה לא בהכרח כיסוי.
זהו אלגוריתם רנדומי - הוא לא תמיד יתן לנו תשובה. באלגוריתם שכזה, אם אם הוא נותן תוצאה טובה (התקבל כיסוי והמחיר נמוך) בהסתברות לא קטנה מידי זה טוב - פשוט נריץ אותו הרבה פעמים, ואז ניקח את הפתרון הכי טוב מבין החוקיים שהתקבלו.
אזי, הדרישות שלנו:

- כיסוי חוקי.
- מחיר $ALG(I) \leq c \cdot log(n) \cdot OPT(I)$.

תחילה, ננתח את ההסתברות בה יתקבל כיסוי חוקי:
נביט על אלמנט באיבר e. לכל $e \in S$, בהסתברות x_S e מכוסה. נניח $t \leq 3 \cdot ln(n)$, $n \geq 2$.

$$Pr[\text{e is not covered}] \leq \prod_{S:S \ni e} 1 - x_S \overset{infi!}{\leq} \prod_{S:S \ni e} \exp^{-x_S} = \exp^{-\sum_{S:S \ni e} x_S} \leq \frac{1}{\exp}$$

$$Pr[\text{e is not covered in t iterations}] \leq \left(\frac{1}{\exp}\right)^t = n^{-k} \leq \frac{1}{4n} \leq \frac{1}{\exp}$$

$$Pr[\exists e \in U \text{ which is not covered in } r\text{runs}] \overset{\text{union}}{\leq} n \cdot \frac{1}{4n} \leq \frac{1}{4}$$

כעת, ננתח את מחיר האלגוריתם:
תחילה, אבחנה (יחסית פשוטה) - מבחינת התוחלת "אנחנו בסדר".

הערה 1.26 לפעמים מסתפקים בלהראות שהתוחלת טובה לעומת האופטימלי. זוהי תוצאה יותר חלשה ממש שנעשה במובן מסויים (אך לא תמיד).

זה קל - נסמן $\hat{x}_S = 1$ אם S בכיסוי, 0 אחרת. כמה האלגוריתם משלם באיטרציה אחת אם הוא בוחר את S? כל S נותן x_S לתוחלת, ולכן:

$$\mathbb{E}\left[w\left(S\right) \cdot \hat{x}_S\right] = x_S \cdot w\left(S\right)$$

עבור כל הקבוצות אם כך נקבל:

$$\mathbb{E}\left[\sum_{S \in \mathcal{S}} w\left(S\right) \cdot \hat{x}_S\right] = \sum_{S \in \mathcal{S}} w\left(S\right) \cdot x_S = OPT_F\left(I\right)$$
$$\mathbb{E}\left[ALG\left(I\right)\right] \leq t \cdot OPT_F\left(I\right)$$

לכן "זוהי סוג התוצאה שרצינו להראות".
נזכר באי שיוויון מרקוב עבור $c \geq 1$:

$$Pr\left[X > c \cdot \mathbb{E}\left[X\right]\right] \leq \frac{1}{c}$$

אזי ההסתברות ש-$ALG\left(I\right) > 4t \cdot OPT_F\left(I\right)$. $\frac{1}{4} \geq$ כלומר, בהסתברות לכל היותר $\frac{1}{4}$ המחיר גדול.

מצד שני, ראינו כי בהסתברות לכל היותר $\frac{1}{4}$ שהפתרון לא חוקי.

לכן, ההסתברות ששני המקרים מתרחשים חסומה ע"י האיחוד, כלומר לכל היותר $\frac{1}{2}$, ולכן לפחות בהסתברות חצי המחיר קטן והכיסוי חוקי. כאשר חוזרים על הניסוי r פעמים, ההסתברות לאי הצלחה היא $\geq \left(\frac{1}{2}\right)^r$.

Max SAT 1.3

16/4/2012

בעיית אופטימיזציה - רוצים לספק כמה שיותר פסוקיות.
נתונה נוסחת CNF, כלומר נוסחא מהצורה:

$$f = \bigwedge_{j=1}^{m} C_j$$

פסוקית C_j מגודל k_j - מכילה k_j ליטרלים, כלומר מהצורה:

$$C_j = \bigvee_{i=1}^{k_j} u_j$$

כאשר u_i הוא אחד מהמשתנים $x_1, ..., x_n$ או השלילה שלהם $\overline{x}_1, ..., \overline{x}_n$, ומשתנה לא מופיע יותר מפעם אחת באותה הפסוקית.

יש גרסא ממושקלת ולא ממושקלת - רוצים למצוא השמה למשתנים המספקת במקרה הלא ממושקל, כמה שיותר פסוקיות, או במקרה הממושקל, פסוקיות במשקל הגדול ביותר.

אנו נדון במקרה הממושקל - לכל פסוקית c_j נתון משקל w_j. נרצה למצוא השמה למשתנים הממקסמת את הביטוי $\sum w_j c_j$ כאשר:

$$c_j = \begin{cases} 1 & C_j \; is \; valid \\ 0 & otherwise \end{cases}$$

לבעיה זו יש כמה מקרים פרטיים מאוד חשובים:

- Max K-SAT - בה לכל j, $k_j \leq k$.
- Max E(XACT) K-SAT - בה לכל j, $k_j = k$.

נדון תחילה באלגוריתם רנדומיים לבעיה, לאחר מכן נראה שיטה כללית בעזרתה ניתן להגיע לאלגוריתם דטרמיניסטי (תהליך זה נקרא דרנדומיזציה).
נשתמש בהגדרה הבאה:

הגדרה 1.27 לאלגוריתם פולינומיאלי רנדומי A יש יחס קירוב α אם לכל מופע I:

$$\mathbb{E}\left[A\left(I\right)\right] \geq \alpha \cdot OPT\left(I\right)$$

1.3.1 אלגוריתם רנדומי ראשון

נתחיל עם אלגוריתם יחסית פשוט:

אלגוריתם 6 MAX SAT I (Johnson)

- נגריל לכל משתנה x_i ערך $0/1$ בהסתברות $\frac{1}{2}$.

משפט 1.28 יחס הקירוב של אלגוריתם 6 הוא $\frac{1}{2}$.

הוכחה:

$$\mathbb{E}\left[A\left(I\right)\right] = \mathbb{E}\left[\sum w_j c_j\right] = \sum w_j \mathbb{E}\left[c_j\right] (*)$$
$$\mathbb{E}\left[c_j\right] = 1 \cdot Pr\left[c_j = 1\right] + 0 \cdot Pr\left[c_j = 0\right] = Pr\left[c_j = 1\right] = 1 - Pr\left[c_j = 0\right]$$
$$Pr\left[c_j = 0\right] = \left(\frac{1}{2}\right)^{k_j}$$

נסמן $\alpha_{k_j} = 1 - \left(\frac{1}{2}\right)^{k_j}$. אזי:

$$\alpha_k \equiv 1 - \left(\frac{1}{2}\right)^{k} \geq \frac{1}{2}$$

על כן, נקבל:

$$(*) \geq \frac{1}{2} \sum w_j \geq \frac{1}{2} OPT$$

■

הערה 1.29 בהמשך נראה איך לעשות דרנדומיזציה לתהליך.

הערה 1.30 מתוך ההוכחה ניתן לראות כי הבעיה שלנו במציאת היחס היא עם פסוקיות בגודל 1 (אם לא היו כאלה היינו יכולים למצוא יחס קירוב טוב יותר). עם זאת, שיפרו את היחס הזה רק בשנת 95. הרעיון - לתת אלגוריתם שיהיה טוב דווקא לפסוקיות הקצרות.

המקרה הכי גרוע - עבור פסוקית עם ליטרל אחד x_i, נקבל עלות $\frac{1}{2}$.

1.3.2 אלגוריתם רנודמי שני

טוב למקרה הכללי, אבל חסום ע"י k, ולכן מאוד טוב עבור בעיית ה־MAX K-SAT (בו $k_j \leq k$).

על מנת להגיע לאלגוריתם המשופר, נביע את הבעיה בצורת ILP:

כאן בניגוד למה שראינו בעבר, השאלה האם פסוקית מסופקת תלויה בשאלה האם המשתנה x_i מופיע בפסוקית בצורה החיובית או השלילית שלו.

בשביל להבדיל בין שני המקרים, נגדיר שתי קבוצות:

$$P_j = \{i \,|x_i \in C_j\}\,,\ N_j = \{i \,|\overline{x}_i \in C_j\}$$

כעת, התוכנית ILP המתאימה לבעיה הינה:

$$\begin{aligned}
&max \sum_{j=1}^{m} w_j c_j \\
&\forall 1 \leq j \leq m : \sum_{i \in P_k} x_i + \sum_{i \in N_j} (1 - x_i) \geq \sum c_j \\
&1 \leq j \leq m \ c_j \in \{0,1\} \\
&1 \leq i \leq n \ x_i \in \{0,1\}
\end{aligned}$$

נהפוך ל־LP (נשנה את שמות המשתנים כדי להבדיל בין התוכניות ביתר קלות בעת ההוכחה):

$$\begin{aligned}
&max \sum_{j=1}^{m} w_j z_j \\
&\forall 1 \leq j \leq m : \sum_{i \in P_k} y_i + \sum_{i \in N_j} (1 - y_i) \geq \sum z_j \\
&1 \leq j \leq m \ 0 \leq z_j \leq 1 \\
&1 \leq i \leq n \ 0 \leq y_i \leq 1
\end{aligned}$$

נשים לב שכאן בניגוד לבעיות הקודמות, אי אפשר לוותר על ההגבלה $z_j \leq 1$ שכן כאן מדובר בבעיית מקסימיזציה ולא מינימיזציה. אפשר לוותר על התנאי $y_i \leq 1$ (שכן ההגבלה $z_j \leq 1$ משליכה על ה־y_i).

האלגוריתם השני משתמש בעיגול רנדומלי:

אלגוריתם 7 MAX SAT II - Goeman-Williamson

- פתור את בעיית ה־LP.
- קבע $x_i = 1$ בהסתברות y_i, $x_i = 0$ בהסתברות $1 - y_i$.

משפט 1.31 לאלגוריתם 7 יש יחס קירוב של $1 - \frac{1}{e} \approx 0.63 \geq \frac{1}{2}$.

הערה 1.32 בהמשך נראה ששילוב של אלגוריתם שטוב לפסוקיות קצרות (7) ואלגוריתם שטוב לפסוקיות ארוכות (6) אפילו יותר טוב!

הוכחה: הוכחת המשפט עוברת דרך טענה:

טענה 1.33

$$\forall 1 \leq j \leq m \ \mathbb{E}[c_j] \geq \beta_{k_j} \cdot z_j$$

כאשר:

$$\beta_l \equiv 1 - \left(1 - \frac{1}{l}\right)^l, \ \beta_l \geq 1 - \frac{1}{e}$$

נשים לב כי:

$$\begin{aligned} \beta_1 &= 1 \\ \beta_2 &= \frac{3}{4} \\ &\vdots \end{aligned}$$

כלומר האיכות מתדרדרת עם אורך הפסוקית, והגבול הוא $1 - \frac{1}{e}$, כאשר השאיפה היא מלמעלה.

זה יספיק לנו כי אז, בדומה לאלגוריתם הקודם, נקבל:

$$\mathbb{E}[A(I)] = \mathbb{E}\left[\sum w_j c_j\right] = \sum w_j \mathbb{E}[c_j] \geq \sum w_j \cdot \beta_{k_j} \cdot z_j \geq \left(1 - \frac{1}{e}\right) \sum w_j z_j$$
$$= \left(1 - \frac{1}{e}\right) OPT_F(I) \geq \left(1 - \frac{1}{e}\right) OPT(I)$$

כנדרש. אם כך נותר רק להוכיח את הטענה: **הוכחה:**

$$\mathbb{E}[c_j] = Pr[c_j = 1] = 1 - Pr[c_j = 0]$$
$$Pr[c_j = 0] = \prod_{i \in P_j} (1 - y_i) \cdot \prod_{i \in N_j} y_i \ (*)$$

כאן נשתמש באי שיוויון הממוצעים: בהנתן $a_1, ..., a_n \geq 0$, הממוצע ההנדסי שלהם חסום ע"י הממוצע החשבוני שלהם, כלומר:

$$\sqrt[n]{a_1 \cdot ... \cdot a_n} \leq \frac{a_1 + ... + a_n}{n}$$

בעזרת אי שיוויון הממוצעים, נשים לב כי מספר האיברים בביטוי מעלה הוא k_j, ולכן:

$$(*) = \left(\sqrt[k_j]{\prod_{i \in P_j} (1-y_i) \cdot \prod_{i \in N_j} y_i} \right)^{k_j} \leq \left(\frac{\sum_{i \in P_j} (1-y_i) + \sum_{i \in N_j} y_i}{k_j} \right)^{k_j}$$

$$= \left(\frac{|P_j| - \sum_{i \in P_j} y_i + |N_j| - \left(1 - \sum_{i \in N_j} y_i\right)}{k_j} \right)^{k_j}$$

$$\overset{|P_j|+|N_j|=k_j}{=} \left(1 - \frac{\sum_{i \in P_j} y_i + \left(1 - \sum_{i \in N_j} y_i\right)}{k_j} \right) \overset{LP \text{ condition}}{\leq} \left(1 - \frac{z_j}{k_j}\right)^{k_j}$$

$$\Rightarrow \mathbb{E}[c_j] \geq 1 - \left(1 - \frac{z_j}{k_j}\right)^{k_j} \overset{?}{\geq} \beta_{k_j} z_j$$

? תמיד מתקיים, כי לכל $0 \leq x \leq 1$, $l > 0$:

$$f(x) = \left(1 - \frac{x}{l}\right)^l \geq \left(1 - \left(1 - \frac{1}{l}\right)^l\right) x$$

זהו תרגיל באינפי, נגיד בכמה מילים איך מוכיחים אותו:
נביט בגרף שתי הפונקציות משני צידי אי השיוויון. הטענה היא כי הגרף של הצד השמאלי והימני מתלכדים ב־0 וב־1, אבל $f(x)$ תמיד מעל הצד השני, שהוא קו ישר - זוהי פונקציה קעורה (וזה קל לבדוק, בעזרת גזירה למשל).

■

■

1.3.3 שילוב שני האלגוריתמים הקודמים

נרצה לעשות שילוב של שני האלגוריתמים הקודמים, על מנת להגיע לתוצאה יותר טובה.

אלגוריתם 8 MAX SAT III

- נגריל בהסתברות שווה בין האלגוריתמים, ונריץ את האלגוריתם שנבחר.

משפט 1.34 לאלגוריתם 8 יחס קירוב $\frac{3}{4}$.

הוכחה: שוב בעזרת טענה:

טענה 1.35

$$\forall 1 \leq j \leq m \; \mathbb{E}[c_j] \geq \frac{3}{4}$$

שכן אם הטענה מתקיימת, אזי:

$$\mathbb{E}\left[A\left(I\right)\right]=\sum w_j\mathbb{E}\left[c_j\right]\geq\frac{3}{4}\overbrace{\sum w_jz_j}^{OPT_F(I)}\geq\frac{3}{4}OPT\left(I\right)$$

הוכחה:

$$\begin{aligned}&\mathbb{E}\left[c_j\right]\leq\frac{1}{2}\alpha_{k_j}+\frac{1}{2}\beta_{k_j}\cdot z_j\geq\frac{1}{2}z_j\left(\alpha_{k_j}+\beta_{k_j}\right)\\&\forall1\leq l\ \frac{1}{2}\left(\alpha_l+\beta_l\right)\overset{?}{\geq}\frac{3}{4}:\\&l=1\ \frac{1}{2}\left(\frac{1}{2}+1\right)=\frac{3}{4}\\&l=2\ \frac{1}{2}\left(\frac{3}{4}+\frac{3}{4}\right)=\frac{3}{4}\end{aligned}$$

וכן בהמשך עבור $l\geq3$:

$$\frac{1}{2}\left(\alpha_l+\beta_l\right)\geq\frac{1}{2}\left(\frac{7}{8}+\left(1-\frac{1}{e}\right)\right)\geq\frac{3}{4}$$

■

■

נרצה לענות שוב על השאלות הרגילות שלנו.

1. נרצה למצוא קלט שבאמת מקבל $\frac{3}{4}$ (כדי להראות שלאלגוריתם אין קירוב טוב יותר, כלומר הניתוח שלנו הוא הכי טוב) - דוגמא פשוטה היא הנוסחא $f=x_1$: עבור האלגוריתם הראשון הוא מקבל קירוב $\frac{1}{2}$, ועבור האלגוריתם השני הוא מקבל קירוב 1, והוא בוחר ביניהם בהסתברות $\frac{1}{2}$.
2. השאלה השניה - פער השלמות במקרה הזה הוא $\frac{3}{4}$. כלומר:

$$\frac{OPT\left(I\right)}{OPT_F\left(I\right)}$$

בדר"כ במינימיזציה אמרו שזה לפחות משהו, כאן, בבעיית מקסימיזציה, יש להראות כי קיים I כך ש:

$$\frac{OPT\left(I\right)}{OPT_F\left(I\right)}\leq\frac{3}{4}$$

נמצא כזה:

$$f=\left(x_1\vee x_2\right)\wedge\left(\overline{x}_1\vee x_2\right)\wedge\left(x_1\vee\overline{x}_2\right)\wedge\left(\overline{x}_1\vee\overline{x}_2\right)$$

כאשר המשקולות כולן 1. אזי $OPT\left(I\right)=3$, הפתרון השברי נותן $x_1=x_2=\frac{1}{2}$, ולכן $OPT_F\left(I\right)=4$, ואז מתקבל היחס שטענו.

3. יש אלגוריתם יותר טוב (נזכיר שאנו דנים אך ורק באלגוריתמים פולינומיאלים) עם יחס קירוב $>\frac{3}{4}$. נראה את הכלי שמאפשר לעשות את זה.

1.3.4 דרנדומיזציה

נרצה להפוך את האלגוריתם 8 לאלגוריתם דטרמיניסטי. מה שנרצה לעשות - במקום להגריל בין שני האלגוריתמים, לבחור ביניהם בצורה חכמה.

נזכיר שכשדיברנו על בעיות מינימיזציה, היה לנו "טריק נחמד" להפוך את התוצאה של התוחלת להסתברות גבוהה - בעזרת א"ש מרקוב עבור $t > 1$:

$$Pr\left[X \geq t \cdot \mathbb{E}\left[x\right]\right] \leq \frac{1}{t}$$
$$\mathbb{E}\left[A\left(I\right)\right] \leq \alpha \cdot OPT\left(I\right)$$
$$\Rightarrow Pr\left[A\left(I\right) > t \cdot \alpha OPT\left(I\right)\right] \leq \frac{1}{t}$$

ואז בהסתברות $1 - \frac{1}{t}$, $A\left(I\right) \leq t \cdot \alpha \cdot OPT\left(I\right)$. אפשר לחזור על זה הרבה פעמים ולקבל תוצאה כרצוננו.

בבעיות מקסימיזציה צריך לעשות משהו קצת שונה (כמו בתרגיל 1 שאלה 4) - בעוד שבמינימיזציה אפשר תמיד לעשות את התהליך הזה, בבעיות מקסימיזציה יש תנאי.

נסתכל על תהליך ההחלטות שאנחנו עושים כתהליך סדרתי, שבו לאט לאט קובעים את ההשמה לחלק מהמשתנים:

- בשלה הראשון: השמה ל-x_1 (נקבע ערך 0 או 1), נסמן ערך זה ב-a_1.
- בשלב השני: השמה ל-x_2, נסמן ערך זה ב-a_2.
- וכך הלאה...
- בשלב ה-l: השמה ל-x_l, נזמן ערך זה ב-a_l.

בכל שלב שכזה קיימת לנו השמה חלקית כלשהיא - בשלב ה-l, השמה זו היא $\nu = (a_1, a_2, ..., a_l)$.

נחשב את תוחלת רווח האלגוריתם בהנתן שכבר בחרנו את ההשמה ν :

$$\mathbb{E}_\nu = \mathbb{E}\left[A\left(I\right)|\nu : x_1 = a_1, x_2 = a_2, ..., x_l = a_l\right]$$

מתחילים כש-$\nu = (\phi)$. בשלב ה-$l \geq 0$ נגדיר:

$$\nu_0 = (a_1, a_2, ..., a_l, 0)\,,\ \nu_1 = (a_1, a_2, ..., a_l, 1)$$

ונבחר את הטוב מביניהם כלומר:

$$\nu' = \underset{i \in \{0,1\}}{argmax} \mathbb{E}_{\nu_i}$$

האפשרות לעשות את התהליך הזה תלויה ביכולת שלנו לחשב את התוחלת $\mathbb{E}_\nu$. נראה שאנחנו אכן יכולים לחשב את הערך הזה - לשם כך מספיקות ההערכות שכבר עשינו:

$$\mathbb{E}\left[A\left(I\right)\right] \geq \alpha_i$$
$$\mathbb{E}\left[A\left(I\right)\right] \geq \beta_i z_i$$

ראינו שההסתברות עבור האלגוריתם הראשון שפסוקית מסתפקת היא $1-\left(\frac{1}{2}\right)^{k_j}$. אם משתנה x_1 מופיע בפסוקית, או שההשמה שנתנו לו קובעת שהפסוקית לא יכולה להסתפק, או שלא, ואז יש לה סיכוי $1-\left(\frac{1}{2}\right)^{k_j-1}$ להסתפק.

כך לכל פסוקית (בשלב ה־l) אפשר לחשב:

$$\mathbb{E}\left[c_j|\nu\right]=1-2\left(\frac{1}{2}\right)^{k_j-l}=1-\prod_{i\in P_j-\{1,\ldots,n\}}(1-y_i)\prod_{i\in N_j-\{1,\ldots,l\}}y_i$$

נשאל, מדוע האלגוריתם הזה טוב לפחות כמו שני הרנדומיים? האלגוריתם הרנדומי בוחר בהסתברות p_0 את הראשון, וב־p_1 את השני. אזי:

$$\begin{aligned}&\mathbb{E}_\nu=p_0\cdot\mathbb{E}_{\nu_0}+p_1\mathbb{E}_{\nu_1}\leq max\left\{\mathbb{E}_{\nu_0},\mathbb{E}_{\nu_1}\right\}\\&\mathbb{E}_{(\phi)}=\mathbb{E}\left[RAND\left(I\right)\right]\leq\mathbb{E}_\nu\ \forall\nu\\&\vdots\\&\nu_n=(a_1,...,a_n)\end{aligned}$$

בסוף מצאנו השמה מסויימת לכל n המשתנים בה אין רנדומיות, כלומר $\mathbb{E}_{\nu_n}$ הוא פשוט ערך הנוסחא להשמה ν_n, וערכה לפחות התוחלת של הרנדומי.

1.4 בעיות נוספות

23/4/2012

1.4.1 הקדמה - קצת על הסתברות

משפט 1.36 (אי שיוויון מרקוב) נתון משתנה מקרי אי שלילי $X:\Omega\to\mathbb{R}^+$ בעל תוחלת $\mathbb{E}\left[X\right]=\mu$. אזי לכל $t\geq 1$:

$$Pr\left[X\geq t\cdot\mu\right]\leq\frac{1}{t}$$

הוכחה:

$$\mu=\mathbb{E}\left[X\right]\geq Pr\left[X\geq t\mu\right]\cdot t\mu$$

■

ניסוח נוסף לא"ש מרקוב:

$$Pr\left[X'>S\right]\leq\frac{\mathbb{E}\left[X'\right]}{S}$$

הגדרה 1.37

$$\begin{aligned}&Var\left[X\right]=\mathbb{E}\left[\left(X-\mathbb{E}\left[X\right]\right)^2\right]\\&\sigma\left(X\right)=\sqrt{Var\left[X\right]}\end{aligned}$$

משפט 1.38 (אי שיוויון צ'בישב) באותם תנאים של א"ש מרקוב:

$$Pr\left[\left|X-\mathbb{E}\left[X\right]\right|\geq t\sigma\left(X\right)\right]\leq\frac{1}{t^{2}}$$

הוכחה: נסמן:

$$\begin{aligned}&X'=\left(X-\mathbb{E}\left[X\right]\right)^{2},\ t'=t^{2}\\&\Rightarrow Var\left[X\right]=\mathbb{E}\left[X'\right]\end{aligned}$$

נציב באי"ש מרקוב ונקבל:

$$Pr\left[\overbrace{\left(X-\mathbb{E}\left(X\right)\right)^{2}\geq t^{2}\cdot Var\left[X\right]}^{\left|X-\mathbb{E}[X]\right|\geq t\sigma(X)}\right]\leq\frac{1}{t^{2}}$$

■

אלו הם שני אי שיוויונות המדברים על ההתפלגות סביב הממוצע. כל זה זו הקדמה לאי שיוויון צ'רנוף $(Chernoff)$:

משפט 1.39 (אי שיוויון צ'רנוף) יהיו $X_1,...,X_n$ מ"מ בלתי תלויים מציינים (ערכים ב־$\{0,1\}$), יהי $X=\sum_{i=1}^{n}X_i$, ונסמן $\mu=\mathbb{E}\left[X\right]$. אזי:

$$\forall\delta>0:\ Pr\left[X\geq\left(1+\delta\right)\mu\right]\leq\left(\frac{e^{\delta}}{\left(1+\delta\right)^{1+\delta}}\right)^{\mu}\tag{1}$$

$$\forall0<\delta<1:\ Pr\left[X\leq\left(1-\delta\right)\mu\right]\leq\left(\frac{e^{\delta}}{\left(1-\delta\right)^{1-\delta}}\right)^{\mu}\tag{2}$$

מסקנה 1.40 יהיו $X_1,...,X_n$ מ"מ בלתי תלויים מצויינם, יהי $X=\sum_{i=1}^{n}X_i$, ונסמן $\mu=\mathbb{E}\left[X\right]$ כך ש־$\mu\leq M$. אזי:

$$\forall\delta>0:\ Pr\left[X\geq\left(1+\delta\right)M\right]\leq\left(\frac{e^{\delta}}{\left(1+\delta\right)^{1-\delta}}\right)^{M}\tag{3}$$

$$\forall0<\delta<1:\ Pr\left[X\leq\left(1-\delta\right)M\right]\leq\left(\frac{e^{\delta}}{\left(1+\delta'\right)^{1+\delta'}}\right)^{M}\tag{4}$$

נוכיח את המסקנה: **הוכחה:** 1. נגדיר את δ' קבוע המקיים $\left(1+\delta\right)M=\left(1+\delta'\right)\mu$. אזי, לפי א"ש צ'רנוף:

$$Pr\left[X\geq\left(1+\delta'\right)\mu\right]\leq\left(\frac{e^{\delta'}}{\left(1+\delta'\right)^{1-\delta'}}\right)^{\mu}$$

נסמן $x = \frac{M}{\mu}$. אזי:

$$\frac{e^{\delta'}}{(1+\delta')^{1+\delta'}} = \frac{e^{(1+\delta)x-1}}{((1+\delta)\,x)^{(1+\delta)x}}$$

נטען כי הביטוי מעלה $\geq \left(\frac{e^{\delta}}{(1+\delta')^{1+\delta'}}\right)^{x}$, ובכך נסיים את הוכחת מסקנה 1.
על ידי מניפולציות אלגבריות על אי השיוויון הזה, נקבל כי הוא שווה ערך ל:

$$\frac{e^{x}}{x^{(1+\delta)x}} \leq e$$

אי שיוויון זה נכון לכל $x \geq 1$, כיוון ו־$e^{x} \leq \left(\frac{e}{x}\right)^{x}$, כאשר אי השיוויון הזה נובע מכך שפונקציה זו מונוטונית יורדת בטווח המבוקש.
2. חלק זה של המסקנה נובע ישירות מא"ש צרנוף, שכן:

$$Pr\left[X \leq (1-\delta)\,M\right] \leq Pr\left[X \leq (1-\delta)\,\mu\right] \leq \left(\frac{e^{\delta}}{(1-\delta)^{1-\delta}}\right) \leq \left(\frac{e^{\delta}}{(1-\delta)^{1-\delta}}\right)^{M}$$

■

נביא סקיצה של ההוכחה של א"ש צ'רנוף: **הוכחה:** נביא סקיצה של ההוכחה:
ההוכחה של שני החלקים דומה מאוד, לכן נעבור רק על ההוכחה של אחד מהם.
נסמן $a = (1+\delta)\,\mu$, ויהי $t > 0$.

$$Pr\left[X \geq a\right] = Pr\left[e^{tX} \geq e^{ta}\right] \overset{\text{Markov},X'=e^{tX},\ t'=\frac{e^{ta}}{\mathbb{E}[X']},\ s=e^{ta}}{\leq} \frac{\mathbb{E}\left[e^{tX}\right]}{e^{ta}}\ (*)$$

$$\mathbb{E}\left[e^{tX}\right] = \mathbb{E}\left[e^{t\sum_{i=1}^{n} X_i}\right] = \mathbb{E}\left[\prod_{i=1}^{n} e^{tX_i}\right] \overset{independance}{=} \prod_{i=1}^{n} \mathbb{E}\left[e^{tX_i}\right]$$

כלומר מספיק להעריך כל מ"מ X_i לחוד כדי לחשב את $(*)$. נעשה זאת כעת:
נגדיר $P_i = Pr\left[X_i = 1\right]$. על כן $\mathbb{E}\left[X_i\right] = P_i$, כן:

$$\mu = \sum_{i=1}^{n} \mathbb{E}\left[X_i\right] = \sum_{i=1}^{n} P_i$$

$$\Rightarrow \mathbb{E}\left[e^{tX_i}\right] = P_i \cdot e^{t} + (1-P_i)\cdot e^{0} = 1 + P_i\left(e^{t}-1\right) \overset{1+x\leq e^{tx}}{\leq} e^{P_i\left(e^{t}-1\right)}$$

$$\Rightarrow \mathbb{E}\left[e^{tx}\right] = \prod_{i=1}^{n} \mathbb{E}\left[e^{tX_i}\right] \leq \prod_{i=1}^{n} e^{P_i\left(e^{t}-1\right)} \leq e^{\sum_{i=1}^{n}\left(e^{t}-1\right)} = e^{\mu\left(e^{t}-1\right)}$$

$$\Rightarrow Pr\left[X \geq a\right] \leq \frac{\mathbb{E}\left[e^{tX}\right]}{e^{ta}} \leq \frac{e^{\mu\left(e^{t}-1\right)}}{e^{t(1+\delta)\mu}} = e^{\mu\left(e^{t}-1-(1+\delta)t\right)^{2}} \overset{?}{\leq} \left(\frac{e^{\delta}}{(1+\delta)^{1+\delta}}\right)^{\mu}$$

נרצה למצוא את t שמביא את המינימום. כשעושים את זה (גזירה כדי למצוא את האופטימלי) מקבלים את מה שרצינו.
באותה דרך אפשר להוכיח את הסעיף השני. ■

"משחק קטן" רק כדי לראות איך א"ש צ'רנוף עוזר לנו:
במקרה שבהן ההתפלגויות זהות, כמו נגיד במיליון הטלות מטבע, אנו מצפים שבחצי מההטלות יצא עץ. אפשר לשאול מהי ההסתברות שיצא עץ $450,000$ פעמים (ולא מחצית מהפעמים). נבחר למשל $\delta = \frac{1}{10}$, אם נציב את המספרים בהרחבה של סעיף 1) נקבל סיכוי קטן מאוד להתרחשות הזו:

$$e^{-\frac{1}{100}\cdot 500{,}000\cdot\frac{1}{3}} \leq e^{-500}$$

Min Capacity Multi Commodity Flow 1.4.2

נתון גרף $G=(V,E)$, ו־k זוגות של קודקודים ב־V s_i, t_i, $i = 1, ..., k$, $n = |V|$.
רוצים למצוא לכל i מסלול בין s_i ל־t_i (שנעביר בו זרימה), ולהביא למינימום את העומס (=מספר המסלולים העוברים דרך הקשת) המקסימלי על קשת.

נעשה פישוט של הבעיה לצורך הדיון הנוכחי:
נניח ש־P_i היא רשימת מסלולים פשוטים בין s_i ל־t_i, ונחשוב עליה כפולינומית.

נרצה להפוך את הבעיה ל־LP. נעשה זאת בצורה הבאה:
לכל מסלול אפשרי $p \in P_i$ (עבור i כלשהוא) נגדיר משתנה $x_p \in \{0,1\}$: $x_p = 1$ אמ"מ מעבירים זרימה במסלול. אזי הבעיה היא:

$$\begin{aligned}
&Min\ W \\
&\forall 1 \leq i \leq k: \sum_{p \in P_i} x_p = 1 \\
&\forall e \in E: \sum_{p: p \ni e} x_P \leq W \\
&\forall p \in P_i,\ 1 \leq i \leq k: \ x_p \in \{0,1\}
\end{aligned}$$

נהפוך את ה־ILP מעלה ל־LP ע"י הרלקסציה ש־$0 \leq x_p \leq 1$.

נשים לב שלמרות שאנו מניחים את ש־P_i פולינומית, עדיין אפשר לפתור את הבעיה אם P_i אקספוננציאלית, כי מספר האילוצים שלנו (השורה השניה והשלישית) הוא פולינומיאלי, ואז שנהפוך לתוכנית הדואלית נקבל מספר פולינומיאלי של משתנים, ולכן המערכת פתירה. לכן במקרים מהסוג הזה תחת תנאי קל אפשר לפתור את זה.

כיצד נפתור את הבעיה שלנו? בעזרת עיגול רנדומי.

אלגוריתם 9 עיגול רנדומי ל־Min Capacity Multi Commodity Flow

- נחשב את הפתרון האופטימלי ל־LP - נסמן פתרון זה ב־x^*.
- לכל $i \leq i \leq k$:
  - נגדיל מסלול אחד p מ־P_i לפי ההתפלגות: p נבחר בהסתברות x^*_p.

האלגוריתם מאוד פשוט - נטען שהביצועים שלו מאוד טובים, ונוכיח זאת בעזרת א"ש צ'רנוף.

נשים לב כי מכיוון ו־$\sum_{p\in P_i} x_p^* = 1$, x_p^* אכן משרים התפלגות תקינה.

משפט 1.41 אם $W^* \geq 1$, אז בהסתברות גבוהה[5] העומס המקסימלי $\geq O(logn)\cdot W^*$, או במילים אחרות יש קירוב $O(logn)$.

משפט 1.42 לכל $\varepsilon > 0$, אם $W^* \geq \frac{12logn}{\varepsilon^2}$, אז בהסתברות גבוהה העומס המקסימלי $\geq (1+\varepsilon)W^*$, או במילים אחרות יש קירוב $1+\varepsilon$.

אלו בעצם שתי גרסאות של אותו החישוב, הנובע כמעט ישירות מצ'רנוף, כי "התוחלת שלנו היא טובה". נוכיח אם כך את שני המשפטים יחדיו: **הוכחה:** נגדיר לכל קשת $e \in E$ משתנה מקרי Y_e = מספר המסלולים העוברים דרך e, וכן נגדיר לכל $1 \leq i \leq k$ וקשת e משתנה $x_e^i = 1$ אם המסלול שנבחר בין s_i ל־t_i עובר דרך e. אזי:

$$Y_e = \sum_{i=1}^{k} x_e^i$$

$$\Rightarrow \mathbb{E}[Y_e] = \sum_{i=1}^{k} \mathbb{E}[x_i^e]$$

נשים לב כי:

$$\mathbb{E}[x_i^e] = \sum_{p\in P_i : p\ni e} x_p^*$$

ואז נקבל:

$$\mathbb{E}[Y_e] = \sum_{i=1}^{k} \mathbb{E}[x_i^e] = \sum_{i=1}^{k} \sum_{p\in P_i : p\ni e} x_p^* = \sum_{p:p\ni e} x_p^* \overset{LP}{\leq} W^*$$

נרצה להשתמש בצ'רנוף כדי להגיד ש"אנחנו לא יותר מידי רחוקים מהתוחלת הזו" לכל e. כאן $\mu = \mathbb{E}[Y_e]$. נבחר $\delta = (1+\varepsilon)\cdot\frac{W^*}{\mu} - 1$:

$$Pr[Y_e \geq (1+\varepsilon)\cdot W^*] = Pr\left[Y_e \geq \overbrace{\left((1+\varepsilon)\cdot\frac{W^*}{\mu}\right)}^{(1+\delta)}\mu\right] \overset{Chernoff}{\leq} e^{-\frac{\left[(1+\varepsilon)^2\left(\frac{W^*}{\mu}\right)-1\right]^2\mu}{3}}$$

$$\Rightarrow \left((1+\varepsilon)\cdot\frac{W^*}{\mu} - 1\right)^2\mu \overset{W^*\geq\mu}{\geq} \left(\varepsilon\cdot\frac{W^*}{\mu}\right)^2\mu \overset{W^*\geq\mu}{\geq} \varepsilon^2\cdot\left(\frac{W^*}{\mu}\right)\mu = \varepsilon^2 W^*$$

$$\Rightarrow Pr[Y_e \geq (1+\varepsilon)\cdot W^*] \leq e^{-\frac{\varepsilon^2 W^*}{3}}$$

[5] ניסוח מקובל במדעי המחשב אם כי מעט מעורפל - כאן נתייחס לכך שההסתברות לכישלון היא $< \frac{1}{n^c}$.
למעשה כאן ובמקרים הקודמים "אנחנו מקבלים את זה בחינם", כי אם נתון לנו כי ההסתברות קטנה מקבוע מסויים, אפשר כפי שראינו לחזור על הניסוי מספר l פעמים להקטין את ההסתברות לכישלון כרצוננו.
כמו כן אפשר לעשות מניפולציה נוספת - אפשר לבחור שאם לא הצלחנו לאחר מספר ניסויים מסויים, מריצים אלגוריתם אקפוננציאלי אחר.

מכאן נגיע לשני המשפטים ע"י הצבה:
עבור המשפט השני, אם נציב את W^* נקבל:

$$Pr\left[Y_e \geq (1+\varepsilon)\cdot W^*\right] \leq e^{-\frac{\varepsilon^2\left(\frac{12logn}{\varepsilon^2}\right)}{3}} \leq \frac{1}{n^4}$$

זה כאמור היה עבור קשת אחת. אזי, מחסם האיחוד (כאשר n^2 הוא מספר הקשתות, כפול ההסתברות שאחת מהן מפרה את התנאי):

$$Pr\left[\underset{e\in E}{max}Y_e \geq (1+\varepsilon)\,W^*\right] \leq n^2\cdot\frac{1}{n^4} = \frac{1}{n^2}$$

עבור המשפט הראשון, יש להשתמש בפישוט השני של צ'רנוף (במקרה ש-δ גדול):

$$\begin{aligned} Pr\left[Y_e \geq 4logn\cdot W^*\right] &= Pr\left[Y_e \geq 4logn\left(\frac{W^*}{\mu}\right)\mu\right] \\ &\overset{Chernoff}{\leq} e^{-4logn\left(\frac{W^*}{\mu}\right)\mu} = e^{-(4lognW^*)} \overset{W^*\geq 1}{\leq} \frac{1}{n^4} \\ \Rightarrow Pr\left[\underset{e\in E}{max}Y_e \geq 4lognW^*\right] &\leq n^2\cdot\frac{1}{n^4} = \frac{1}{n^2} \end{aligned}$$

■

Knapsack 1.4.3

נתון שק בגודל B, ו-n פריטים $S = \{a_1, ..., a_n\}$. לכל פריט גודל $size(a')$ ורווח $profit(a_i)$.
רוצים למצוא $S' \subseteq S$ כך ש: $\sum size(a_i) \leq B$ כך ש-$profit(S') = \sum_{a_i\in S'} profit(a_i)$ מקסימלי.
נניח כי $profit(a_i), size(a_i) \in \mathbb{Z}^+$, ואי אפשר לקחת חלק מפריט (או לוקחים אותו או לא לוקחים אותו כלל).

נסמן $P = \underset{i}{max}profit(a_i)$. נרצה למצוא אלגוריתם שרץ בזמן פולינומי ב-n וב-P.

אלגוריתם 10 אלגוריתם דינמי ל-Knapsack

- מחשבים לכל ערך $1 \leq p \leq nP$ (שלם), ולכל $1 \leq i \leq n$ $A(p,i)$ = המשקל המינימלי שנותן רווח כולל P עבור i הפריטים הראשונים. נחשב אותו בצורה הבאה:

$$A(p,i+1) = min\begin{cases} A(p,i) \\ A(p-profit(a_{i+1})) + size(a_{i+1}) \end{cases}$$

זמן הריצה של האלגוריתם - $O\left(n^2P\right)$.

נזכור כי $FPTAS$ = קיים קירוב $1+\varepsilon$ בזמן ריצה פולינומי ב־$n, \frac{1}{\varepsilon}$. נביא אלגוריתם $FPTAS$ לבעיה:

אלגוריתם 11 אלגוריתם $FPTAS$ ל־Knapsack

- נגדיר $k = \frac{\varepsilon P}{n}$.
- $\forall 1 \leq i \leq n : \ profit'(a_i) = \left\lfloor \frac{profit(a_i)}{k} \right\rfloor$ ־ נסמן instance זה ב־I'.
- נסמן $P' = \max_i profit'(a_i) = \left\lfloor \frac{P}{k} \right\rfloor = \left\lfloor \frac{P}{\frac{\varepsilon P}{n}} \right\rfloor = \left\lfloor \frac{n}{\varepsilon} \right\rfloor$.
- פותרים את הבעיה החדשה בעזרת אלגוריתם דינמי ־ נסמנו ALG'.

זמן הריצה של האלגוריתם הנ"ל ־ $O\left(n^2 P'\right) = O\left(\frac{n^3}{\varepsilon}\right)$.

טענה 1.43

$$ALG(I) \geq (1-\varepsilon) OPT(I)$$

הוכחה: נשים לב:

$$ALG(I) \geq k \cdot ALG'(I') = k' \cdot OPT'(I') \geq k \cdot \left(\frac{OPT(I)}{k} - n\right) = OPT(I) - nk \ (*)$$

כי יכול להיות שעל כל אחד מהאיברים אנחנו מפסידים 1 בעיגול. אזי:

$$(*) = OPT(I) - something \in P \overset{P \leq OPT(I)}{\geq} (1-\varepsilon) OPT(I)$$

■

1.5 Primal-Dual Algorithms

30/4/2012

ראינו את הרעיון הבסיסי כבר ־ היום נכנס לעומק ונראה דוגמאות נוספות.
בהנתן תוכנית לינארית בצורה הקנונית המיוחדת P:

$$\begin{aligned} &min\ c^T x \\ &Ax \geq b \\ &x \geq 0 \end{aligned}$$

התוכנית הדואלית D שלה תהיה:

$$\begin{aligned} &max\ b^T y \\ &A^T y \leq c \\ &y \geq 0 \end{aligned}$$

ראינו את עקרון העודף המשלים:

משפט 1.44 יהיו x, y פתרונות אפשריים ל־P, D בהתאמה. x, y הם פתרונות אופטימליים אמ"מ מתקיימים שני התנאים הבאים:

1. לכל $1 \leq i \leq n$, או $x_i = 0$ או $\sum_{1 \leq j \leq m} a_{ij} y_j = c_i$.
2. לכל $1 \leq j \leq m$, או $y_j = 0$ או $\sum_{1 \leq i \leq n} a_{ij} x_i = b_j$.

וכן ראינו את עקרון העודף המשלים המורחב:

משפט 1.45 יהיו x, y פתרונות ל־P, D בהתאמה (בצורה קנונית מיוחדת) המקיימים את התנאים הבאים:

1. לכל $1 \leq i \leq n$, או ש־$x_i = 0$ או $\sum_j a_{ij} y_j \geq \frac{c_i}{\alpha}$.
2. לכל $1 \leq j \leq m$, או ש־$y_j = 0$ או $\sum_j a_{ij} x_i \leq \beta \cdot b_j$.

כאשר $\alpha, \beta \geq 1$.
אזי:

$$c^T x \leq \alpha\beta \cdot b^T y$$

מסקנה 1.46 מדואליות חלשה:

$$\leq \alpha\beta \cdot b^T y* = \alpha\beta \cdot OPT_F(I) \leq \alpha\beta \cdot OPT(I)$$

Min Multi Cut 1.5.1

תחילה, הגדרות בסיסיות:

הגדרה 1.47 בהנתן קבוצה $S \subseteq V$, $\left(S, \overline{S}\right)$ הוא חתך לשתי קבוצות שמתאים ל־S את הקבוצה:

$$\{(u, v) \,|u \in S,\ v \notin S\}$$

בצורה דומה, אפשר להגדיר חתך למספר קבוצות:

הגדרה 1.48 תהי C קבוצה של קשתות. C היא חתך ל־n קבוצות אם היא מפרידה את הגרף ל־n חלקים $S_1, S_2, ..., S_n$ זרים. כלומר:

$$C = \{(u, v) \,|u \in S_i,\ v \in S_j,\ i \neq j\}$$

כעת, נגדיר את הבעיה:
נתון גרף $G=(V,E)$, וקבוצה של k זוגות:

$$(s_1,t_1),(s_2,t_2),...,(s_k,t_k)$$

כאשר לכל i: $s_i,t_i\in V$, ולכל קשת $e\ni E$ משויך משקל $c_e\ni\mathbb{R}^+$.
המטרה: רוצים למצוא חתך בגרף G כך שכל זוג (s_i,t_i) מופרד, בעל משקל מינימלי (כאשר משקל החתך הוא סה"כ משקל הצלעות שהוא מכיל).
נשים לב שאין כאן מגבלה על מספר החלקים שאפשר לחלק את הגרף אליהם, וכמו כן אין דרישה ש־s_i,t_i שונים זה מזה. לכן תמיד יש פתרון לבעיה (מקסימום פשוט מפרידים בין כל הזוגות, שכן כאמור אין מגבלה על מספר הקבוצות).

כאשר יש לפחות שני זוגות הבעיה הזו היא בעיה NP־קשה.
אנו נדבר היום רק על בעיה זו בעצים, כלומר כאשר G הוא עץ - בעץ כאמור יש מסלול אחד מכל קודקוד לקודקוד אחר, ולכן זהו איזשהוא פישוט של הבעיה. גם במקרה זה הבעיה היא קשה - אפשר לראות את זה די בקלות: בעיה זו כוללת בתוכה כמקרה פרטי את בעיית ה־Vertex Cover. מדוע?
נביט במקרה הפרטי של כוכב (ישנו קודקוד מיוחד באמצע, וממנו מסלול לכל כוכב).
ציור
בציור חדש, לכל קשת בציור הקודם נשים קודקוד חדש שיתאים לקשתות (כאשר נעביר את משקל הקשת להיות משקל הקודקוד המתאים), וכן נשים נשים קשתות בין הקודקודים רק כשיש מסלול בין (s_i,t_i):
ציור 2
אזי נקבל כי הבעיה בציור החדש היא בעיית Vertex Cover השקולה לבעיה המקורית.
זה קורה ספציפית בדוגמא הזו, אבל אנו צריכים למקרים כלליים את הניתוח של התוכנית הדואלית על מנת לפתור את הבעיה (אחרת היינו יכולים לפתור את הבעיה בצורה שפתרנו את בעיית ה־Vertex Cover).

נגדיר את הבעיה הפרימלית:
נתאים משתנה x_e שיציין האם הקשת בחתך. אזי, התוכנית הפרימלית P השלמה תהיה:

$$\begin{aligned}&min\ \sum_{e\in E}c_ex_e\\&\forall 1\le i\le k,\ \sum_{e\in P_i}x_e\ge 1\\&\forall e\in E\ x_e\in\{0,1\}\end{aligned}$$

כאשר P_i יהיה המסלול היחיד בין s_i ל־t_i.
הרלקסציה ל־LP תהיה:

$$\forall e\in E\ x_e\ge 0$$

ואכן קיבלנו תוכנית שקולה לזו בבעיית ה־VC.

התוכנית הדואלית D תהיה:

$$max \sum_{i=1}^{k} y_i$$
$$\forall e \in E : \sum_{i:e \in P_i} y_i \leq c_e$$
$$\forall 1 \leq u \leq k \; y_i \geq 0$$

אפשר לחשוב עליה כסוג של בעיית זרימה, כאשר y_i הן הזרימות דרך המסלול P_i (לא דרך קשת!), והדרישה הראשונה c_e היא המכסה הכוללת של הזרימה.
במקרה הפרטי של הכוכב, זה יראה כך:
ציור 3

לפי הצגת האולגוריתם, לכל s_i, t_i, מבין הקודקודים על המסלול s_i, t_i נגדיר את v_i להיות: $v_i = lca(s_i, t_i)$ (least common ancestor), הוא בעל העומק המינימלי.

אלגוריתם 12 Primal Dual for the Min Multi Cut problem

- אתחול: $\forall e \in E : \; x_e = 0$, $\forall i : \; y_i = 0$.
- מציאת חתך: נבחר באופן שרירותי קודקוד שיהיה השורש של העץ. זה ישרה סדר על העץ, ולכל קודקוד נוכל להגדיר את העומק (ללא קשר למשקל הצלעות):

$$depth(v) = \text{distance from the root}$$

נתבונן בקודקודים $v_1, ..., v_k$ לפי סדר לא עולה של העומק:

- נגדיל את הזרימה דרך P_i (כלומר נגדיל את y_i) עד שנגיע לרוויה. כלומר, עבור קשת כלשהיא $e \in P_i$ (יכול להיות יותר בבת אחת), האילוץ הדואלי יהיה הדוק.
- קובעים את $x_e = 1$ (כלומר $e \in C$) לכל $E \ni e$ שהגיעה לרוויה.

מה רע באלגוריתם הזה? יש בעיה במסלולים מאוד ארוכים. למשל, אם הגרף הוא מסלול אחד מאוד ארוך, וכל $c_e = 1$, כל הצלעות יכנסו בבת אחת לחתך. נוסיף שלב נוסף ע"מ לשפר הביצועים:

- "ניקוי": עוברים על הקשתות בסדר הפוך להכנסתן לחתך C, ומורידים קשת e אם $C \setminus \{e\}$ עדיין מהווה חתך חוקי (שמפריד את כל הזוגות).

משפט 1.49 לאלגוריתם מעלה על עצים יש יחס קירוב 2.

הוכחה: תחילה, לכל מסלול בחרנו לפחות קשת אחת לחתך, אחרת היינו ממשיכים בתהליך, לכן הפתרון חוקי.
שנית, האילוצים של התוכנית הדואלית D מתקיימים ישירות מהגדרת האלגוריתם (לא ממשיכים להגדיל את הזרימה אם אחת הקשתות הגיעה לרוויה).
נותר להראות כי התנאים של עקרון העודף המשלים המרוחב מתקיימים:

התנאי הראשון עם $\alpha = 1$ ברור כי מתקיים, ישירות מהגדרת האלגוריתם - אם $x_e \neq 0$, אז הגענו לרוויה בקשת e, וזה שקול לכך שהאילוץ הדואלי הדוק.
נראה כי תנאי 2 מתקיים עם $\beta = 2$:

טענה 1.50 יהיו (s_i, t_i) זוג כך ש-$y_i \neq 0$ (יש להראות כי במקרה הזה התנאי של התוכנית הפרימלית מתקיים עד כדי β), ויהי $v_i = lca(s_i, t_i)$; אזי יש לכל היותר קשת אחת ב-C על המסלול בין v_i ל-s_i, ובדומה לכל היותר קשת אחת בין v_i ל-t_i , כלומר במסלול בין s_i ל-t_i יש לכל היותר 2 צלעות בחתך.

נזכר כי התנאי בתוכנית הפרימלית היה:

$$\sum_{e \in P_i} x_e \geq 1$$

כדי להוכיח את תנאי 2 יש להראות כי אם $y_i \neq 0$, אזי:

$$\sum_{e \in P_i} x_e \leq 2$$

לכן אם הטענה נכונה, התנאי מתקיים.
נוכיח את הטענה: **הוכחה:** נוכיח עבור המסלול בין s_i ל-v_i, ההוכחה שקולה עבור המסלול בין t_i ל-v_i.
נניח בשלילה שקיימות e, e' צלעות בחתך שהן במסלול בין s_i ל-v_i, כלומר שתיהן "שרדו" את הניקוי. (התיקון: לא צריך להניח שe נכנסת אחרי e', פשוט לבחור את הסדר שלהן, וזה אמור להסתדר - יתוקן בקרוב לפי הספר).
אם e נכנסה אחרי e', היא נבדקת לפני e', לכן e נמצאת על מסלול נוסף בין s_j ל-t_j (מסלול שהיא הכרחית לגביו), וישנו $v_j = lca(s_j, t_j)$. לכן e' לא על המסלול v_j. אז $depth(v_j) > depth(v_i)$, אז מסתכלים על v_j לפני שהסתכלנו על v_i, לכן לפני ש- e' נבחרה, כלומר לפני ש-e נבחרה, כלומר ישנה קשת אחרת e'' שהגיעה לרוויה והוספה בשלב הזה, אבל אז אין צורך ב-e, בסתירה לכך שהיא הכרחית על המסלול הזה.
אם הצלעות נכנסו באותו הזמן, אפשר להחליף את תפקידיהם במידת הצורך כך ש-e תהיה הנמוכה מביניהן, ואז מה שכתבנו מעלה נכון. ■

■

1.5.2 Minimum Knapsack

נתונה קבוצת איברים $I = \{1, ..., n\}$, ולכל איבר i נתון גודל $s_i \ni \mathbb{R}^+$ ורווח $v_i \ni \mathbb{R}^+$.
כאן הבעיה מעט שונה ממה שהכרנו - יש לנו דרישת רווח D, ונרצה למצוא $X \subseteq I$ המקיימת את התנאי $\sum_{i \in X} v_i \geq D$ כך ש-$\sum_{i \in X} s_i$ מינימלי.
נסמן $x_i = 1$ אם $i \in X$. בעיית ה-ILP נראית כך:

$$min \sum_{i \in I} s_i x_i$$
$$\sum_{i \in I} v_i x_i \geq D$$
$$\forall i \in I : \ x \in \{0, 1\}$$

הרלקסציה ל־LP:

$$\forall i \in I \; 0 \leq x_i \leq 1$$

עד כאן אנחנו מכירים ־ החידוש: נראה כי התוכנית האינטואיטיבית הזו לא טובה לבעיה. כלומר, פער השלמות גדול ־ הוא D. בשביל זה, נראה דוגמא:

$$I = \{1, 2\}, \; s_1 = 0, \; s_2 = 1$$
$$v_1 = D - 1, \; v_2 = D$$

נרצה להראות כי:

$$\frac{OPT\left(\mathcal{I}\right)}{OPT_F\left(\mathcal{I}\right)} \geq D$$

ב־$OPT\left(\mathcal{I}\right)$ אין מנוס מלקחת את האיבר השני ־ עלות פתרון זה הינה 1.
ב־$OPT_F\left(\mathcal{I}\right)$ אפשר לעשות את הדבר הבא:

$$x_1 = 1, \; x_2 = \frac{1}{D}$$
$$\Rightarrow \sum_{i \in I} v_i x_i = D - 1 + \frac{1}{D} \cdot D = D$$
$$\Rightarrow OPT_F\left(\mathcal{I}\right) = \frac{1}{D}$$
$$\Rightarrow \frac{OPT\left(\mathcal{I}\right)}{OPT_F\left(\mathcal{I}\right)} = \frac{1}{\frac{1}{D}} = D$$

אפשר לעשות שני דברים:

- לקחת תוכנית שאינה תוכנית לינארית. אפשרות זו נראה בהמשך.
- בשיעור הבא ניקח את התוכנית שלנו, ונשנה קצת את האילוצים כך שהגיוני שיתקיימו, אבל שונה במעט.

7/5/2012

נחזור ל־ILP לרגע[6], ונשאל את עצמנו, אילו עוד דרישות נוכל להכניס לתוכנה כך שפער השלמות יקטן? זוהי שאלה לא טריוויאלית בכלל.

נביט למשל על הדוגמא בה השתמשנו בשיעור הקודם ־ כיצד אפשר לשפר ביחס אליה?

תחילה ניקח את האיבר הראשון, העלות שלו קטנה ולכן זה כדאי. כעת נשאל, האם כדאי או לא כדאי להוסיף את האיבר השני, וכמה אעריך שזה יעלה לי?

אנחנו בעצם רוצים להגיד שההערכה $\frac{1}{D}$ אינה נכונה. למה? המטרה שלנו היא למצוא פתרון לא שברי. להוסיף D כשאנחנו רוצים להוסיף יחידה אחת זה לא הגיוני, כי הוא מוסיף לנו רק 1, הוא לא תורם יותר מזה. אזי בשלב הזה נרצה להסתכל על האיבר כאילו הוא נותן לנו בדיוק 1. כלומר, בכל שלב נרצה לבדוק מהי התועלת הנוכחית של שאר האיברים, בהתחשב במה שיש לנו כרגע בשק. במקרה שלנו, הערך החדש של האיבר השני בהנחה שלקחנו את הראשון הוא $v_2' = 1$.

נרשום בצורה מסודרת:

[6] אל דאגה, נחזור לתוכנית ה־LP בסוף, זה רק יעזור לנו בתרגיל המחשבתי הנוכחי

בהנחה ש־A קבוצת האיברים שכבר נלקחו, והרווח על הקבוצה $v(A) < D$, נוכל להגדיר את הדרישה על הקבוצה הזו להיות:

$$D_A = D - v(A)$$

בדוגמא שלנו, $A = \{1\}$, $v(A) = D - 1$, ולכן $D_A = 1$.

לכל איבר בשלב הזה נוכל להגדיר:

$$v_i^A = min\{v_i, D_A\}$$

בדוגמא שלנו, $v_2^A = 1$.

נעיר בשלב זה כי מספר הדרישות החדש שלנו יהיה אקספוננציאלי, אבל נתעלם מכך בשלב זה ־ נביט בתוכנית רק לצורך הניתוח, ובהמשך נראה איך נעקוף את הבעיה בעזרת פתרון ה־Primal-Dual.

נרשום את תוכנית ה־ILP החדשה:

$$\begin{aligned} &min \sum_{i \in I} s_i x_i \\ &\forall A \subseteq I : \sum_{i \in I \setminus A} v_i^A \cdot x_i \geq D_A \\ &\forall i \in I \; x_i \in \{0,1\} \end{aligned}$$

טענה 1.51 הבעיות שקולות.

הוכחה: בכיוון הראשון, נשים לב כי עבור הקבוצה הריקה בתוכנית החדשה אנו מקבלים את אותה הדרישה שמופיעה בתוכנית המקורית, לכן כל פתרון בבעיה החדשה הוא פתרון של הבעיה המקורית.

בכיוון השני, אם ניקח פתרון לתוכנית המקורית, מתקיים:

$$\sum_{i \in A} v_i x_i \leq v(A)$$

ולכן ברור שהרווח שנשאר על האיברים שאינם ב־A, מקיים:

$$\sum_{u \in I \setminus A} v_i x_i \geq D - v(A) = D_A$$

כלומר התנאים של התוכנית החדשה מתקיימים.

על כן הבעיות אכן שקולות, כנדרש. ■

הרלקסציה תהיה:

$$x_i \geq 0$$

אנחנו לא דורשים $x_i \leq 1$, כי בחירה שכזו "לא עוזרת" ־ האילוץ מתחשב ב־D_A שמניח שלקחנו עד 1 מכל איבר, ולכן "לקחת יותר" לא יעזור לתוצאה הסופית.

התוכנית המקורית היא סוג של "בעיית אריזה", ותוכנית ה־Primal-Dual המתאימה היא סוג של בעיית cover:

$$\begin{aligned} &max \sum_{A\subseteq I} D_A y_A \\ &\forall i \in I: \sum_{A\subseteq I,\ i\notin A} v_i^A y_A \leq s_i \\ &\forall A \subseteq I:\ y_A \geq 0 \end{aligned}$$

אלגוריתם 13 אלגוריתם Primal-Dual לבעיית ה־Min Knapsack

$y \leftarrow 0\,(\forall B \subseteq I:\ y_B := 0),\ A \leftarrow \phi,\ x_i = 0;$
$while\,(v\,(A) < D)\,\{$

Increase y_A until $\exists i \in I\backslash A: \overbrace{\sum_{B\subseteq I: i\notin B} v_i^B y_B = s_i}^{\text{Dual constraint}}$

$A := A \cup \{i\}$
$x_i = 1\}$

מדוע הריצה פולינומית?
נשים לב כי מספר ה־y_A שלא יהיו 0 תחומים ע"י n מסויים, שכן אנחנו מגדילים תמיד עבור קבוצות מסויימות - הקבוצה הריקה, הקבוצה שתכיל את האיבר הראשון, את האיבר הראשון והשני, וכן הלאה, עד מילוי השק עבור n איברים.
נדמה את ריצה האלגוריתם על הדוגמא שראינו:
בהתחלה $A = \phi$.
מכיוון ו־$s_1 = 0$ האילוץ מתקיים מיידית, ולכן $y_\phi = 0$ נשאר, ובוחרים $x_i = 1$, $i_1 = 1$, $A = \{1\}$.
בשלב הבא האילוץ של $i = 2$ מביטים על כל תתי הקבוצות שלא מכילות את 2, ישנן שתיים כאלו:

$$v_2^\phi y_\phi + v_2^{\{1\}} y_{\{1\}} \overbrace{=}^{\text{we want}} s_2 = 1$$

האיבר הראשון שווה לאפס (כבר טיפלנו ב־y_ϕ והוא נשאר 0), לכן נגדיל את $y_{\{1\}}$ עד שהאילוץ יתקיים, כלומר עד $y_{\{1\}} = 1$, $i_2 = 2$, וכן $A = \{1, 2\}$.
במקרה הספציפי הזה קיבלנו פתרון אופטימלי. ננתח עבור המקרה הכללי:

משפט 1.52 לאלגוריתם 1.5.2 יש יחס קירוב 2.

הוכחה: תחילה, בסוף האלגוריתם, נסמן $X \equiv A_{final} = \{i_1, i_2, ..., i_l\}$ קבוצת האיברים שנבחרו. אזי:

$$v(X) \geq D \Leftrightarrow \sum_{i \in I} v_i x_i \geq D$$

וכבר ראינו קודם כי צד ימין גורר את קיום כל האילוצים של P.

כמו כן, התנאים של האילוצים של התוכנית הדואלית D מתקיימים מתוך הגדרת האלגוריתם - לא מגדילים ערך y_A אם נחרוג מהאילוץ.

נותר להראות כי התנאים של עקרון העודף המשלים המורחב מתקיימים עם $\alpha = 1,\ \beta = 2$:

תנאי 1 מתקיים מהגדרת האלגוריתם:

$$\forall i \in I:\ x_i \neq 0 \Rightarrow \sum_{A \subseteq I : i \notin A} v_i^A y_A = s_i$$

שכן עצרנו כאשר האילוץ התמצה.

תנאי 2 אומר:

$$\forall A \subseteq I:\ y_A \neq 0 \Rightarrow \sum_{i \in I \setminus A} v_i^A x_i \leq 2 \cdot D_A$$

עבור $A = X$ זה ברור, שכן זהו השלב האחרון. נתבונן ב-$D > v(A)$ עבורו $y_A \neq 0$ ואינו X, כלומר בשלבי הביניים. ננתח:

$$\sum_{i \in I \setminus A} v_i^A x_i = \sum_{i \in X \setminus A} v_i^A = v_{i_l}^A + \sum_{i \in X \setminus A,\ i \neq i_l} v_i^A$$

עבור $i_l \neq i \in X \setminus A$:

$$\begin{aligned} &v_i < D - v(A) \Rightarrow v_i^A = min\{v_i, D_A\} \leq v_i \\ &\Rightarrow \sum_{i_l \neq i \in X \setminus A} v_i^A \leq \sum_{i_l \neq i \in X \setminus A} v_i = v(X \setminus \{i_l\}) - v(A) < D - v(A) = D_A \end{aligned}$$

עבור i_l האיבר האחרון שהוסף ל-X:

$$\begin{aligned} &v_{i_l}^A = min\{v_i, D_A\} \leq D_A \\ &\Rightarrow \sum_{i \in I \setminus A} v_i^A x_i \leq D_A + D_A = 2 \cdot D_A \end{aligned}$$

■

1.6 $Semi - Definite\ Programming$

כאן נראה שוב שהבעיה הלינארית הטריוויאלית לא עובדת, וגם כאן נצטרך לעבוד קצת יותר קשה.

נדון בבעיה הקלאסית לנושא זה:

Max Cut 1.6.1

נתון גרף ממושקל $G=(V,E,w)$, $w:E\to\mathbb{R}^+$.
רוצים למצוא חלוקה של הקודקודים לשתי קבוצות $S,\overline{S}=V\backslash S$ כך שמשקל החתך שנוצר:

$$\delta(S)=\left\{(u,v)\,|u\in S,\ v\in\overline{S}\right\}$$

יהיה מקסימלי:

$$C(S)=\sum_{e\in\delta(S)}w(e)$$

אנחנו יודעים (מהתרגיל) קירוב $\frac{1}{2}$ די פשוט לבעיה.
נשאל, האם יש קירוב טוב מכך?
בעזרת Linear Programming לא מצאו דרך יותר טובה, ואז בא רעיון אחר. קודם כל נראה כיצד פותרים, ואז נחזור אחורה ונבין את הרעיון.
בשלב הראשון, כותבים תוכנית המתארת את הבעיה אך <u>אינה</u> לינארית לבעיה - בעיה ריבועית:
לקודקוד $i\in V$ נגדיר משתנה $y_i=\{1,-1\}$ שיציין באיזה צד נמצא הקודקוד: $y_i=1$ אמ"מ $i\in S$, $y_i=-1$ אמ"מ $i\in\overline{S}$.
אזי:

$$y_iy_j=-1\Leftrightarrow(i,j)\in\delta(S)$$
$$y_iy_j=1\Leftrightarrow(i,j)\notin\delta(S)$$

נקבל את התוכנית הריבועית $Quadratic\ Programming$ הבאה:

$$max\ \sum_{i,j}w_{ij}\cdot\frac{1}{2}\cdot(1-y_iy_j)=C(S)$$
$$\forall i\in V:\ y_i^2=1$$

הבעיה הזו היא בלתי פתירה בזמן פולינומי, אפילו אם נבצע רלקסציה מהסוג שהכרנו (הרשאת שברים) - היא שקולה באופן מדוייק לבעיה של ה-$Max\ Cut$.
מה שנעשה הוא להביט ברלקסציה מסוג אחר ממה שהכרנו:

תוכנית וקטורית - VP הרעיון: מביטים בוקטורים דו מימדיים, על דרישה מהסוג $y_i^2=1$ שהיא דרישה דו מימדית ולא מספר.
נחליף אותה בדרישה:

$$v_i\cdot v_i=1$$

כלומר הנורמה היא 1. כל הנקודות על ספרת היחידה מקיימות את התנאי הזה - נבצע רלקסציה ונרשה את כל הוקטורים הללו.
זוהי דרך חדשה ויותר מורכבת להחליש את הדרישה (בתכנות לינארי פשוט הרשנו שברים וזה הספיק).

נקבל את תוכנית ה־VP הבאה:

$$
\begin{aligned}
&max \sum_{i \neq j} w_{ij} \cdot \frac{1}{2} \cdot (1 - v_i v_j) \\
&\forall i \in V : \ v_i \cdot v_i = 1 \\
&v_i \in \mathbb{R}^n
\end{aligned}
$$

בתקווה בשיעור הבא נראה כי n הוא מספר הנקודות.

טענה 1.53 הבעיה הריבועית ובעיית ה־VP שקולות.

זאת כי אם נצא מ־VP, ונתרגם את הוקטור $v_i = (y_i, 0, ..., 0)$, נקבל את התוכנית הריבועית.

Semi-Definite Programming

14/5/2012

הגדרה 1.54 מטריצה ממשית ריבועית A נקראת מוגדרת חיובית (positive semi definite - PSD) אם לכל $x \in \mathbb{R}^n$, $x^T A x \geq 0$.

תוכנית Semi-Definite היא תוכנית מהצורה הבאה:

$$
\begin{aligned}
&max/min \sum_{i,j} C_{ij} X_{ij} : \ X \in M_{n \times n}(\mathbb{R}) = (X_{ij})_{1 \leq i,j \leq n} \\
&\forall k \ \sum a_{ij,k} X_{ij} = b_k \\
&X = X^T \equiv X_{ij} = X_{ji} \\
&X \overset{PSD}{\geq} 0
\end{aligned}
$$

ניתן להגדירה גם כך:

$$
\begin{aligned}
&C \cdot X = \sum_{w} C_{ij} X_{ij} \\
&A_K \cdot X = b_k \\
&X = X^T \\
&X \geq 0
\end{aligned}
$$

טענה 1.55 תהי $X \in M_{n \times n}(\mathbb{R})$ ממשית סימטרית. אזי, התכונות הבאות שקולות:

1. כל הערכים העצמיים של X הם אי שליליים[7].
2. קיימת מטריצה $M \in M_{n \times n}(\mathbb{R})$ כך ש־$X = MM^T$.

[7] מימין או משמאל, מפני שהיא סימטרית זה אותו הדבר.

3. X היא PSD, כלומר:

$$\forall x \in \mathbb{R}^n,\ x^T X x \geq 0$$

הוכחה: $1 \Leftarrow 2$: עפ"י המשפט הספקטרלי למטריצות ממשיות סימטריות, האומר שכל מטריצה כזו היא לכסינה אורתוגונלית:

$$X = V \bigwedge V^T$$

כאשר $\bigwedge$ מטריצה אלכסונית, ו־$0 \neq \bigwedge_{ij} \geq 0$, ו־$V$ אורתוגונלית.
נגדיר D אלכסונית כך ש־$D_{ij} = \sqrt{\bigwedge_{ij}}$ ו־$M = VD$ $\left(DD^T = \bigwedge\right)$. אזי:

$$MM^T = VDD^T V^T = V \bigwedge V^T = X$$

$2 \Leftarrow 3$:

$$xXx^T = xMM^T x^T = \left\langle M^T x, M^T x \right\rangle = \left\| M^T x \right\| \geq 0$$

$3 \Leftarrow 1$: יהי v ו"ע עם ע"ע λ, אזי:

$$0 \leq v^T X v = v^t \lambda v = \lambda v^t v = \lambda \left\langle v, v \right\rangle = \lambda \left\| v \right\|^2 \Rightarrow \lambda \geq 0$$

■

נוכל לתרגם את $C_{ij}\left(v_i \cdot v_j\right)$ ע"י התכונה $X = MM^T$ כך ש:

$$M = \begin{pmatrix} \cdots & v_1 & \cdots \\ \cdots & v_2 & \cdots \\ & \vdots & \\ \cdots & v_n & \cdots \end{pmatrix}$$

ולכן $X_{ij} = v_i \cdot v_j$.

נחזור לדון ב־$MAX - CUT$. ראינו את תוכנית ה־$VP\,(SDP)$:

$$max \sum_{i<j} w_{ij} \cdot \frac{1}{2} \cdot (1 - v_i \cdot v_j)$$
$$\forall i \in V\ v_i \cdot v_i = 1 :\ v_i \in \mathbb{R}^n$$

ניתן למצוא פתרון ל־VP בזמן פולינומי (ב־QP קשה למצוא פתרון). הפתרון של ה־VP הינו n מימדי, וכל מימד הוא שברי. נרצה להמיר את הפתרון של ה־VP לפתרון מלא. נעשה זאת בעזרת אלגוריתם עיגול:

אלגוריתם 14 אלגוריתם עיגול ל־VP לבעיית ה־$MAX-CUT$

1. פתור את בעיית ה־$SDP \Leftrightarrow VP$, וקבל פתרון $\{v_i\}_{i=1}^n$.
2. הגרל וקטור $r \in \mathbb{R}^n$ בהתפלגות אחידה על ספרת היחידה ב־$\mathbb{R}^n$.
3. הפתרון לבעיית ה־$MAX-CUT$ יהיה הקבוצה:

$$S = \{ i \in V | \langle v_i, r \rangle \geq 0,\ v_i \in Sol\,(VP) \}$$

נשים לב:

$$v_i \cdot v_j = \|v_i\| \cdot \|v_j\| \cdot cos\theta_{ij} = cos\theta_{ij}$$
$$\frac{1}{2} \cdot (1 - v_i \cdot v_j) = \frac{1}{2} \cdot (1 - cos\theta_{ij})$$

נביט בהטלה של הוקטורים על מעגל היחידה (נוכל בקלות לבצע לאחר מכן הכללה ל־n מימדים):

איור

אם הוקטורים הם אנטיפודליים[8], התרומה שלהם לחתך תהיה 1, עבור זוויות קרובות ל־π נקבל פחות, על כן ניקח וקטורים אנטיפודלים ל־S.

r נורמל לעל־מישור מפריד של פתרון התוכנית, כאשר על המישור המפריד הזה הוא $\{ u | \langle u, v \rangle = 0 \}$ - כלומר, בחירת r קובעת את העל מישור. ניקח צד אחד של העל־מישור ל־S. נסמן ב־r' את ההיטל של r.

נעריך את איכות הפתרון של האלגוריתם:

$$\mathbb{E}\left[ALG\,(I)\right] = \sum_{i<j} w_{ij} \cdot Pr\left[(v_i, v_j) \in C\,(S)\right]$$
$$\mathbb{E}\left[\sum_{i<j} w_{ij} \cdot X_{ij}\right] = \sum_{i<j} w_{ij} \cdot \mathbb{E}\left[X_{ij}\right]$$

למה 1.56 לכל i, j:

$$Pr\left[(v_i, v_j) \in C\,(S)\right] = \frac{\theta_{ij}}{\pi}$$

כאשר $\theta_{ij} = arccos\,(v_i \cdot v_j)$.

משפט 1.57 לאלגוריתם יש יחס קירוב α, כלומר:

$$\mathbb{E}\left[ALG\,(I)\right] \leq \alpha \cdot \sum_{i<j} w_{ij} \cdot \frac{1}{2} \cdot (1 - v_i \cdot v_j)$$

[8] כלומר הזווית הקטנה בין הוקטורים היא $\theta_{ij} = \pi$.

$$\alpha = \min_{0\leq\theta\leq\pi} \frac{\frac{\theta}{\pi}}{\frac{1}{2}\cdot(1-cos\theta)} \approx 0.878...$$

הוכחה: נביט בזוג וקטורים v_i, v_j ובעל מישור H המכיל את שניהם (המישור המוגדר על ידם).

יהי r' ההיטל של r על H. ההתפלגות האחידה של r על ספרת היחידה משרה התפלגות אחידה של r' על מעגל היחידה.

על מנת ששני הוקטורים יהיו בחתך, על r' לחתוך את אחת הקשתות הקטנות הנוצרות על מעגל היחידה.

מהלמה, ההסתברות ש־r' נופל באחת הקשתות הללו הינה:

$$\frac{\theta_{ij}}{\pi} = \frac{2\theta_{ij}}{2\pi}$$

מכיוון ו־$\sum_{i<j} w_{ij}\cdot\frac{1}{2}\cdot(1-v_i\cdot v_j)$ הוא הפתרון של VP, ומטענה קודמת הוא שקול לתוכנית הריבועית, נקבל כי אכן לאלגוריתם ה־SDP יש את יחס הקירוב הנ"ל, כנדרש. ■

2 אלגוריתמים מקוונים - Online Algorithms

2.1 הגדרות

28/5/2012

[9]ללא קשר לאינטרנט - אלו הם אלגוריתמים המתמודדים עם חלק מהמידע ומה שידוע בזמן ריצה.

בעיות במערכות הפעלה - דפדוף למשל.

בעיות מהחיים - מסחר במניות, הזמנת מוניות, וכו'.

זהו נסיון של המחשב לבצע אנליזה לבעיות מהסוג הזה. יש עוד מודלים לעניין זה - הם הסתברותיים. המודל שאנו דנים בו מתייחס ל"מקרה הגרוע ביותר".

באופן בלתי פורמלי, בדר"כ האלגוריתם מקבל סדרת בקשות באופן סדרתי:

$$\sigma_1, \sigma_2, ..., \sigma_n$$

כלומר בשלב ה-i האלגוריתם קיבל את הבקשות $\sigma_1, ..., \sigma_i$, והוא צריך להוציא פלט a_i בזמן a_i:

$$f_i(\sigma_1, \sigma_2, ..., \sigma_n) = a_i$$

דהיינו מוציא פלט על סמך העבר בלבד.

נביא הגדרה פורמלית שמתאימה לבעיות מסויימות (תזמון, דפדוף וכו'):

הגדרה 2.1 נתונה קבוצה של בקשות אפשריות R וקבוצה של תשובות אפשריות A.

מקבלים סדרת בקשות - נסמן כי בשלב ה-i מקבלים את הבקשה $\sigma_i \in R$, כאשר בשלב זה התקבלה הסדרה:

$$\sigma^{(i)} = \sigma_1\sigma_2...\sigma_i \ \left(\sigma^{(0)} = \phi\right)$$

בשלב ה-i צריך לתת תשובה $a_i \in A$, כאשר $a^{(i)} = a_1a_2...a_i$ היא סדרת התשובות בשלב ה-i.

האלגוריתם הוא סדרה של פונקציות:

$$f_i : R^i \to A, \ f_i\left(\sigma^{(i)}\right) = a_i$$

כלומר בכל שלב, הפונקציה נותנת תשובה לבקשה ה-i.

לכל סדרת בקשות $\sigma^{(i)}$ ותשובות $a^{(i)}$ משוייך מחיר (מינימיזציה)\רווח (מקסימיזציה):

$$cost : R^i \times A^i \to \mathbb{R}$$

ומחיר האלגוריתם על פני סדרת בקשות הינו:

$$(ONLINE =)\, ON\left(\sigma^{(i)}\right) = cost\left(\sigma^{(i)}, a^{(i)}\right)$$

ד

[9]כאן מכיל חומר מהשיעור הקודם ומהשיעור ב-28/5 בו המרצה חזר ועדכן פורמלית את ההגדרות.

אופן הניתוח תמיד יהיה בהשוואה לאלגוריתם האופטימלי הלא־מקוון אשר יש לו את כל הידע (את כל סדרת הבקשות).

הגדרה 2.2 המחיר האופטימלי על סדרת בקשות הינו:

$$OPT\left(\sigma^{(i)}\right) = \inf_{b^{(i)} \in A^i} cost\left(\sigma^{(i)}, b^{(i)}\right)$$

אנו מנתחים את טיב האלגוריתם לפי יחס תחרותיות (competitive ratio):

הגדרה 2.3 נגיד שיש לאלגוריתם יחס תחרותיות חזק c אם:

$$\forall \sigma : \ ON\left(\sigma\right) \leq c \cdot OPT\left(\sigma\right)$$

כאשר σ היא סדרת בקשות.

עבור סדרות אינסופיות (או מאוד ארוכות), מגדירים מדד חלש יותר:

הגדרה 2.4 אם $\exists \sigma \ s.t. OPT\left(\sigma\right) \to \infty$, נגיד שיש לאלגוריתם יחס תחרותיות חלש c, כאשר:

$$c = \sup_{\sigma : OPT(\sigma) \to \infty} \frac{ON\left(\sigma\right)}{OPT\left(\sigma\right)}$$

ובאופן שקול:

$$\exists a \in \mathbb{R}^+ \ s.t. \ \forall \sigma : \ ON\left(\sigma\right) \leq c \cdot OPT\left(\sigma\right) + a$$

2.2 דוגמאות לאלגוריתמים דטרמיניסטיים

2.2.1 בעיית איזון עומסים - Load Balancing

או בשם אחר - makespan minimization:

נתונות m מכונות חישוב, וסדרת תהליכים, כאשר לתהליך ה־i משוייך מדד עומס $0 < a_i$, והמטרה היא להחליט על השמה של כל תהליך i לאחת המכונות j.

נסמן ב־l_j^t את עומס המכונה j בשלב t = סכום עומסי התהליכים שהושמו למכונה. למעשה מכיוון ובכל נקודת זמן t מתקבלת עבודה i כלשהיא, אפשר להסתכל על t כאחד ה־i־אים.

כמו כן, נגדיר:

$$\text{makespan in time t} := \max_j l_j^t$$

המטרה: למצוא השמה בעלת $makespan$ מינימלי (מינימלי מכל המקסימומים של l_j^t).

כאן אנו מניחים כי כל המכונות הן זהות. ניתן לדון גם בבעיה בה המכונות שונות, למשל מכונה אחת מהירה פי שתיים מאחרת, וכו'. יתכן ונדון בהרחבה זו בהמשך.

משפט 2.5 לאלגוריתם החמדן (מקוון - בוחר מכונה הכי פחות עמוסה ומקצה לה תהליך, נקרא גם $LIST$) יש יחס תחרותיות חזק $2 - \frac{1}{m}$.

משפט 2.6 עבור $m = 2$, יש חסם תחתון של $\frac{3}{2}$ על יחס התחרותיות.
כלומר, קיים σ כך שלכל אלגוריתם מקוון ALG מתקיים:

$$ALG(\sigma) \geq \frac{3}{2} \cdot OPT(\sigma)$$

עבור $m = 3$, קיים חסם תחתון של $\frac{4}{3}$.

הערה 2.7 לערך m גדול יותר, קיים חסם תחתון אך הוא אינו הדוק, בניגוד לשניים הקודמים.
היום ידוע כי עבור $m \to \infty$, יודעים היום כי (חסם עליון - כלומר קיים אלגוריתם טוב מהחמדן) $1.92 \leq$יחס תחרותיות (של הבעיה) ≤ 1.85 (קיים חסם תחתון - כלומר אין אלגוריתם עם יחס תחרותיות < 1.85).

זהו משפט אבסולוטי ללא קשר לסיבוכיות. **הוכחה:** עבור $m = 2$: יש להראות כי לכל אלגוריתם אונליין ON, קיימת סדרת בקשות σ כך ש:

$$ON(\sigma) \geq c \cdot OPT(\sigma)$$

כאשר במקרה שלנו נרצה להוכיח עבור $c = \frac{3}{2}$.
במקרה הזה מספיקות שתי הסדרות (אחת ממשיכה את השניה):

$$\sigma^{(2)} = 1, 1$$
$$\sigma^{(3)} = 1, 1, 2$$

כאן נכיר צורת חשיבה מקובלת במדעי המחשב - אפשר לחשוב על האלגוריתם שלנו כמתחרה מול "יריב", שאיננו רק אלגוריתם, הוא גם מייצר את הסדרה.
עבור הבקשה הראשונה אין חשיבות למכונה שנבחרה (שהרי הן זהות).
בשלב השני $\left(\sigma = \sigma^{(2)}\right)$ יש שתי אופציות:

- אם אלגוריתם ON בחר לשים את שני התהליכים על אותה המכונה, אז סיימנו, כי אז:

$$makespan = 2,\ OPT(\sigma) = 1$$

ואז $\frac{OPT(\sigma)}{ON(\sigma)} > \frac{3}{2}$ כפי שרצינו להוכיח.

- אחרת נביט בשלב שלוש $\left(\sigma = \sigma^{(3)}\right)$. בכל מקרה האלגוריתם ישים את התהליך השלישי במכונה שיש בה כבר תהליך אחד, ואז:

$$ON(\sigma) = 3,\ OPT(\sigma) = 2$$

שכן האלגוריתם האופטימלי יכול היה לשים במכונה אחת את שתי המכונות בעלות עלות 1, ובמכונה השניה את התהליך בעלות 2.

הערה 2.8 עבור אלגוריתם רנדומי הניתוח הנ"ל לא יתפוס.

■

נוכיח את משפט 2.5: **הוכחה:** נוכיח באינדוקציה על אורך הסדרה n:

$$ON\left(\sigma^{(n)}\right) \leq \left(2 - \frac{1}{m}\right) \cdot OPT\left(\sigma^{(n)}\right)$$

נניח שמתקיים עבור $n-1$.

- מקרה א' - ה-$makespan$ לא השתנה, כלומר $ON\left(\sigma^{(n)}\right) = ON\left(\sigma^{(n-1)}\right)$, ואז הטענה נובעת מהנחת האינדוקציה.
- מקרה ב' - ה-$makespan$ גדל[10], כלומר (שכן ON חמדן):

$$ON\left(\sigma^{(n)}\right) = \min_j l_j^{n-1} + a_n$$

כאן בדומה לאלגוריתמי קירוב, יש להגיד דברים על OPT. מתקיים:

$$OPT\left(\sigma^{(n)}\right) \overset{(*)}{\geq} a_n$$

כי עלות האלגוריתם האופטימלי גדול (או שווה) מעלות מכל משימה (שכן כל משימה מוקצה למכונה כלשהיא). כמו כן:

$$\max_j l_j^n = OPT\left(\sigma^{(n)}\right) \geq \frac{\sum_{j=1}^m l_j^n}{m} \overset{(**)}{=} \frac{\sum_{i=1}^n a_i}{m}$$

וזו ישירות מההגדרות - $(**)$ נובע מכך שהממוצע על העומסים שווה לממוצע על עלות העבודות. אזי:

$$\begin{aligned} ON\left(\sigma^{(n)}\right) &= \min_j l_j^{n-1} + a_n \leq \frac{\sum_{j=1}^m l_j^{n-1}}{m} + a_n \overset{(**)}{=} \frac{\sum_{i=1}^{n-1} a_i}{m} + a_n \\ &= \frac{\sum_{i=1}^n a_i}{m} + \left(1 - \frac{1}{m}\right) \cdot a_n \overset{(*)}{\leq} OPT\left(\sigma^{(n)}\right) + \left(1 - \frac{1}{m}\right) OPT\left(\sigma^{(n)}\right) \\ &= \left(2 - \frac{1}{m}\right) \cdot OPT\left(\sigma^{(n)}\right) \end{aligned}$$

■

2.2.2 הרחבה לבעית איזון העומסים - מכונות תלויות

המכונות אינן זהות - למכונה ה-j יש מהירות $v_j \leq 1$.

כמו כן, המכונות תלויות $(related)$ - העומס של תהליך i על מכונה j הוא $\frac{a_i}{v_j}$ (עבור מכונות בלתי תלויות, העומס נקבע ע"י פרמטר a_{ij} - כאן היחס הזה הוא דוגמא לתלות, קיימות גם הגדרות אחרות שמגדירות מכונות תלויות).

אפשר לדבר כאן על האלגוריתם החמדן, שפועל בצורה שבה לאחר הצבה של תהליך, העומס הוא הכי נמוך. לאלגוריתם זה יש יחס $\log n$, ומסתבר שאפשר להגיע ליחס קבוע (לא ע"י אלגוריתם חמדן).

[10] אלו שני המקרים היחידים שכן $makespan$ לא יכול לקטון.

2.2.3 בעיית הפרה האבודה

זוהי בעיה קצת אחרת במובן שהיא דומה לבעיה של רובוטים, בה הרעיון של סדרת הוראות הוא פחות אינטואיטיבי.

המרצה מעדיף להתייחס אליה כ"בעית המכונית האבודה" בשביל התייחסות יותר מציאותית.

המצב:

יש שביל אינסופי, שאיפשהוא יש עליו פרה, אנחנו לא יודעים איפה, אנחנו מתחילים מנקודה מסויימת ומנסים למצוא אותה.

בכל שלב מותר לעשות צעד שמאלה או צעד ימינה. השאלה היא כמה זמן יקח למצוא את הפרה.

אנחנו נדון במקרה בו הפרה סטטית.

כאן ההחלטות שביצענו בשלבים הקודמים נותנות לנו את המידע בכל שלב - כאשר המידע הוא ש"אין פרה" עד ש"יש פרה" (בוריאציה שלנו אין עניין של יחסיות, למרות שניתן לדון בבעיות שכאלו).

אלגוריתם 15 אלגוריתם לבעית הפרה האבודה

```
d←1 side=left
while cow not found:
 return to origin
 walk d in direction (side)
 d←2d
 side←opposite side
```

משפט 2.9 האלגוריתם הנ"ל הוא 9־תחרותי חזק.

הוכחה: נחשוב על יריב - יראה את הצעדים ויחליט היכן ישים את הפרה[11]. בכל צעד, היריב ישים את הפרה קצת רחוק באותו הכיוון, כדי להכריח את האלגוריתם ללכת לכיוון השני ואז לחזור ע"מ למצוא את הפרה.

באופן פורמלי, כאשר $j+2$ הוא מספר האיטרציות של האלגוריתם, אז מרחק הפרה מהראשית היא $d = 2^j + \varepsilon$.

$$OPT(\sigma) = d = 2^j + \varepsilon \sim 2^j$$

$$ON(\sigma) = \overbrace{2\left(1+2+...+2^{j+1}\right)}^{\text{steps taken}} + \overbrace{2^j + \varepsilon}^{\text{get to the cow from origin}} \leq 2 \cdot 2^{j+2} + 2^j = 9 \cdot 2^j$$

■

[11] הפרה סטטית, אבל האלגוריתם דטרמיניסטי אז אנחנו יודעים מה הוא יעשה על כל קלט, ולכן זה שווה ערך לבחירת הקלט הכי גרוע.

2.2.4 בעית הסקי

זוהי אחת הבעיות הראשונות התעסקו בה בתחום.
יוצאים לחופשת סקי, לזמן לא ידוע מראש n (לא ידוע כמה ימים החופשה תמשך). יש שתי אפשרויות:

- לקנות ציוד סקי במחיר כלשהוא $B > 1$.
- לשכור ציוד ליום במחיר 1.

הערה 2.10 בעיה זו קיימת בהרבה בעיות אחרות בצורה פחות ברורה - למשל, רוצים לבצע תהליך באינטרנט - אפשר לשלם עבור כל פעולה בנפרד, או "לקנות מנוי".

אפשר להסתכל על הבעיה כסדרת בקשות לעשות סקי עד עצירה:

$$\sigma = \overbrace{1111111....1}^{n\text{times}}0$$

כאשר $OPT(B) = min\{B, n\}$.
את קבוצת האלגוריתמים המקוונים האפשריים לבעיה ניתן להגדיר באופן הבא:
עבור $\mathbb{N} \ni s \geq 1$, ON_s הוא האלגוריתם ששוכר $s-1$ ימים ואז קונה. אזי:

$$ON_s(\sigma) = \begin{cases} n & s > n \\ s - 1 + B & s \leq n \end{cases}$$

אנו נסתכל על האלגוריתם ON_B. נציב ונקבל עבורו:

$$ON_B(\sigma) = \begin{cases} n & s > n \\ 2B - 1 & s \leq n \end{cases}$$

משפט 2.11 האלגוריתם ON_B הוא $2 - \frac{1}{B}$ תחרותי חזק.

הוכחה: עבור $B > n$:

$$\frac{ON_B(\sigma)}{OPT(\sigma)} = \frac{n}{n} = 1$$

עבור $B \leq n$:

$$\frac{ON_B(\sigma)}{OPT(\sigma)} = \frac{2B-1}{B} = 2 - \frac{1}{B}$$

■

משפט 2.12 יש חסם תחתון $2 - \frac{1}{B}$ לבעיית הסקי.

הוכחה: יש להראות כי לכל $s \geq 1$ קיימת סדרה σ ($\Leftrightarrow$ קיים ערך של n) כך ש: $\frac{ON_s(\sigma)}{OPT(\sigma)} \geq 2 - \frac{1}{B}$.

נבחר $n = s$[12]. במקרה זה:

$$\frac{ON_s(\sigma)}{OPT(\sigma)} = \frac{s-1+B}{min\{s,B\}} = \frac{s}{min\{s,B\}} + \frac{B-1}{min\{s,B\}} \geq \frac{s}{s} + \frac{B-1}{B} = 2 - \frac{1}{B}$$

■

2.3 אלגוריתמים רנדומיים

הגדרה 2.13 אלגוריתם ON רנדומי הוא c-תחרותי חזק אם:

$$\mathbb{E}[ON(\sigma)] \leq c \cdot OPT(\sigma)$$

בעוד שעבור אלגוריתמים רנדומיים אנחנו לא יכולים לדבר על התנהגות במקרה מסויים, אפשר לנתח את התוחלת של ההתנהגות שלו.

2.3.1 דוגמא - שיפור לניתוח לבעית הסקי

נגדיר אלגוריתם רנדומי:

אלגוריתם 16 אלגוריתם ON רנדומי לבעית הסקי

נגריל הסתברות θ לקנות ביום הראשון, אחרת נפעיל את אלגוריתם ON_B.

למה אינטואיטיבית זה משפר? המקרה הכי גרוע שלנו הוא במקרה בו החופשה ארכה B ימים, שהרי אז העלות היא $2B-1$, ואז עדיף היה לנו לקנות ביום הראשון.

ניתוח פורמלי:

$$\mathbb{E}[ON(\sigma)] = \theta \cdot B + (1-\theta) ON_B(\sigma)$$

במקרה בו $n < B$:

$$\frac{\mathbb{E}[ON(\sigma)]}{OPT(\sigma)} = \frac{\theta \cdot B + (1-\theta) n}{n} = \frac{\theta \cdot B}{n} + 1 - \theta \leq \theta \cdot B + 1 - \theta$$

במקרה בו $n \geq B$:

$$\frac{\mathbb{E}[ON(\sigma)]}{OPT(\sigma)} = \frac{\theta \cdot B + (1-\theta)(2B-1)}{B} = \theta + (1-\theta) \cdot \left(2 - \frac{1}{B}\right)$$
$$= -\theta + 2 - \frac{1}{B} + \frac{\theta}{B}$$

נרצה למצוא את θ שממקסם את התוצאה. עבור $\theta = 0$ נקבל את האלגוריתם ON_B (אף פעם לא נגריל לקנות ביום הראשון).

[12] שכן קביעת n מגדירה לנו את הסדרה.

הפונקציה שקיבלנו במקרה הראשון עולה ב־θ, השניה יורדת ב־θ, ונקודת המפגש של שתי הפונקציות תמקסם את כל הביטוי. נשווה ביניהם ונמצא:

$$\Rightarrow \theta\,(B-1) = (1-\theta)\cdot\frac{B-1}{B}$$
$$\Rightarrow B\cdot\theta = 1-\theta$$
$$\Rightarrow \theta = \frac{1}{B+1}$$

נציב הערך שהתקבל באחת מהפונקציות ונקבל יחס של $2-\frac{2}{B+1}$, שהוא קצת יותר טוב ממה שקיבלנו קודם.

כאשר B קרוב לאחד מקבלים כי אלגוריתם זה קרוב לאופטימלי.

2.4 בעיית דפדוף - $paging$

4/6/2012

תיאור הבעיה: במערכת מחשב[13] ישנו זיכרון איטי M בגודל n, וזיכרון מהיר $K\ (cache)$ בגודל k.

מקבלים סדרה של בקשות לדפים. כאשר מקבלים בקשה לדף p, הוא צריך להיות בזיכרון המהיר. אם הוא לא נמצא בזיכרון המהיר, צריך להביא אותו לזיכרון המהיר מהזיכרון האיטי. למצב כזה של "החטאה" קוראים $page\ fault$, ויש לה עלות. במקרה שאנו דנים בו, לאלגוריתם יש עלות 1 בעקבות מקרה כזה.

אם הזיכרון המהיר מלא, חייבים להוציא ממנו דף אחר q. ההחלטה איזה דף להוציא היא ההחלטה העיקרית של אלגוריתם דפדוף.

זוהי בעיה קלאסית במערכות הפעלה, ויש לה מספר אלגוריתמים המיושמים היום. אולם, כאשר הסתכלו על הבעיה בקונטקסט שאנו מדברים אליו, לא היה ניתוח לבעיה הזו. המטרה היתה לתת הסבר מתי אלגוריתם דפדוף יותר או פחות טוב.

כאשר מסתכלים על הבעיה הזו כבעיית אופליין, כלומר כאשר סדרת הבקשות ידועה מראש, ידוע אלגוריתם פולינומי אופטימלי לפתרונה:

אלגוריתם 17 האלגוריתם האופטימלי לבעית $offline$ - LFD - Longest Forward Distance

---

בהנתן סדרת בקשות לדפים, האלגוריתם מוציא את הדף שיבקשו אותו הכי רחוק בעתיד.

כבר ראינו את ההוכחה בקורסים קודמים - מסתכלים על סדרת בקשות, ומוכיחים באינדוקציה שאם לא היינו עושים את הצעד הזה, היה אפשר לשפר את המחיר רק על ידי דחיית ביצוע הפעולה.

Sleator and Tarjan הם שני חוקרים מפורסמים שהציעו את הצורה הזו של הניתוח, והתחילו עם הבעיה הזו (ובעיה נוספת שלא נדבר עליה).

ראינו שתחרותיות היא להערכת הביצועים של אלגוריתם אונליין דטרמיניסטי מול האלגוריתם האופטימלי. ראינו כי אנו לא מצפים למחיר דומה לאופטימלי, וכן ראינו כי אפשר למצוא תמיד סדרה שאלגוריתם אחד יעבוד עליה טוב, ואחר לא.

נתחיל דווקא מהצד הלא טוב של ההערכה הזו - יכול להיות שנהיה הרבה יותר גרועים מהאופטימלי. ננסח זאת באופן פורמלי:

[13] אפשר לדבר גם על הבעיה הזו ברשתות.

משפט 2.14 לבעיית הדפדוף יש חסם תחתון של k על יחס התחרותיות.

הוכחה: לכל אלגוריתם מקוון ON ניתן לייצר סדרת בקשות σ כך ש:

$$ON(\sigma) = |\sigma|$$

באופן הבא: בהתחלה מבקשים $k+1$ דפים שונים. לאחר מכן היריב תמיד יבקש את הדף שהאלגוריתם הוציא.

בשביל להראות כי החסם התחתון הוא k, יש להראות כי המחיר של האופטימלי הוא לא יותר מ־$\frac{1}{k} \cdot |\sigma|$.

טענה 2.15 תחת ההנחה $n = k+1$[14], לכל סדרת בקשת σ, $OPT(\sigma) \leq \frac{|\sigma|}{k}$.

הוכחה: נחלק את סדרת הבקשות σ לפאזות, כך שבכל פאזה $\sigma^{(i)}$ יש בקשות ל־k דפים שונים, ואז:

$$\sigma = \sigma^{(1)}\sigma^{(2)}...\sigma^{(t)},\ t \leq \frac{|\sigma|}{k}$$

כלומר $\sigma^{(i)}$ היא הפאזה שהיא הסדרה הארוכה ביותר של בקשות המכילה k דפים שונים.

האלגוריתם האופטימלי, בעת קבלת הבקשה הראשונה ב־$\sigma^{(i)}$, יוציא את הדף הראשון ב־$\sigma^{(i+1)}$, שכן הוא הדף ה־$k+1$ השונה בסדרת הבקשות, כלומר יבקשו אותו הכי רחוק בעתיד, ויכניס את הדף הראשון ב־$\sigma^{(i)}$, ולכן יש לו page fault (פסיקה) בכל פאזה.

על כן, מכיוון ומספר הפאזות $t \leq \frac{|\sigma|}{k}$, נקבל כי $OPT(\sigma)$ הוא לכל היותר t. ועל כן $OPT(\sigma) \leq \frac{|\sigma|}{k}$, כנדרש. ■

■

כאשר k קטן זה טוב, אחרת "זה לא נשמע כל כך אטרקטיבי". אבל לא סיימנו, כי אפשר לדבר עדיין על אלגוריתמים רנדומיים, שם החסם התחתון הזה לא קיים. כמו כן, התחרותיות היא אולי באמת לא יחס כל כך טוב, אבל היא כן נותנת לנו מושג "על מה הולך".

2.4.1 אלגוריתמים מכוונים לבעיה

- LRU -Least Recently Used - הוצא את הדף שהתבקש הכי רחוק בעבר מכל הדפים שנמצאים ב־$cache$.

  לאור האלגוריתם האופטימלי, אלגוריתם זה אינטואיטיבי, אבל היעילות שלו תלויה בצורה של סדרת הבקשות. במידה ויש locality בסדרה, יש הגיון מסויים באלגוריתם זה, שכן זה הדף שהתרחקנו ממנו הכי הרבה.

  היו אנשים שניסו לסבך את המודל הזה ולהכניס את ה־locality לתוכו ולהגיד דברים מעניינים, אבל אז יש ויכוחים על המודל עצמו.

- $FIFO$ - First In Last Out - הוצא את הדף שנכנס ראשון לזיכרון המלא.

  גם וריאציות של אלגוריתם זה נמצאות היום בשימוש.

- FWF - Flush When Full - כשהזיכרון מלא, הוצא את כל הדפים.

  אפשר להגיד עליו דווקא משהו חיובי בעזרת צורת הניתוח של תחרותיות, וזה אולי אומר משהו רע על צורת הניתוח, שכן זה אלגוריתם מאוד לא יעיל לשימוש אמיתי.

[14] ההנחה לא מגבילה אותנו, שכן במקרים גדולים יותר אפשר להסתכל רק על חלק מהזיכרון, ואם הוא קטן יותר הזיכרון המהיר לא יתמלא.

כל האלגוריתמים מעלה טובים מבחינת יחס התחרותיות. הבא לעומת זאת לא:

- $LIFO$ - Last in First Out - הוצא את הדף שנכנס אחרון לזיכרון המהיר.

הוא רע מבחינת תחרותיות למשל באופן הבא:
נמלא את ה־$cashe$ בדפים $1, ..., k$, מבקשים את הדף ה־$k+1$ ומוציאים את k, שוב מבקשים את k אז מוציאים את $k+1$, וחוזר חלילה, אזי עלותו תהיה $|\sigma|$. האלגוריתם האופטימלי יוציא שני דפים אחרים, ואז עלותו תהיה קבועה: $OPT(\sigma) = k+1$. על כן היחס ביניהם:

$$\frac{LIFO(\sigma)}{OPT(\sigma)} \xrightarrow{|\sigma|\to\infty} \infty$$

כלומר אינו חסום.
הדוגמא הזו מראה קודם כל שהבעיה אינה טריוויאלית לגמרי, ויחס התחרותיות יכול להיות אינסופי. כלומר קיבלנו מיחס התחרותיות דרך להגיד ש־$LIFO$ גרוע יותר מ־$FIFO$.[15]

כדי להגיד משהו על חלק מהאלגוריתמים הנ"ל, נגדיר קבוצה גדולה יותר של אלגוריתמים:
אלגוריתמי סימון (Marking Algorithms) היא מחלקה של אלגוריתמים, שנראה כי הם k־תחרותיים (כלומר יחס התחרותיות שלהם אופטימלי). לפני שנגדיר מחלקה זו, יש צורך בפיתוח נוסף:

הגדרה 2.16 נגדיר חלוקה של סדרת בקשות σ לפאזות בעזרת תהליך של סימון דפים בזיכרון המהיר:

1. בהתחלת פאזה כל הדפים לא מסומנים.
2. כאשר מבקשים דף במהלך פאזה (בין אם הוא בזיכרון המהיר ובין אם לאו), מסמנים אותו.
3. פאזה מסתיימת כאשר כל הדפים בזיכרון המהיר מסומנים ונוצר $p.f.$. בסיום פאזה מוחקים את כל הסימנים.

הגדרה 2.17 אלגוריתם הוא אלגוריתם סימון אם בכל שלב בפאזה, בהנתן פסיקה $(p.f.)$ האלגוריתם מוציא את אחד הדפים הלא מסומנים.

כלומר, האלגוריתם מוציא דפים שלא התבקשו עדיין במהלך הפאזה. הפאזה מסתיימת כאשר כל הדפים שנמצאים בזיכרון המהיר כבר בוקשו, ובעת קבלת בקשה חדשה מתחילה פאזה חדשה.
שניים מהאלגוריתמים שראינו הם אלגוריתמי סימון - $FIFO$ לא. לא קשה לראות כי $FWF,\ LRU$ הם אלגוריתמי סימון.

משפט 2.18 כל אלגוריתם סימון הוא k־תחרותי.

הוכחה: במהלך פאזה, תמיד יש בקשות ל־k דפים שונים (יכול להיות שחלקם בזיכרון המהיר וחלקם לא), אחרת לא נסמן k דפים ולא נעבור פאזה. על כל דף אנו משלמים אם הדף לא היה ב־$cache$. אזי, מחיר האלגוריתם $MARK$ בפאזה $k \geq$ (במקרה הגרוע ביותר מתרחש

[15] לבעיה הזו וגם למצבים דומים

כאשר אף אחת מהבקשות אינן בזיכרון המהיר). נראה שלאלגוריתם האופטימלי יש עלות של לפחות 1 לכל פאזה:
נביט בפאזות $\sigma^{(i)}, \sigma^{(i+1)}$. נסמן ב־$p$ את הדף הראשון ב־$\sigma^{(i)}$, וב־q את הדף הראשון ב־$\sigma^{(i+1)}$.
במהלך פאזה, יש כאמור בקשות ל־k דפים שונים. על כן, על הסדרה שמתחילה בבקשה ל־p ומסתיימת בבקשה ל־q (המכילה $k+1$ בקשות לדפים שונים), לאלגוריתם האופטימלי יש עלות ≥ 1. אפשר ליחס את העלות לפאזה ה־i. ■

וריאציה דומה מראה שגם ל־$FIFO$ יש יחס כזה. אפשר להגיד עוד על אלגוריתמי סימון רק אם משנים את המודל. על כן, נמשיך ונדבר על אלגוריתמים רנדומיים:

2.4.2 אלגוריתמים רנדומיים - Random Marking

אלגוריתם 18 Random Marking

מוציאים דף לא מסומן באקראי ע"פ התפלגות אחידה על הדפים הלא מסומנים.

במקרה הכי גרוע זה כמו כל אלגוריתם סימון אחר, כלומר יחס תחרותיות k. אחרת, היריב לא תמיד יכול לצפות את הצעד שלך, ואז יחס התחרותיות משתפר.

משפט 2.19 RM הוא $2H_k$־תחרותי, כאשר:

$$H_k = 1 + \frac{1}{2} + \frac{1}{3} + ... + \frac{1}{k},\ ln(k) < H_k \leq ln(k) + 1$$

משפט 2.20 לבעיית הדפדוף (לא נראה היום) לאלגוריתמים רנדומים יש יחס תחתון של H_k.

הערה 2.21 האלגוריתם במשפט מעלה הוא הרבה יותר מסובך.

נוכיח את המשפט הראשון: **הוכחה:** נסתכל על חלוקה של סדרת הבקשות σ לפאזות: $\sigma = \sigma^{(1)}\sigma^{(2)}...\sigma^{(t)}$.
במהלך פאזה מסמנים את הדפים המבוקשים, לעיתים צריך להביא דף מהזיכרון האיטי ומסמנים אותו. כאשר אנחנו מוציאים, אנו בוחרים מתוך הדפים שלא מסומנים בהסתברות אחידה. נגדיר:
דף יקרא "חדש" בפאזה ה־i אם הוא התבקש במהלך הפאזה $\sigma^{(i)}$ ולא היה ב־$\sigma^{(i-1)}$ (כלומר לא סומן בפאזה הקודמת) - מכיוון והזיכרון המהיר התמלא בבקשות אחרות, דף "חדש" אינו בזיכרון המהיר בתחילת הפאזה ה־i, ועל כן כאשר מבקשים אותו, יש להביא אותו מה־$cache$. דף "ישן" הוא דף שאינו חדש, כלומר שהתבקש במהלך פאזה $\sigma^{(i-1)}$, ולכן הוא היה בזיכרון המהיר בתחילת הפאזה ה־i.
נסמן ב־n_i את מספר הדפים החדשים בפאזה ה־i.

טענה 2.22

$$\mathbb{E}[ON(\sigma)] \leq H_k \cdot \sum_{i=1}^{t} n_i$$

יותר ספציפית, אם נסמן ב־ON_i את המחיר פר פאזה, אז:

$$\mathbb{E}\left[ON_i\right] \leq H_k \cdot n_i$$

טענה 2.23

$$OPT\left(\sigma\right) \geq \frac{1}{2} \cdot \sum_{i=1}^{t} n_i$$

לצורך הטענה מעלה, נוכיח את הטענה הבאה:
אם נסמן ב־OPT_i את המחיר של האלגוריתם האופטימלי לפאזה ה־i, אזי:

$$OPT_{i-1} + OPT_i \geq n_i$$

ברור כי משתי הטענות נקבל את הנדרש.
נוכיח תחילה את טענה 2: **הוכחה:** בסה"כ על שתי הפאזות $i-1, i$ יש בקשות ל־$k+n_i$ דפים שונים, ולכן כל אלגוריתם חייב לשלם עלות של לפחות n_i במהלכן, כלומר:

$$OPT_{i-1} + OPT_i \geq n_i$$

אזי:

$$\sum_{i>1} \left(OPT_{i-1} + OPT_i\right) \geq \sum_{i>1} n_i$$

$$\Rightarrow 2 \cdot OPT\left(\sigma\right) = OPT_1 + OPT_1 + 2 \cdot \sum_{i>1} OPT_i \geq \sum_{i>1} n_i + n_1 = \sum_{i=1}^{n} n_i$$

■

נותר להוכיח את הטענה הראשונה ע"מ לסיים: **הוכחה:** בפאזה ה־i יש $k-n_i$ דפים ישנים שמתבקשים במהלכה. נסמנם ב־$p_1, p_2, ..., p_{k-n_i}$.
אזי, ניתן לתאר את התוחלת של מחיר האלגוריתם באופן הבא:
כאמור את הדפים החדשים חייבים להביא מהזיכרון האיטי, ועל כן חייבים לשלם עליהם. על הדפים הישנים יש לשלם רק אם הוצאנו אותם לפני שביקשנו אותם בפאזה ה־i. אזי:

$$\mathbb{E}\left[ON_i\right] = \overbrace{n_i}^{new} + \overbrace{\sum_{j=1}^{k-n_i} Pr\left[p_j \text{ wasn't in the cache when it was requested for the first time in } \sigma^{(i)}\right]}^{old}$$

נסמן ב־$\tilde{n}_j$ את מספר הדפים החדשים שהתבקשו עד לבקשה הראשונה ל־p_j. נשים לב כי $\tilde{n}_j \leq n_i$.
בשלב זה, יש $j-1$ דפים ישנים שכבר התבקשו בפאזה ה־i לפני הבקשה הראשונה ל־p_j, ולכן הם מסומנים כבר.

הקבוצה שנשארה היא $k-(j-1)-\tilde{n}_j$ דפים שאינם מסומנים ונמצאים בזיכרון המהיר. אבל בסה"כ, יש $k-(j-1)$ דפים ישנים שאינם מסומנים, ו־$\tilde{n}_j$ מהם בזיכרון האיטי. נקרא לקבוצת הדפים הישנים שאינם מסומנים והם בזיכרון האיטי ב־Z, כלומר הדפים הישנים שהוצאו מהזיכרון המהיר ועדיין לא בוקשו, אך יבוקשו במהלך הפאזה. מתוך $k-(j-1)$ הדפים הישנים שאינם מסומנים, הקבוצה Z נבחרת בהתפלגות אחידה, כלומר:

$$Pr\left[p_j \in Z\right] = \frac{\tilde{n}_j}{k-(j-1)}$$

זה נכון, כי האלגוריתם בוחר את הדף שיצא בהתפלגות אחידה, ואנו בוחרים אותו מתוך הדפים הישנים שעדיין לא סומנו הנמצאים בזיכרון המהיר (כי דף חדש על פי הגדרות יסומן כאשר מביאים אותו מהזיכרון האיטי). אזי:

$$\mathbb{E}\left[ON_i\right] = n_i + \sum_{j=1}^{k-n_i} \frac{\tilde{n}_j}{k-(j-1)} \leq n_i + n_i \cdot \sum_{j=1}^{k-n_i} \frac{1}{k-(j-1)} = n_i \cdot \left(1 + \sum_{j=1}^{k-n_i} \frac{1}{k-(j-1)}\right)$$

$$= n_i \cdot \left(1 + \frac{1}{k} + \frac{1}{k-1} + ... + \frac{1}{n_i+1}\right) \leq n_i \left(1 + \frac{1}{2} + \frac{1}{3} + ... + \frac{1}{k}\right)$$

■

■

נרצה גם להגיד משהו על החסם התחתון:

טענה 2.24 לכל אלגוריתם סימון רנדומי יש יחס תחרותיות לפחות H_k.

הוכחה: נגביל את עצמו ל־$n = k+1$[16].

הסימונים במהלך פאזה, גם בתחילת\סיום פאזה, הם תהליכים דטרמיניסטיים (מה שלא הוא הדף שמוציאים, אבל בכל מקרה כל פעם שמבקשים דף הוא מסומן, ועל כן הסימון דטרמיניסטי).

בתחילת הפאזה מבקשים את הדף[17] שלא נמצא בזיכרון המהיר (דף חדש). בכל שלב, היריב יבקש את **<u>אחד</u>** הדפים הלא מסומנים (היריב כאמור רוצה להקשות על האלגוריתם, לכן אם יבקש דף מסומן הוא בטוח לא "יעלה" דבר לאלגוריתם).

כבר הראנו שמחיר האופטימלי לפאזה $= 1$.

היריב בוחר לבקש מבין הדפים הלא מסומנים את הדף בעל ההסתברות הגבוהה ביותר (לפי התפלגות אלגוריתם הסימון) לא להמצא בזיכרון המהיר.

בשלב ה־j של התהליך (כלומר לאחר $j-1$ בקשות), בוחרים מבין $k-(j-1)$ דפים לא מסומנים, ולכן חייב להיות אחד מהם שההסתברות שהוא לא נמצא בזיכרון המהיר היא $\leq \frac{1}{k-(j-1)}$. נקבל כי:

$$\mathbb{E}\left[ON\left(\sigma\right)\right] \geq \overbrace{1}^{\text{new page}} + \sum_{j>1}^{k-1} \frac{1}{k-(j-1)} = H_k$$

■

[16] כאמור זו לא באמת מגבלה, שכן אפשר אם n גדול יותר פשוט להתעלם משאר הדפים.

[17] שכן הזיכרון המהיר בגודל k ויש בסה"כ $k+1$ דפים.

2.5 בעיית k־השרתים

זוהי בעיה מאוד יפה וקלאסית בנושא הזה, והרבה אנשים עסקו בה לאורך השנים.
נזכר בהגדרה של מרחב מטרי:

הגדרה 2.25 מרחב מטרי מוגדר ע"י זוג (X,d), כאשר X היא קבוצת נקודות, $d:X^2\to\mathbb{R}^2$ היא פונקציית מרחק, כלומר פונקציה המקיימת את התכונות הבאות:

1. רפלקסיביות - $\forall u,v\in X\ u=v\Leftrightarrow d(u,v)=0$.
2. סימטריה - $\forall u,v\in X\ d(u,v)=d(v,u)$.
3. א"ש המשולש - $\forall u,v,w\in X\ d(u,v)+d(v,w)\geq d(u,w)$.

הערה 2.26 אפשר לדבר גם על פונקציות מרחק שלא מקיימות את התכונות האלו, אבל אז המתמטיקה הרבה פחות "זורמת יפה", ועל כן אנו נשאר בהגדרה הזו.

הגדרת הבעיה: נתון מרחב מטרי (M,d). ב־k נקודות במרחב נמצאים k שרתים. מגיעה סדרת בקשות $\sigma=\sigma_1\sigma_2...$, כאשר כל בקשה היא נקודה $\sigma_i\ni M$.
בהנתן בקשה $r\ni M$, צריך למלא את הבקשה ע"י הזזת אחד השרתים למיקום הבקשה r. העלות המיוחסת לשירות הבקשה היא המרחק בין מיקום השרת לבקשה.
רוצים להביא למינימום את העלות הכוללת.

2.5.1 בעיות שקולות

בעיה זו כוללת בתוכה מספר בעיות נוספות, וזהו חלק מהמשיכה שלה.

בעיית ה־$Paging$: מרחב שווה־צלעות\מרחב יוניפורמי - $equiilateral\ space$: כל המרחקים הם 1. דוגמא באיור - מרחב מגודל 4:
איור
נקודות במרחב הם דפים, נק' שיש בהם שרתים הם דפים בזיכרון המהיר. נניח שהדפים $2,4$ בזיכרון המהיר, אז אם יש בקשה בזיכרון המהיר העלות היא 0, ואם יש בקשה לדף שאינו בזיכרון, אז צריך להזיז את אחד השרתים, והעלות לכך היא 1.

דפדוף ממושקל $Weigted\ Paging$ עוד מקרה פרטי של הבעיה.

תיאור הבעיה: הבאה של דף i לזיכרון עולה מחיר w_i.

הערה 2.27 המרחב במקרה הזה אינו מטריקה! קל לראות כי אין רפלקסיביות, וכן אי שיוויון המשולש לא מתקיים.
עם זאת, זו עדיין בעיית k־השרתים עם פונקציית מרחק "משונה במקצת".

אולם, ניתן להמיר את הבעיה לבעיה שקולה על מרחב מטרי של כוכב ממושקל על $n+1$ נקודות. למשל, בדוגמא הקודמת:
איור
לגרף כולל מרכז חדש, ואת n הנקודות המקוריות. המרחב המטרי החדש מוגדר כך שעלות המעבר\המרחק בין נקודות i,j היא העלות המעבר בין i למרכז, ובין המרכז ל־j.
על מנת לקבל את בעיית הדפדוף הממושקל, נגדיר את המרחק\עלות בין נקודה i למרכז להיות $\frac{1}{2}\cdot w_i$.

טענה 2.28 בהנתן סדרת בקשות כלשהיא ואלגוריתם לשרת אותן, העלות בבעיה המטרית שווה לעלות בבעיית ה־$weighted\ paging$ עד כדי קבוע אדיטיבי.

הוכחה: בסדרה מספיק ארוכה, נסתכל על כל הפעמים שהבאנו שרת לנקודה אחת. פרט לפעם הראשונה והאחרונה, אנחנו משלמים w_i ־ בהבאה של השרת לנקודה משלמים מחצית, וכאשר מוציאים אותו ומעבירים את השרת לנקודה אחרת משלמים שוב מחצית.
אז נשארנו עם קבוע אדטיבי עבור הפעם הראשונה והפעם האחרונה שהבאנו את השרת לנקודה מסויימת ־ מחצית מהנקודה ממנה הגענו. ■

הערה 2.29 מכיוון ואנו מתעניינים בתחרותיות לא חזקה, קבוע אדטיבי לא משנה.

$\mathbb{R} = M$ הוא הישר הממשי, k הנקודות נמצאות עליו. נקודה על הישר היא בקשה, וצריך להזיז את השרתים לנקודה.

טענה 2.30 זהו מודל מאוד אבסטרקטי לבעיית תזמון ראשי דיסק.

בפועל יכולה להיות הגבלה של איזורים של ראשי דיסק, אבל בפועל אין לכך חשיבות בבעיה, שכן זה "לא כדאי" (למרות שנראה אלגוריתמים שכן משתמשים בזה).

2.5.2 ניתוח הבעיה

ראינו את הגדרה של α־תחרותיות:

$$\exists a,\ \forall \sigma\ ON(\sigma) \leq \alpha \cdot OPT(\sigma) + a$$

הערה 2.31 הקבוע האדטיבי פחות רלוונטי כאשר האלגוריתם $Offline$ האופטימלי והאלגוריתם ה־$Online$ מתחילים באותו מצב (כלומר אין יתרון התחלתי לאלגוריתם האופטימלי ־ לא מאפשרים לו לסדר את הבקשות בסדר שונה).

משפט Mannase-Mrgeach-Sleator

משפט 2.32 לבעיית k־השרתים בכל מרחב מטרי יש חסם תחתון של k על יחס התחרותיות.

משפט Koutsoupias-Papadioution

משפט 2.33 (לכל מרחב מטרי) קיים אלגוריתם $(2k-1)$־תחרותי לבעיית k־השרתים.

ישנם מקרים ספציפיים שאנו יודעים לגביהם שקיים אלגוריתם k־תחרותי, למשל:

משפט 2.34 קיים אלגוריתם Double Coverage k־תחרותי לבעיית k־השרתים על הישר.

לא נעסוק כמעט באלגוריתמים רנדומיים לבעיה - שם דווקא היתה התקדמות לאחרונה.
נוכיח את המשפט הראשון - החסם התחתון. ההוכחה שלו היא "כ"כ נקיה ונחמדה" שהיא משרה קצת אמונה שאולי המשפט לגבי החסם העליון גם יהיה פשוט להוכחה. אבל, לצערנו, עד כה לא מצאו כזו. **הוכחה:** נתרכז במקרה בו $n = k + 1$[18].
נסמן $M = \{1, 2, ..., k + 1\}$, ונניח שבהתחלה השרתים בנקודות $\{1, ..., k\}$.
בהנתן אלגוריתם מקוון ON דטרמיניסטי, רוצים להגדיר יריב (ADV) וסדרת בקשות σ כך ש:

$$ON(\sigma) \geq k \cdot ADV(\sigma) - a$$

עבור a קבוע. זה מספיק, שכן אם נוכיח את זה, בעצם נראה שלא ייתכן שקיים a' ו-$k > \alpha$ כך ש:

$$ON(\sigma) \leq \alpha \cdot ADV(\sigma) + a$$

לכל σ, כפי שאנו רוצים להוכיח.
למעשה, במקום להגדיר יריב אחד, נגדיר k יריבים שונים:

$$ADV_1, ..., ADV_k$$

ונראה שקיימת סדרה σ כך ש:

$$ON(\sigma) \geq \sum_{i=1}^{k} ADV_i - a$$

כאמור, אם לא נרשה יתרון התחלתי ליריבים, הקבוע האדטיבי לא רלוונטי.
ברור כי אם נראה זאת, זה יספיק, שכן במקרה זה נבחר ADV_i המקבל את המחיר הנמוך ביותר על הסדרה, נכפול ב-k, ואז זה בוודאי נכון.
הסדרה σ מוגדרת כך שבכל שלב, ניתן בקשה בנקודה שאין בה שרת של האלגוריתם[19].
השאלה היא איך היריב משרת את הבקשות הללו. אנו נגדיר כאמור k יריבים, כל אחד ישרת את הבקשה בצורה שונה:
היריב ה-ADV_i בהתחלה יעבור לקונפיגורציה של שרתים שבה שרתים נמצאים בנקודות $M - \{i\}$[20].
נשמור על שתי תכונות:

1. בכל שלב, לכל היריבים יש שרת בנקודה שבה לאלגוריתם ON אין שרת.
2. לכל נקודה i שבה יש לאלגוריתם ON שרת, קיים יריב יחיד שאין לו שרת בנקודה זו.

כלומר אנו רוצים לשמור על היחס בין היריבים לאלגוריתם ON כפי שהוא במצב ההתחלתי.
נותר להראות כיצד עוברים בעת קבלת בקשה, מהי עלות האלגוריתם ON ומהי העלות עבור ADV_i:
נגיד שהתקבלה בקשה, בה"כ לדף $k + 1$ - אזי ל-ON אין בה שרת, אז הוא יעביר שרת מנקודה i לנקודה המבוקשת. אזי הבקשה הבאה תהיה i. על מנת לענות על הבקשה (לפני

[18] אם יש יותר נקודות, אפשר לצמצם את ההתייחסות ל-$k + 1$ נקודות, כפי שראינו בבעית ה-*paging*.
[19] כמו ב-paging - נותנים בקשה לדף שאינו ב-*cache*.
[20] כאמור יש k שרתים, $k - 1$ נקודות, על כן בכל נקודת זמן יש שרתים בכל הנקודות פרט לאחת.

i), לכל היריבים פרט ליריב ה־i יש שם שרת. אזי העלות שלהם היא 0. על מנת למלא את הבקשה, היריב ה־i יעביר את השרת שלו מנקודה $k+1$ ל־i.

מחיר ON לבקשה: $d(i,k+1)$.

מחיר היריב ה־i: $d(k+1,i)$, ולשאר היריבים 0.

זה מוכיח את הטענה, שכן מחיר $\sum_{i=1}^{k} ADV_i = ON$ פרט לקבוע האדטיבי a, שהוא המחיר העולה לכל יריב לעבור לקונפיגורציה ההתחלתית במידה והיא שונה מזו של ON. ■

הערה 2.35 תרגיל כיף לבית - החליפו את העלויות ב־1 בהוכחה הקודמת, וקבלו הוכחה אחרת לטענה עבור בעיית ה־$paging$.

נוכיח את המשפט השלישי, עבור הבעיה על הישר. למען האמת נגיד כאן מספר דברים שהם גם רלוונטים לבעיה הכללית. נביט במצב הבא:

איור

$k=2, n=3$. מצב התחלתי - השרתים בנקודות A, C.

במקרה ונקבל סדרת בקשות $BABABAB...$, האלגוריתם החמדן ישלם מחיר אינסופי, שכן כל הזמן יזיז שרתים. האלגוריתם האופטימלי לעומתו יביא את השרת הרחוק מ־C ל־B, ואז $OPT(\sigma) = d(B,C)$.

על כן, האלגוריתם החמדן אינו תחרותי, וצריך לחשוב כאן על "משהו יותר מתוחכם". זוהי אחת המוטיבציות לאלגוריתם, שלכאורה עושה דבר "קצת מוזר", אותו נתאר בהוכחה - נרשה להזיז יותר משרת אחד לבקשה.

במקרה שתיארנו, אפשר להזיז את השרת מ־A ל־B - נסמן את המרחק ב־x, ונזיז גם את C ב־x (אין שם נקודה, אבל זה יעזור בתיאור האלגוריתם). הסדרת מספיק ארוכה, כך שבשלב מסויים השרת יתקרב לאיזור הבקשות, ויגיע לבסוף לנקודה B (או A בהתאם למצב של השרת השני באותו זמן).

לא באמת נריץ את האלגוריתם, אלא נריץ אותו כסימולציה ברקע, ונזיז "באמת" את השרת רק כאשר הוא באמת יהיה השרת שימלא את הבקשה (כאשר הסימולציה תגיע ל־B או A).

היתרון של האלגוריתם הזה - הוא ללא זיכרון, הוא לא צריך לשמור את הקונפיגורציות הקודמות.

רעיון של אלגוריתם רנדומי (מגריל אילו שרתים לקחת כאשר ההסתברות היא בהתאם למרחקים) כאן גם עובד, בערך באותה המידה.

אלגוריתם 19 *Double Coverage*

שני מקרים:

1. הבקשה מצד אחד של כל השרתים - אז נזיז את השרת הקרוב.
2. אחרת, הבקשה r בין s_i ל־s_{i+1}, בה"כ s_i קרוב יותר. נסמן $\delta = d(s_i, r)$. אזי נזיז את s_i לשרת את הבקשה, ונזיז את s_{i+1} מרחק δ לכיוון הבקשה.

הוכחה: נגדיר שבעת קבלת בקשה, קודם כל היריב ממלא את הבקשה, ולאחר מכן האלגוריתם ממלא אותה.

נסמן ב־t את הזמן מיד לאחר הבקשה ה־t, כנ"ל לגבי $t+1$, וכן נגדיר t' זמן ביניים - הזמן שלאחר היריב שירת את הבקשה, ולפני שהאלגוריתם שירת אותה.

לצורך ההוכחה נשתמש בפונקציית פוטנציאל[21]:
מגדירים פונקציית פוטנציאל:

$$\Phi : M^k \times M^k \to \mathbb{R}^+$$

כאשר M^k הראשונה היא קבוצת כל הקונפיגורציות (=מיקום השרתים) של האלגוריתם, השניה היא קבוצת כל הקונפיגורציות של היריב.
נדרוש מהפונקציה Φ לקיים את התכונות הבאות $\forall t$:

- $$\Phi_{t'} - \Phi_t \leq k \cdot \Delta ADV_t$$

  כאשר ΔADV_t הוא מחיר היריב על הבקשה ה־t.

- $$\Phi_{t+1} - \Phi_{t'} \leq -\Delta ON_t$$

  כאשר ΔON_t הוא מחיר האלגוריתם על הבקשה ב־t.

למה 2.36 אם קיימת פוטנציאל כנ"ל Φ, אזי ON הוא k־תחרותי.

הוכחה:

$$\begin{aligned}
&\Phi_{t+1} - \Phi_t \leq k \cdot \Delta ADV_t - \Delta ON_t \\
&\Rightarrow \sum_{0 \leq t \leq f} \Phi_{t+1} - \Phi_t \leq \sum_{0 \leq t \leq f} \left(k \cdot \Delta ADV_t - \Delta ON_t\right) \\
&\Rightarrow -\Phi_0 \leq \overbrace{\Phi_f}^{\geq 0} - \Phi_0 \leq k \cdot ADV\left(\sigma\right) - ON\left(\sigma\right) \\
&\Rightarrow ON\left(\sigma\right) \leq k \cdot ADV\left(\sigma\right) + \Phi_0
\end{aligned}$$

כאשר Φ_0 קבוע אדיטיבי התלוי בקונפיגורציות ההתחלתיות. ■

איור
s_i ־ מיקום שרתי האלגוריתם.
a_i ־ מיקום שרתי היריב
נגדיר פונקציות שירכיבו את הפוטנציאל[22]:

$$\begin{aligned}
\Psi &= \sum_{i=1}^{k} d\left(s_i, a_i\right) \\
\Theta &= \sum_{1 \leq i < j \leq j} d\left(s_i, s_j\right) \\
&\Rightarrow \Phi = k \cdot \Psi + \Theta
\end{aligned}$$

אנו מניחים וסדר השרתים לא משתנה עם הזמן, כי אם כן אפשר להחליף זאת בהזזת שרתים שעלותם קטנה או שווה לעלות הזו.
נראה כי Φ אכן מקיימת את התכונות המבוקשות:

[21] זוהי שיטה נפוצה מאוד בתחום הזה ־ סוג של הוכחת אינדוקציה.
[22] בתרגיל בבית מראים זאת, כלומר מוצאים סדרה שתקיים את זה.

1. נניח שהיריב משרת את הבקשה ה־r בעזרת שרת a_l. אזי:

$$
\begin{aligned}
&ADV_t = d\left(a_l, r\right) \\
&\Theta_{t'} - \Theta_t = 0 \ \text{(the algorithm's servers do not move)} \\
&\Psi_{t'} - \Psi_t \leq d\left(a_l, r\right) \\
&\Rightarrow \Phi_{t'} - \Phi_t \leq k \cdot \Delta ADV_t
\end{aligned}
$$

2. נחלק למקרים:

(א) הבקשה מצד אחד של כל השרתים: *איור*. אזי:

$$
\begin{aligned}
&\Delta ON_t = d\left(s_k, r\right) \\
&\Psi_{t+1} - \Psi_{t'} = -d\left(s_k, r\right) \\
&\Theta_{t+1} - \Theta_{t'} = (k-1)\, d\left(s_k, r\right) \ \ (s_k \text{ is now further apart from the rest of the servers}) \\
&\Rightarrow \Phi_{t+1} - \Phi_{t'} = -k \cdot d\left(s_k, r\right) + (k-1)\, d\left(s_k, r\right) = -d\left(s_k, r\right) = -\Delta ON_t
\end{aligned}
$$

(ב) הבקשה r בין s_i ל־s_{i+1}, בה"כ s_i קרוב יותר. *איור*

$$
\begin{aligned}
&\Delta ON_t = 2\delta \\
&\Theta_{t+1} - \Theta_{t'} = 2\delta \ \text{(for the rest, one server is further, other closer, so only diff is for } s_i, s_{i+1}) \\
&\Psi_{t+1} - \Psi_{t'} \leq 0 \ \text{(either } a_i \text{ or } a_{i+1} \text{ one will make worse, other better)} \\
&\Rightarrow \Phi_{t+1} - \Phi_{t'} \leq -2\delta = -\Delta ON_t
\end{aligned}
$$

■