page 1

به نام خدا

مرضیه علیدادی ـ 810101236

تکلیف 7 الگوریتم پیشرفته

(1)

الف) این گراف $G=(V,E)$ را درنظر می‌گیریم. هر node از گراف را به صورت تصادفی و با احتمال یکسان و مساوی، در یکی از مجموعه node های V_1 یا V_2 قرار می‌دهیم. با این کار، گراف ما به دو بخش تقسیم می‌شود و یک cut حاصل می‌شود. نام آن را C می‌گذاریم. پس از این الگوریتم تصادفی برای محاسبه max-cut استفاده می‌کنیم.

ب) مقدار تقریب الگوریتم قسمت قبل: $|C| \geq \frac{1}{2}|C^*|$ ← اثبات:

$|C^*|$ مقدار بهینه‌ی max-cut

$|C|$ مقدار بدست آورده الگوریتم برای max-cut

+ گفتیم که هر یک از node ها را با احتمال مساوی در یکی از V_1 یا V_2 قرار می‌دهیم. پس هر یال $e \in E$ از گراف را که درنظر بگیریم، احتمال قرارگیری آن یال در C به این شکل خواهد بود:

$$P[e \in C] = \frac{1}{2} \quad (1)$$

+ یک متغیر تصادفی به نام X_e، با دامنه‌ی $\{0,1\}$ تعریف می‌کنیم. که 1 بودن آن، نشان دهنده‌ی این است که یال e در کات C قرار گرفته است. و 0 بودن آن نشان دهنده‌ی این است که یال e در کات C قرار نگرفته است.

$$X_e \begin{cases} 0 & \rightarrow e \notin C \\ 1 & \rightarrow e \in C \end{cases} \quad (2)$$

پس اندازه‌ی کات C بدست آمده توسط الگوریتم به این شکل خواهد بود:

$$(3) \quad |C| = \sum_{e \in E} X_e$$

+ درنهایت، امید ریاضی اندازه‌ی C به این شکل خواهد بود:

$E(X+Y)=E(X)+E(Y)$ خاصیت امید ریاضی

$$E[|C|] \overset{\text{طبق (3)}}{=} E\Big[\sum_{e \in E} X_e\Big] = \sum_{e \in E} E[X_e] \overset{\text{طبق (2)}}{=} \sum_{e \in E} P[e \in C] \overset{\text{طبق (1)}}{=} \sum_{e \in E} \frac{1}{2} = \frac{|E|}{2} \overset{(|E| \geq |C^*|)}{\geq} \frac{1}{2}|C^*|$$

← مقدار تقریب این الگوریتم، $\frac{1}{2}$ است. ✓

Page2

② الف)

⊕ برای سادگی فرض می‌کنیم هر گره با k تا گره‌ی دیگر همسایه است. (یال مستقیم دارد)

⊕ به طور تصادفی از بین گره‌ها با احتمال یکنواخت و مساوی، یک سری از آن‌ها را در مجموعه‌ی V' قرار می‌دهیم. یعنی هر کدام از node ها با احتمال $\frac{1}{2}$ برای قرارگیری در V' انتخاب می‌شود، به شرطی که هیچ یک از همسایه‌های آن node، در V' قرار نگرفته باشد.

⊕ پس برای هر گره‌ی v_i، یک متغیر تصادفی به نام X_i در نظر گرفته می‌شود، که دارای دامنه‌ی $[0,1]$ است. که این مقادیر با احتمال یکنواخت، به آن تخصیص داده می‌شوند. یک بودن مقدار آن، نشان‌دهنده‌ی قرارگیری گره‌ی v_i، در V' است. و صفر بودن مقدار آن، نشان‌دهنده‌ی عدم قرارگیری این گره در مجموعه‌ی V' است.

$$\begin{cases} P(X_i=1)=\frac{1}{2} \\ P(X_i=0)=\frac{1}{2} \end{cases}$$

⊕ پس برای هر گره که در V' نیست، X متناظرش برابر 0 است. و برای هر گره‌ای که در V' است، X متناظرش برابر 1 است، و X متناظر با همه‌ی همسایه‌هایش برابر 0 است.

ب)

⊕ یک event به این شکل تعریف می‌کنیم:

$S[i] \longrightarrow$ بیانگر این اتفاق است که گره‌ی v_i در مجموعه‌ی V' قرار گرفته است و k همسایه‌ی آن در V' قرار نگرفته‌اند:

$$\Longrightarrow S[i] = X_i \cap \Big(\bigcap_{\substack{j\neq i \\ v_j \in V'}} \overline{X_j} \Big)$$

← احتمال وقوع این event:

$$P[S[i]] = \frac{1}{2} \times \left(\frac{1}{2}\right)^k = \left(\frac{1}{2}\right)^{k+1}$$

⊕ امید ریاضی اندازه‌ی مجموعه گره‌های V' به این شکل است: (فرضاً $|V|=n$ است)

$$E[|V'|] = \sum_{i=1}^{n} P[S[i]] = n \times \left(\frac{1}{2}\right)^{k+1}$$

← ضریب تقریب:

V' حداکثر شامل کل گره‌هاست: $\boxed{c^* \leq n}$

$$\frac{c}{c^*} \geq \frac{c}{n} = \frac{n\left(\frac{1}{2}\right)^{k+1}}{n} = \left(\frac{1}{2}\right)^{k+1} \longrightarrow c \geq \left(\frac{1}{2}\right)^{k+1} c^*$$

ضریب تقریب این الگوریتم برابر $\left(\frac{1}{2}\right)^{k+1}$ است. α

page3

③ الف) ILP :

⊕ یک متغیر X_i تعریف می‌کنیم :

$X_i \xrightarrow{=1}$ مجموعه‌ی S_i در M' وجود دارد (یعنی برای cover کردن اعضای A ، انتخاب می‌شود)

$X_i \xrightarrow{=0}$ مجموعه‌ی S_i در M' وجود ندارد (یعنی برای cover کردن اعضای A ، انتخاب نمی‌شود)

⊕ تابع هدف :

$$\min \sum_{i=1}^{k} X_i W_i$$

مجموع هزینه‌ی S_i های موجود در M'

⊕ شروط :

$$\forall_{j=1}^{m} a_j \in A \ , \sum_{\substack{1 \le i \le k \\ a_j \in S_i}} X_i \geq 1$$

$$0 \le x \le 1 \ , \ \text{int}$$

ب) LP relaxation :

$$\min \sum_{i=1}^{k} X_i W_i$$

$$\forall_{j=1}^{m} a_j \in A \ , \sum_{\substack{1 \le i \le k \\ a_j \in S_i}} X_i \geq 1$$

$$0 \le x_i \le 1$$

⊕ مسئله‌ی موجود در این سوال همان set cover است. که یک مسئله‌ی NP_hard است. با توجه به اینکه ILP یک NPC است، می‌توان این مسئله را در ILP مدل کرد. و با توجه به اینکه LP polynomial است، می‌توان این مسئله را به LP ، reduce کرد. و جوابی که LP می‌دهد، معتبر خواهد بود ← و با توجه به اینکه feasible area مربوط به LP ، بزرگتر از ILP است ، جوابی که LP می‌دهد بزرگتر مساوی جوابی است که ILP می‌دهد.

« ـــــ پس یک الگوریتم randomized بر روی LP در نظر می‌گیریم و جواب تقریبی این مسئله‌ی set cover را بدست می‌آوریم.

page 4

ادامه‌ی ③

ج)

⊕ در این مسئله با توجه به مقدار X_i، تصمیم می‌گرفتیم که آیا مجموعه‌ی S_i را انتخاب کنیم یا نه. در LP، مقدار این متغیر، بین ۰ و ۱ است. و هر مقداری (نه لزوماً ۰ و یا ۱) می‌تواند داشته باشد. پس در این حالت تصمیم‌گیری راحت نخواهد بود.

⊕ یک روش این است که روی X_i شرط بگذاریم که اگر از $\frac{1}{2}$ کمتر است، مقدارش را برابر ۰ در نظر بگیریم و S_i را انتخاب نکنیم. و اگر از $\frac{1}{2}$ بزرگتر است، مقدارش را برابر ۱ در نظر بگیریم و S_i را برای قرارگیری در M' انتخاب کنیم.

← البته به جای $\frac{1}{2}$ می‌توانیم مقدار دیگری بین ۰ و ۱ به عنوان این آستانه در نظر بگیریم برای مقایسه با X_i.

← این روش و این گرد کردن، نتیجه‌ی خوبی نخواهد داشت. مثلاً اگر داشته باشیم: $X_i = \frac{1}{K^2}$ با افزایش K، از یک حد به بعد، مقدار X_i از هر آستانه‌ای کمتر خواهد بود. و مقدار گرد شده‌ی X_i، همیشه برابر با ۰ خواهد شد. و هیچ S_iای برای قرارگیری در M' انتخاب نخواهد شد.

← پس این، روش مناسبی برای گرد کردن نیست. و باید روش دیگری استفاده کرد.

⊕ می‌توان به X_iها به عنوان یک سری احتمال نگاه کرد. یعنی مقدار X_i این را نشان می‌دهد که S_i با چه احتمالی در M' قرار می‌گیرد.

← پس یک سری عدد رندوم بین ۰ و ۱ تولید می‌کنیم و با X_i مقایسه می‌کنیم. و هر S_i را با احتمال X_i برای قرارگیری در M' انتخاب می‌کنیم.

← در این انتخاب‌ها، در نهایت [2] باید حواسمان باشد که هزینه‌ی [1] انتخاب S_iهای M'، مناسب باشد. و این مجموعه‌های انتخاب شده، حتماً همه‌ی اعضای A را پوشش بدهد.

⊕ امید ریاضی این الگوریتم:

$$E = \sum_{i=1}^{K} w_i X_i$$

⊕ احتمال اینکه جواب بدست آمده، کل اعضای A را پوشش بدهد یا ندهد:

← یکی از a_jها را در نظر می‌گیریم. فرض کنید a_j، در مجموعه‌های S_1 و S_2 و ... و S_z وجود داشته باشد. احتمال اینکه a_j در جواب نهایی الگوریتم، پوشش داده نشده باشد، به این شکل است:

$C_j \to 1$: a_j پوشش داده شده است.

$C_j \to 0$: a_j پوشش داده نشده است.

$$P(C_j = 0) = (1-X_1)(1-X_2)\cdots(1-X_z)$$

S_1 در جواب انتخاب نشده باشد. — S_2 در جواب انتخاب نشده باشد — S_z در جواب انتخاب نشده باشد

page5

ادامه‌ی ③

ادامه‌ی ج)

← ما باید سعی کنیم که این احتمال را ضعیف کم کنیم. و این احتمال که جواب ما set cover است و تمام اعضای A را پوشش می‌دهد را به 1 نزدیک کنیم.

⊕ یک upper bound برای این احتمال بدست می‌آوریم:

$$P(c_j=0) = (1-x_1)(1-x_2)\cdots(1-x_z) \underset{(1-x)\leqslant e^{-x}}{\leqslant} e^{-x_1}\cdot e^{-x_2}\cdots e^{-x_z} = e^{-(x_1+x_2+\cdots+x_z)} \leqslant e^{-1}$$

جمع این احتمال‌ها (ها x) حداقل برابر 1 است.

⊕ برای هر یک از a_j ها، این احتمال گفته شده در بالا وجود دارد.

← احتمال set cover نبودن جواب این الگوریتم، به این شکل خواهد بود:

$$\sum_{j=1}^{m} P(c_j=0) \leqslant \sum_{j=1}^{m} e^{-1} = m\cdot e^{-1}$$

← این احتمال، زیاد است. پس باید چندین بار الگوریتم را تکرار کنیم تا این احتمال کوچک شود. و احتمال موفقیت الگوریتم به حد مطلوبی برسد.

احتمال موفقیت: $1-\frac{m}{e}$ ← به 1 نزدیک است

⊕ t بار الگوریتم را تکرار می‌کنیم:

← احتمال پوشش داده نشدن هر a_j در جواب الگوریتم:

$$\left(P(c_j=0)\right)^t \leqslant (e^{-1})^t = e^{-t}$$

← احتمال set cover نبودن جواب الگوریتم:

$$\sum_{j=1}^{m} \left(P(c_j=0)\right)^t \leqslant \sum_{j=1}^{m} e^{-t} = m\cdot e^{-t}$$

⊕ تعداد تکرار را به این شکل در نظر می‌گیریم:

$$t = \ln(m) + O(1)$$

احتمال شکست ←: $m\cdot e^{-t} = m\cdot e^{-(\ln(m)+O(1))} = m\, e^{-(\ln(m))}\, e^{O(1)} = m\cdot m^{-1}\, e^{O(1)} \simeq e$

← به یک عدد ثابت تبدیل شد. که مناسب است. و احتمال موفقیت خوبی با این تعداد تکرار خواهیم داشت.

page 6

ادامه‌ی ③

ادامه‌ی ج)

⊕ احتمال مناسبی بدست آوردیم. حال باید بررسی کنیم که آیا هزینه‌ی مجموعه‌ی بدست آمده به عنوان جواب، همچنان خوب است یا خیر.

که هر بار تکرار الگوریتم، می‌تواند هزینه را حداکثر $\sum_{i=1}^{k} X_i W_i$ افزایش بدهد.

که پس با توجه به این تعداد تکرار، که از $O(\log(m))$ است، جواب بدست آمده پس از این تعداد تکرار، حداکثر برابر $\underbrace{result(LP)}_{\text{جوابی که LP به ما می‌دهد}} \times O(\log m)$ است.

⊛ ← پس یک الگوریتم و تکرار مناسب برای آن بدست آوردیم، که با احتمال نزدیک به 1، جواب درست می‌دهد. و هزینه‌اش هم از بالا به $\underbrace{O(\log m)}_{\text{مقدار تقریب این الگوریتم}}$ برابر جواب LP محدود شده است.

✓

5

⊕ از روش LP_relaxation استفاده می‌کنیم، و rounding را randomized انجام می‌دهیم.

⊕ یک متغیر y_i متناظر با هر متغیر x_i در SAT، درنظر می‌گیریم، که نشان می‌دهد آیا مقدار آن true یا false است.

⊕ برای هر کلاز c از SAT، یک متغیر S_c تعریف می‌کنیم. که نشان می‌دهد آیا آن کلاز satisfy شده است یا خیر.

⊕ LP_relaxation مربوط به این مسئله، به این شکل خواهد بود:

objective function ـه:

$$\max \sum_{c \in \mathcal{C}} S_c$$

مجموعه‌ی کلازها

ـه شروط:

$$\forall c \in \mathcal{C} \quad S_c \leq \sum_{i \in c^+} y_i + \sum_{j \in c^-} (1-y_j)$$

متغیرهایی که خودشان عیناً در کلاز c وجود دارند.

متغیرهایی که complement شان در کلاز c وجود دارد.

$$\forall_{i=1}^{n}, \forall_{c \in \mathcal{C}} \quad 0 \leq S_c, y_i \leq 1$$

متغیرهای شرکت شده در SAT را $x_1, x_2, \ldots, x_n$ فرض می‌کنیم.

⊕ الگوریتم به این شکل خواهد بود:

به هر متغیر x_i، با احتمال y_i، مقدار true نسبت می‌دهیم.

ادامه ⑤

⊕ احتمال اینکه یک کلاز satisfy نشود به این شکل توصیف می‌شود:

$$P\left[\begin{matrix}\text{satisfy}\\ \text{نشدن کلاز}\end{matrix}\right] = \underbrace{\prod_{i\in C^+}(1-y_i)\cdot\prod_{j\in C^-} y_j}_{} \underset{\substack{\uparrow\\ 1-t\leq e^{-t}}}{\leq} \prod_{i\in C^+} e^{-y_i}\cdot\prod_{j\in C^-} e^{-(1-y_j)} =$$

اگر متغیرهای C^+، false شوند،
و متغیرهای C^-، true شوند،
کلاز satisfy نمی‌شود

$$= e^{-\left(\sum_{i\in C^-} y_i + \sum_{j\in C^+}(1-y_j)\right)} \leq e^{-S_c}$$

⊕ پس احتمال اینکه کلاز C، satisfy شود، به این شکل خواهد بود:

$$P\left[\begin{matrix}\text{satisfy}\\ \text{شدن کلاز}\end{matrix}\right] \geq 1-e^{-S_c} \underset{\substack{\uparrow\\ S_c\leq 1}}{\geq} S_c(1-e^{-1})$$

پس این الگوریتم $(1-\frac{1}{e})$-تقریب است. ∅