פרויקט אופטימיזציות - הגנה על שטח מפני פורצים

עומרי פרידנטל ודניאל קגנוביץ'

27.1.2020

<u>מבוא:</u>

בפרויקט זה, הגדרנו בעייה, ופתרנו אותה, דבר זה מוצג בחלק 1. לאחר מכן כיוון שהרגשנו שזה לא מספיק, הרחבנו את הבעיה - חלק 2 והראינו איך משתמשים בחלק 1 כדי לפתור אותה.
בחלק 1, ו2 מוצגים לעומק הבעיות ופתרונם (הגדרת בעיית האופטימיזציה, ופתרונה).
חלק 3 מסביר על מבנה הקוד בצורה שיהיה יותר קל לקרוא אותו לאחר קריאת המסמך הזה.
חלק 4 נותן הסבר קצת על התוצאות בפועל.

<u>חלק 1:</u>

1. הבעיה:

Security structure optimization

“Given a grid-based terrain, construct obstacles to make an attacker's task most difficult.”

<u>קלט:</u>

<u>נתונים אודות הבסיס:</u>

- **לוח בסיס B עם n משבצות, המתאר שטח עבודה, עליו היינו רוצים להגן מפני תוקפים אפשריים.**
  למשל:
  - שטח של מתקן צבאי - נרצה להציב שומרים.
  - בנק המכיל חדר כספות - נרצה לשים חיישני אבטחה.
  - אזור גאוגרפי המכיל כבישים, בזמן מרדף משטרתי אחר עבריין.- נרצה להציב ניידות משטרה לפני שהוא נכנס לאזור זה.

<u>נתונים סטטיסטים אודות התוקף:</u>

- **וקטור התחלה $r \in (0,1)^n$ - הסתברויות כניסות אפשריות של תוקף לבסיס. סכום r יהיה כמובן 1.**
  למשל, עבור בסיס עם 6 משבצות ושתי כניסות במיקומים 2,5 הוקטור יראה כך:
  $r = (0, 0.3,\ 0,\ 0,\ 0.7,\ 0)$

- **מטריצה $M \in (0,1)^{n\,x\,n}$ המתארת הסתברויות לצעדים אפשריים של התוקף. $M_{i,j}$ היא ההסתברות למעבר בין משבצת i ל j . המטריצה חייבת להיות סטוכסטית - סכום כל שורה יהיה 1.**
  למשל, $M_{2,3}$ הוא הסיכוי שהתוקף ינוע מהמשבצת ה2 ל3. ערכים אלו הם שקלול של תנאי הבסיס וסטטיסטיקה לגבי פעולת התוקף - כלומר עבור מעבר לא חוקי ההסתברות תהיה 0. אפשרי שבבסיס מלבני רוב המטריצה תהיה אפסים.

משאבי ההגנה:

- $k \in N$ מספר ההגנות הזמינים לרשותנו. הגנה היא שיבוץ מכשול על משבצת כלשהי בלוח אשר תופסת תוקף.
לדוגמא, אם למתקן צבאי יש כוח אדם של 2 שומרים - $k = 2$.

פלט:

- $D = \{d_1, d_2, ..., d_k\}$ **שיבוץ ההגנות המביא למקסימום את הסיכוי לתפוס את הפורץ.**
למשל בדוגמא של שדה התעופה, נקבל D מיקומים בהם עלינו לשים את הניידות על מנת לקבל אבטחה מקסימלית.
הערה: מדובר על $k << n$ כלומר מספר ההגנות הזמינות קטן משמעותית מגודל הלוח, ו $k << |\{r_i > 0\}|$ כלומר גם קטן משמעותית ממספר הכניסות ללוח (אחרת יכלנו להציב הגנה בכל כניסה).

2. הפתרון:

1. הגדרת בעיית האופטימיזציה:

נרצה לחשב **באיזה k משבצות** למקם סנסורים כך ש**ההסתברות לתפוס את הפורץ - ההסתברות שפורץ ידרוך באחת מהן** תהיה **מקסימלית.**
כלומר אם נגדיר את $p = (p_1, ..., p_k)$ להיות משבצות המסלול של הפורץ לכל k, כאשר משבצת הכניסה מתאימה למוגדר בr, והמסלול נלקח על פני כל המסלולים האפשריים המתאימים להסתברויות המוגדרות בM.
ונגדיר $d_1, ..., d_l$ שיבוצי הסנסורים שלנו -
נסמן $d_j \in P$ אם d_j מופיע במסלול p כלשהו עבור k כלשהו.
נרצה $max\, Pr_P\, (d_1 \in P \vee ... \vee d_l \in P))$.
נפשט ראשית לסנסור אחד d ולאחר מכן נכליל.

2. נצטרך לחשב את ההסתברויות כתלות במשתנים - נשתמש בשרשראות מרקוב.

שרשרת מרקוב מאפשרת לנו פה לחשב את הסיכוי שפורץ ימצא במשבצת x_j בהנתן שהמסלול שלו עד כה היה $x_1, ..., x_{j-1}$. כלומר עבור מסלול באורך k, $Pr\, (p_k = x_k | p_1 = x_1 \wedge ... \wedge p_{k-1} = x_{k-1})$.
ניתן להוכיח (http://u.math.biu.ac.il/~amirgi/ariel_notes2.pdf) שבסופו של דבר -

$$Pr(p_k = j \mid p_1 = i) = (M^k)_{i,j}$$

ולכן אם נקח בחשבון גם את הוקטור ההתחלתי r:

$$Pr(p_k = j) = \sum_{i=1}^{n} Pr(p_k = j \mid p_0 = i) \cdot Pr(p_0 = i) = \sum_{i=1}^{n} (M^k)_{i,j} \cdot r_i$$

כעת נקח בחשבון גם את העובדה שאורך המסלול k אינו ידוע ונקבל עבור $p = (p_1, ..., p_k)$:
כאשר נסמן

$$Pr(j \in P) = \sum_{k=1}^{\infty} Pr(p_k = j) = \sum_{k=1}^{\infty} \sum_{i=1}^{n} (M^k)_{i,j} \cdot r_i$$

לכן נקבל שאם נרצה לשים סנסור d כך שההסתברות $Pr(d \in P)$ יהיה ניתן לחשב $Pr(d \in P)$.

3. ננסח כבעיית אופטימיזציה:
נגדיר n משתנים בינארים, $D_1,\ ...,\ D_n$ כאשר כל $D_i = 1$ אמ"מ נציב סנסור במשבצת ה i . כעת נרצה:

$$max\ \ D_1 \cdot Pr(1 \in P)\ + ... + D_n \cdot Pr(n \in P)$$

תחת האילוץ $D_1 + ...\ +\ D_n\ <=\ k$.

4. ננסה לשלב את בעיית האופטימיזציה עם החישוב של ההסתברויות:
נסמן F פונקציית הרווח אותה ננסה למקסם:

$$F\ =\ \ D_1 \cdot Pr(1 \in P)\ + ... + D_n \cdot Pr(n \in P)$$

$$F = \sum_{j=1}^{n} D_j \cdot \sum_{k=1}^{\infty} \sum_{i=1}^{n} (M^k)_{i,j} \cdot r_i = \sum_{k=1}^{\infty}\ \ (\sum_{j=1}^{n} D_j \cdot \sum_{i=1}^{n} (M^k)_{i,j} \cdot r_i)$$

נסתכל רק על הביטוי שבתוך סיגמה של k - כלומר ההסתברויות בהנחה שהמסלול הוא באורך k:

$$F_k = \sum_{j=1}^{n} D_j \cdot \sum_{i=1}^{n} (M^k)_{i,j} \cdot r_i =\ \ D_1 \cdot r_1 \cdot ({M_{11}}^k + ... + {M_{n1}}^k) + ... + D_n \cdot r_n \cdot ({M_{1n}}^k + ... + {M_{nn}}^k)$$

כלומר ניתן לראות שאם נחשב את M^k ו"נקבע אותה" וכנל ל r , אנחנו סה"כ רואים פה גישה לכל איבר ב M^k פעם אחת וגישה לכל איבר ב r פעם אחת. ביחד נוצר משוואת $F_k = a_1 \cdot D_1 + ... + a_n \cdot D_n$ עבור הקבועים $a_i = r_i \cdot ({M_{1i}}^k + ... + {M_{ni}}^k)$.
ניתן לפתירה על ידי שיטות שלמדנו בכיתה - (הכנס שם של שיטה כאן).
כעת נרצה לשלב את זה לכל k :
ניתן בעצם לעשות כך:

$$F = \sum_{k=1}^{\infty} F_k = \sum_{k=1}^{\infty} a_{k1} \cdot D_1 + ... + a_{kn} \cdot D_n = (\sum_{k=1}^{\infty} a_{k1}) \cdot D_1 + .. + (\sum_{k=1}^{\infty} a_{kn}) \cdot D_n$$

עבור a_{ki} שיהיה a_i ממקודם עבור k זה.
ובעצם גם זה אותו סגנון של משוואה שלמדנו איך ממקסמים.
הבעיה - הקבועים הם חישוב של טור אינסופי של קבועים.
הפתרון -
אופציה אחת היא להגביל את אורך המסלול, למשל $k <= n^2$ (כלומר מניחים שהפורץ לא חוזר על אף מעבר - אך דבר זה גורם שאנחנו מניחים הנחה לא ידועה על מסלול הפורץ
אופציה דומה אך טיפה יותר טובה היא לחשב את הגבול של טור הקבועים (הסבר למה זה מתכנס),
כלומר לחשב לכל $1 <= i <= n$ את $\alpha_i = \sum_{k=1}^{\infty} a_{ki} = \lim_{k\to\infty} \sum_{j=1}^{k} a_{ji}$

ואז לנסות למקסם עבור $F = \alpha_1 \cdot D_1\ + ... + \alpha_n \cdot D_n$ **, תחת האילוץ** $D_1 + ...\ +\ D_n\ <=\ k$ **.**
נשים לב שזה פשוט אומר לחשב את $A\ =\ \{\alpha_i\ |\ 1 <= i <= n\}$ ונקח k פעמיים argmax על A.
וכך אכן קורה בקוד שלנו.
אלגוריתמית ניתן להגדיר את הפתרון שלנו:

$Maximize\ -\ q1\ (M,\ r)$:

$D\ = \{\}$

$A\ = \{\lim_{k\to\infty} \sum_{i=1}^{n} (M^k)_{i,j}\ \ |\ 1 <= j <= n\}$

$for\ i\ = 1\ to\ \ k$:

$best - j \;=\; argmax\, A$

$D.add(\, best - j)$

$A \;=\; A - \{best - j\}$

$return\ D$

כאמור, הקוד פשוט עושה k פעמים argmax על A.

<u>חלק 2:</u>

1. החלטנו לשדרג את הבעיה המקורית לבעיה הבאה:

הקלט על הלוח נשאר זהה - $M,\ r$.

יש לנו עדיין k סנסורים זמינים, אך נוסיף למשאבי ההגנה:

- **$l \in N$ מספר חסימות הדרכים הזמינות לרשותנו. חסימת דרך היא שיבוץ מכשול על קשת (i,j) כך שהפורץ לא יכול לעבור מהמשבצת ה-i למשבצת ה-j.**

נגדיר ש l צריך להיות קטן משמעותית מN, אך הרבה יותר זמין מk. ברמת ה $k^2 = l$ בערך. כלומר משאב זה יקר אך עדיין הרבה יותר זמין מסנסורים.

למשל, בדוגמא של מרדף משטרתי בשטח של כבישים, סנסורים יהיו ניידות משטרה ומחסומי דרכים יהיו מחסומים פיזים על קטעי כביש, אך שאינם מאוישים - ולכן יותר זמינים.

נציין שמטרת מחסומי הדרכים תהיה להסיט את הפורץ לדרכים אחרות שדרכם יותר סביר שיגיעו לסנסורים שנציב.

הפלט יהיה להחזיר את D קבוצת הסנסורים (מוגדרת כמו מקודם), ו E קבוצת החסימות - $E \;=\; \{e_1,\ ...,\ e_l\}$, אשר כאשר נשים את הסנסורים במקומות D, ונחסום את הקשתות E, נקבל סיכוי מקסימלי לתפיסה של הפורץ.

2. הפתרון:

נגדיר את בעיית האופטימיזציה החדשה באופן דומה. נגדיר פונקציה $block(M,\ E)$ אשר מחזירה M' מטריצת סיכויי מעבר לאחר חסימה לפי E.

כעת נגדיר שפונקציית ההצלחה היא אותה F כמו קודם לפי $M',\ r$.

לכן מה שנרצה זה לבחור D, E סנסורים וחסימות כך שנמקסם את F.

ראשית נגדיר כמו שצריך את את block:

נגדיר $block(M,\ e)$ כאשר חוסמים קשת $e = (i,j)$, ההסתברות למעבר הזה יתחלק למקומות האחרים שאפשר לעבור מ-i באופן שבו ההסתברויות נשארות פרופורציונליות אחת לשניה, ועדיין נסכמות ל1. למשל עבור שורה מתוך M:

$$M_{i1} + ... + M_{in} = 1$$

אשר נרצה לחסום את הקשת ה (i,j), נגדיר:

$$if\ M_{ij} = 1:\ \ return\ M' \;=\; M\ (its\ the\ only\ transition\ i,\ we\ don't\ allow\ to\ remove\ it).$$

$$else: M_{ij} = 0,\ for\ each\ k \neq j:\ \ M'_{ik} = M_{ik} / (1 - M_{ij})$$

ויתקיים:

$$\sum_{k=1}^{n} M'_{ik} = M_{ij} + \sum_{k\neq j} M_{ik}/(1-M_{ij}) = 0 + 1/(1-M_{ij}) \cdot \sum_{k\neq j} M_{ik} = 1/(1-M_{ij}) \cdot (\sum_{k=1}^{n} M_{ik} - M_{ij}) =$$

$$1/(1-M_{ij}) \cdot (1-M_{ij}) \; = \; 1$$

ולכן סכום שורה הוא 1, זה עדיין חוקי, ועדיין יהיו הפרופורציות בין ההסתברויות זהות למקודם, כיוון שכפלנו את כולן באותו קבוע.

נגדיר $block(M,\ E)$, להיות $block(block(...block(M,\ e_1), e_2....),\ e_n)$ כלומר הפעלת block של קשת על כל הקשתות אחת אחרי השנייה.

כעת ננסה להראות את הגישה שנקטנו על מנת לייצר פתרון אופטימלי לבעיה המשודרגת, המשתמש בפתרון בבעיה המקורית. כלומר למצוא גם את E בנוסף ל D עם הפונקציה שיצרנו בסעיף הקודם.

האלגוריתם שלנו הוא כזה:

$Maximize - q2\ (M,\ r)$:

$E \; = \; \{\}$

$for\ i \; = 0 \;\; to\ l$:

$\quad D \; = Maximize - q1\ (M_{i-1},\ r)$

$\quad p \; = \; \sum_{j\in D} \lim_{k\to\infty} \sum_{i=1}^{n} (M^k)_{i,j}$

$\quad for\ e\ in\ NxN$:

$\quad\quad M' \; = \; block(M,\ e)$

$\quad\quad p' \; = \; \sum_{j\in D} \lim_{k\to\infty} \sum_{i=1}^{n} (M'^k)_{i,j}$

$\quad\quad if\ p' \; > \; p$:

$\quad\quad\quad worst - e = e, \quad p \; = \; p'$

$\quad E.add\ (\ worst - e\ )$

$\quad M \; = block(M,\ worst - e)$

$return\ (D, E)$

בהסבר מילולי :

נחשב ראשית את הסנסורים הממקסמים את הסיכוי לתפיסה ללא חסימות דרכים.

כעת נחפש איזה קשת אפשר לחסום כך שעם הסנסורים הקיימים, סיכוי התפיסה גדל הכי הרבה, ונחסום אותה.

כעת נחזור על התהליך - נחשב מחדש את הסנסורים המתאימים ללוח שלאחר החסימה, וננסה למצוא את הקשת הבאה לחסום על הסנסורים האלה... וכך האלה l פעמים עד שמילאנו את E.

ניתוח -

נשים לב, שתהליך זה לא נאיבי אשר עובר על כל האפשרויות של סנסורים וחסימות יחד - דבר אשר היה לוקח המון זמן ריצה, אך היה מביא לתוצאה אופטימלית בהכרח.

אנחנו משתמשים בשיטה אשר משתמשת בשיטה מהסעיף הקודם, ומשפרת את התוצאה - ההסתברות לתפיסה על ידי חסימת קשתות.

בכל סיבוב $1 <= i <= l$, p גדול או שווה ל p מהסיבוב הקודם, ולכן בסופו של דבר הגדלנו את הסיכוי לתפיסה. לכן יש פה בהכרח אלגוריתם שמשפר את התוצאה לעומת האלגוריתם הקודם ומשיג תוצאה טובה, אך לא בהכרח מוצא בדיוק את התוצאה האופטימלית מתוך כל האופציות.

חלק 3:

הסבר קצר על הקוד (מתועד גם בתוכו):
כשמריצים את הקוד זה די אינטראקטיבי ומובן, אך באופן כללי זה נותן לבחור N לפיו נגדיר $n = N^2$, שטח ריבועי. ולבחור את k, l . את M, r זה מגריל לפי n כך שיש עד $N/2$ כניסות לא אפסיות ללוח, והכיוון הכללי של פורץ יהיה לכיוון מרכז הלוח (משבצת n/2).
בפירוט יותר:

```
find_sensors(M, r, k)
```

היא בעצם מממשת את האלגוריתם $maximize - q1$.

```
get_sensors_with_roadblocks(M, r, k, l)
```

זה המימוש לאלגוריתם $maximize - q2$.

הפונקציה `node_prob(M,r)` מחזירה בעצם את $A = \{\lim_{k\to\infty} \sum_{i=1}^{n} (M^k)_{i,j} \mid 1 <= j <= n\}$. חישוב הגבול לא מתבצע מתמטית אלא על ידי הגדרת $\varepsilon = 0.0001,\ T = 3$ ובדיקה מתי קורה $\exists\ t < T : |\sum_{i=1}^{n}(M^{k+t})_{i,j} - \sum_{i=1}^{n}(M^k)_{i,j}| < \varepsilon$ ואז מחזירים את $\sum_{i=1}^{n}(M^k)_{i,j}$. (לכל i בנפרד כמובן).

הפונקציה `block_path(M,edge)` מממשת את $block(M, e)$.

הפונקציה generate_random_example שמייצרת M,r רנדומליים לגרף ריבועי בשטח קלט $n = N^2$ אשר נכונים לפי ההגדרה (המטריצה סטוכסטית וr נסכם ל1). משתמשת בחבילתrandom ובפונקציית העזר neigbors אשר מחזירה את הקודקודים הסמוכים בגרף מלבני.
נשים לב שפונקציה זו גם מקבלת target שפה הוא $n / 2$ וזה בוחר את M כך שכאשר הוא שם ערכים רנדומליים לneibors של משבצת i (מגריל ואז מנרמל - שסכומם יהיה 1), זה שם אותם מההסתברות הגדולה לקטנה, מהשכן הקרוב ביותר לtarget עד לרחוק ביותר.
את מספר הכניסות שזה מגריל את ההסתברויות שלהם לr הוא יקבע ל $N/2$.

לבסוף draw_graph_and_results בקוד יוצר סימולציה של הגרף, עם החבילה **networkx**. כאן אין מה לנסות להבין את הקוד אלא יותר את החבילה. כמו שכתוב בפלט של התכנית -
בתמונות של שני החלקים, מופיע הגרף עם ההסתברויות של המעברים לפי M רשומות על הקשתות, והכניסות לפי r מופיעות כקודקודים בצורת משולש (השאר עיגול).
בתמונה הראשונה, הסנסורים שנבחרו לאופטימליים מופיעים בצבע סגול.
בתמונה השנייה, הסנסורים שנבחרו לאופטימליים מופיעים בצבע סגול, והקשתות שנבחרו לאופטימליות לחסימה מופיעות בצבע צהוב - בצורה שכמעט מוחקת אותם.

חלק 4:

הסבר קצר על התוצאות:

הקלט ניתן לבחירה אינטרקטיבית בהרצת הקוד.

התוצאות מתקבלות באופן דומה כשבוחרים אותם פרופוציות $n = f(k),\ n = g(l)$ בין הרצות.

ניתן לראות שכאשר נבחר להגריל גרף עם $n = N^2$ כלשהו, ויחס סביר של $k^4 = n$ (על מספר כניסות קבוע בקוד כמו שאמרנו של $N/2 + 1$).

למשל על $k = 6$ $n = 20x20 = 1600,$ נקבל הסתברות תפיסה של 0.25 , יחסית מכובד בשביל מספר קטן של סנסורים על שטח גדול.

בסעיף ב, נבחר יחס $n = l^2$ ונקבל למשל פה על $l = 20$ והסתברות לתפיסה תגדל ל0.61.

נסיים בתמונות להמחשה של הגרף ותוצאותיו על n=4x4 , עם k=2: הסתברות תפיסה 0.31, אחר כך l = 4, מעלה את ההסתברות ל0.66. התמונות בצד ימין ושמאל בהתאמה.

<u>הערה למי שמריץ</u>: הכיתובים על הגרף הם בגודל פונט קבוע ולכן הגרף יהיה קריא רק לגדלים קטנים שלו. (אבל לכל גודל האלגוריתם עובד ורץ כמו שצריך).

תודה על הקריאה (: