Azrieli
College of Engineering
Jerusalem
עזריאלי
מכללה אקדמית להנדסה
ירושלים

המחלקה להנדסת תוכנה

פרוייקט גמר – תשע"ח

הפרדה מינימלית ומאוזנת של קודקודים ברכיב קשירות

Minimal balanced node separator

מאת: עמרי מזרחי

מנחה אקדמי: ד"ר יהודה חסין אישור: תאריך:

רכז הפרויקטים: ד"ר שפנייר אסף אישור: תאריך:

עבודה משותפת עם הפרויקט של עדי טיירי

Ramat Bet Hakerem, P.O.B 3566, Jerusalem 9103501 רמת בית הכרם, ת.ד. 3566, ירושלים מיקוד
Tel. +972-2-6588000 | Fax. +972-2-6588001 | Email: info@jce.ac.il | Web: www.jce.ac.il

מערכות ניהול הפרויקט:

#	מערכת	מיקום
1	מאגר קוד	https://github.com/mazmaz2k/Modular-Construction-of-Minimal-Models.git
2	יומן	https://calendar.google.com/calendar?cid=azlodGIxNG5qaHRldGU2bW1ndmk2NTlhdDhAZ3JvdXAuY2FsZW5kYXIuZ29vZ2xlLmNvbQ
5	סרטון גירסת אלפא	https://drive.google.com/open?id=14dDiWowsQS__SQtMxJCg7GE_SIzpWYT6

תקציר

במסגרת פרויקט הגמר במחלקה להנדסת תוכנה בעזריאלי (JCE), הוטלה עלי המטלה לבצע פרויקט במסגרת של – 400 שעות .

הפרויקט שאעשה הוא פרויקט מחקרי שנעשה בשיתוף פעולה עם הסטודנט עדי טיירי והמנחים שלנו ד״ר יהודה חסין ופרופ׳ רחל בן אליהו זהרי, כאשר בפרויקט של עדי נעסוק בחלק של מבנה הנתונים של הפסוקיות, ובפרויקט זה נעסוק בחלק הגרפי של האלגוריתם, גרף זה נבנה ממבנה הנתונים של הפרויקט של עדי טיירי.

הבעיה בהינתן נוסחת SAT חיובית יש למצוא מודל מינימלי לנוסחה .
לאחרונה הציעו ב [1] אלגוריתם המוצא מודל מינימלי שתלוי בגודל רכיב הקשירות הגדול ביותר. כל נוסחה ניתנת לתרגום לגרף.
אנו נממש את האלגוריתם כפי שמוצע במאמר [1], מטרת האלגוריתם [1] הוא מציאת מודל מינימלי לסט חוקים בצורת CNF תוך כדי שיפור זמן הריצה של האלגוריתם.
מציאת המודל המינימלי תעשה על ידי בנית גרף לסט החוקים עפ״י סדר מסוים וזמן הריצה של האלגוריתם תלוי בגודל רכיבי הקשירות של הגרף. לכן אם נצליח להקטין את גודל רכיבי הקשירות בגרף נצליח להקטין את זמן הריצה של [1] .
כדי לפרק את רכיב הקשירות נשתמש במאמר [40] מאמר זה מאפשר לנו לדעת איזה קודקודים כדאי להוריד ע״מ שרכיב הקשירות, שגודל אלגוריתם [1] תלוי בו, יתפרק באופן מאוזן.

מתברר שבהרבה בעיות קשות כל הגרף הוא רכיב קשירות אחד גדול. אנו ננסה לפרק את רכיב הקשירות הגדול ע״מ להקטין את זמן הריצה. אנו נציע כפתרון את אלגוריתם [40] כאשר מטרתו היא מציאת קשתות שיפרקו את הגרף, ונשתמש במידע שאלגוריתם זה נותן בשביל למצוא קדקודים שאם נסיר אותם מהגרף נצליח לפרק את רכיב הקשירות הגדול ביותר בגרף, ובכך להקטין את זמן הריצה של אלגוריתם [1] .

ניהול שני הפרויקטים דורשים המון סדר ואחריות, ולכן יש לנו שתי יחידות מפתח להתייחס אליהם, אחד זה סנכרון בין הפרויקטים השונים, בשביל שנוכל להגיע למצב ששני האלגוריתמים יעבדו יחדיו חובה עבודה שבועית ויצירת מערכת ניהול של שני הפרויקטים יחדיו, החלק השני זה עבודה צמודה עם המנחים מכיוון שהפרויקט בחלקו הוא מימוש תיאוריה והתבססות על מחקרים דומים שנעשו.

מילון מונחים , סימנים וקיצורים :

פורמט החוקים יהיה בצורה הבאה :

$$a_1 \wedge \ldots \wedge a_n \rightarrow b_1 \vee \ldots \vee b_m$$

החלק שלפני החץ נקרא body והחלק שאחריו נקרא head .

מודל – השמה שמספקת את סט החוקים.

מודל מינמלי – הוא מודל עבורו מספר ההשמות של ערכי TRUE במשתנים עבור סט של חוקים הוא מינימלי (קיים הסבר מפורט יותר במבוא).

רכיב קשירות- בגרף רכיב קשירות הוא אוסף של קודקודים שמכל קדקוד ניתן להגיע לקדקוד אחר.

סופר גרף- גרף של רכיבי קשירות בהחלט, כל קדקוד יהיה בעצם רכיב קשירות בגרף המקורי.

Source – קדקוד בסופר גרף אשר לא נכנסים אליו קשתות אלה רק יוצאים ממנו קשתות.

Vertex separator (ספארטור) -עבור תת קבוצה של קודקודים $S \subset V$, S זה סאפארטור, עבור קודקודים שאינם סמוכים a ו b -אם הסרת ה- S מהגרף מפרידה בין a ו b לרכיבי קשירות נפרדים.

Source גדול- גודלו צריך לקיים $\frac{|source|}{|V|} > 0.2$

E׳ – סאפארטור של קשתות, רשימת קשתות שאם נסיר אותם נפרק את רכיב הקשירות.

V׳ – סאפארטור של קודקודים.

$$t=\left||S| - \frac{|v|}{2}\right|$$

V- כמות הקדקודים בגרף.

S- בהינתן חתך בין 2 קודקודים בגרף, S מכיל את קודקודים שבצד של הקדקוד שממנו מתחילים את ההזרמה. לכל קדקוד נשמור את גודל S .

K -גודל החתך.

A – גרף עזר שמתקבל כתוצאה מהרצת אלגוריתם [40], באמצעות הגרף נמצא את E׳, ישנה הוכחה שגרף זה הוא עץ.

1. מבוא

בפרויקט זה נתייחס לבעיית ה - SAT או בעברית בעיית הספיקות שהיא בעיית הכרעה שהוכחה כ- NP שלמה (קוק-לוין) משמעות זו היא שלא קיימת לבעיה זו אלגוריתם שפותר אותה בזמן שאינו מעריכי. בבעיית הקביעה אם קיימת פרשנות המספקת נוסחה בוליאנית נתונה. במילים אחרות, הוא שואל אם המשתנים של נוסחה בוליאנית נתונה יכולים להיות מוחלפים בעקביות על ידי ערכי TRUE או FALSE בצורה כזו שהנוסחה מעריכה ל- TRUE. אם זה המקרה, הנוסחה נקראת סיפוק. מצד שני, אם לא קיימת משימה כזו, הפונקציה המבוטאת על ידי הנוסחה היא FALSE עבור כל המטלות המשתנות האפשריות והנוסחה אינה ניתנת לתיאוריה..

אנו נתמקד בבעיות SAT חיוביות, כלומר כל פסוקית שלו מכילה לכל הפחות ליטרל חיובי אחד . למשל (¬xV¬zV¬w) היא פסוקית שאינה יכולה להתקבל במקרה שלנו.

לבעיית מציאת מודלים מינימלים של בעיות לוגיות (SAT) יש היסטוריה ארוכה. אלגוריתמים חדשים מסוימים לפתרון בעיות SAT כגון : GSAT(SLM92),WALKSAT(SKC94), וגרסאות משופרות של Davis Putnam algorithm [DLL62,CA93,LA97], מאפשרים לנו לפתור בעיות SAT גם עבור סט גדול של חוקים.

חישוב מודלים מינימליים הוא נושא מרכזי בבינה מלאכותית (AI), ועומד במרכזם של מערכות רבות כגון logic programming, default reasoning , minimal diagnosis, planning .

במודל זה אנו נתייחס לתיאוריה עבור סט חוקים מהצורה $a_1 \wedge \ldots \wedge a_n \rightarrow b_1 \vee \ldots \vee b_m$ כאשר m>0 .

<u>מודל מינימלי</u>

בהנחה ש m הוא מודל של תיאוריה מסוימת T, נגדיר positive (m) להיות קבוצת המשתנים להם m מציבה True . נאמר ש M היא קבוצת כל המודלים אזי $m \in M$ הוא ***<u>מודל מינימלי</u>*** עבור T אם״ם לא קיים מודל $m' \in M$ כך ש $positive(m') \subset positive(m)$

חישוב מודל מינימלי מתחלק לשתי משימות הראשונה מציאת מודל והשנייה בדיקה האם המודל הוא מינימלי.

מאמר [1] הציע פתרון למציאת מודל מינימלי, הוצע שאם נעביר את המודל לגרף, ניצר סופר גרף SG ונציב ערכים רק עבור החוקים שהמשתנים שלהם נמצאים ב source של SG. למעשה בפתרון זה אנו גורמים לזמן הריצה להיות תלוי בגודל רכיב הקשירות הגדול ביותר . למעשה עבור אלגוריתם [1] פירוק רכיב הקשירות הגדול ביותר הוא דבר הכרחי כי אלמלא זה האלגוריתם [1] לא מקצר לנו זמני ריצה כלל.

אך פירוק רכיב הקשירות לא מספיק, אנו נבקש גם פירוק מאוזן אנו נרצה שעבור V' מינימלי נמצא פירוק הרכיב הגדול למספר רכיבים שקטנים משמעותית ממנו. כלומר עבור רכיב קשירות בגודל 100 האלגוריתם [1] מציע לנסות להציב 2^{100} פעמים שזה מספר הצבות עצום, לכן אם נפרק את רכיב הקשירות בצורה יחסית מאוזנת לדוגמה רכיב בגודל 60 ורכיב שני בגודל 40, רכיב הקשירות הגדול ביותר יהיה כעת בגודל 60 ואז ההצבה תהיה 2^{60} שזה משמעותית קטן יותר מההצבה הקודמת.

עבורנו רמת האיזון של הפירוק חשובה, אך במידה ויינתן לנו פירוק שיפרק לגמרי את רכיב הקשירות, כלומר כל רכיבי הקשירות שיישארו לאחר הפירוק יהיו ממש קטנים זה יהיה לנו טוב אף יותר.

חישוב פירוק מאוזן (balance separator) V' זאת בעיה שהוכחה כ NP שלמה [43], ואולם יש המון מאמרים בנושא רוב המאמרים ככולם מתרכזים בנושא פירוק מאוזן עבור גרף עם קשתות לא מכוונות. בפרויקט זה אנו עובדים רק עם גרף מכוון מהסיבה שאלגוריתם [1] דורש גרף מכוון (ללא גרף מכוון לא נצליח לעקוב אחרי הקשר בין הקדקודים).

חשוב לציין כדאי שהפירוק יהיה מאוזן לגמרי זאת בעיה NP שלמה [43]_וכלל שנבקש פירוק פחות מאוזן כך הבעיה תהיה יותר ליניארית. הפירוק הטוב ביותר שיתאים למאמר [1] זה דווקא פירוק שיפרק את רכיב הקשירות להרבה רכיבים קטנים ולאו דווקא ל-2 רכיבים מאוזנים.

הפירוק נעשה בכמה שלבים, בהינתן רכיב קשירות גדול שאותו צריך לפרק אנו נשתמש באלגוריתם ממאמר [40], בהינתן גרף שכולו רכיב קשירות האלגוריתם מעביר את גרף הקלט לבניית גרף עזר כדי לאחסן מידע אודות קישוריות קצה בין כל זוג קודקודים בזמן O (F(n)), כאשר F הוא המורכבות בזמן כדי למצוא את הזרימה המקסימלית בין שני קודקודים בתרשים G ו- n = |V|.
עבור כל ערך של k, ניתן לקבוע את רכיבי ה - k-edge על ידי חציית הגרף העזר בזמן O (n).
אלגוריתם זה ממפה לנו את מספר הדרכים בין כל קדקוד לקדקוד ברכיב קשירות ובעצם אם ניקח את הקשת עם המשקל המינימלי נמצא את הקשת שאם נוריד אותה נצליח לפרק את הגרף E'.
למרות שזה לא היעד שלנו (מציאת קשתות שיפרקו) ולמרות שהקשתות שמוחזרות לפירוק לא יפרקו בהכרח את הרכיב בצורה מאוזנת, אנו נצליח להפיק מידע רב מהאלגוריתם הנ"ל ובעזרת מספר שיפורים יהיה אפשר למצוא V' קטנה.

2. תיאור הבעיה

הבעיה של מציאת V' היא נושא שמופיע בהרבה מחקרים, אך עבור מציאת V' עבור גרף מכוון אין הרבה מחקרים בנושא, לכן אחת הבעיות זה מציאת חומר רקע שעליו ניתן יהיה להסתמך כמקור .

תיאור הבעיה המרכזי היא בהינתן גרף מכוון בעל רכיב קשירות בגודל הגרף מצא קבוצת קודקודים מפרידים קטנה שתפריד את רכיב בצורה מאוזנת.

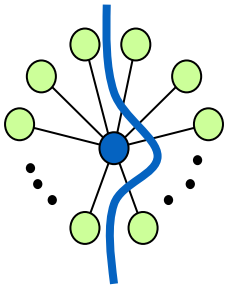
בעיות בבניית מבני הנתונים לגרף:

בניית הגרף צריכה להתבצע בהתאם למבנה הנתונים של החוקים ובהתאם ללוגיקה של סט החוקים. יש למצוא את רכיב הקשירות הגדול ביותר בגרף ולבצע עליו מניפולציות כך שיהיה ניתן לדעת איזה קודקודים (משתנים) חשודים שיגרמו לפירוק רכיב הקשירות.

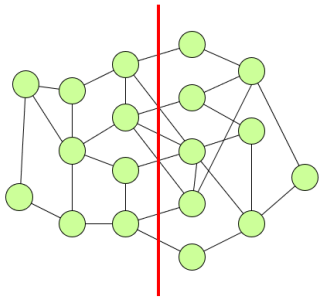
האלגוריתם של החוקים יוצר מחדש כל פעם גרף מסט החוקים שבהם לא נעשו הצבות, אלגוריתם זה הוא רקורסיבי וגוזל משאבים רבים.
מהגרף שנוצר מרכיב הקשירות אנו נמצא את כל רכיבי הקשירות וכך ניצור סופר גרף SG.
האלגוריתם שמחפש רכיבי קשירות מחזיר את ה-source .
אם ה source קטן אז האלגוריתם שמוצא מודל מינימלי יציב את ה- source בחוקים.
אם ה-source גדול - בפרויקט זה אנו נתייחס רק למקרה הזה.
ניצור גרף מרכיב הקשירות הגדול הנ"ל נקרא לו G.

הבעיה בהפרדת רכיב הקשירות

אנו מיישמים אלגוריתם "k-Edge connected component" [40], מטרתו היא מציאת k דרכים מקדקוד אחד לקדקוד שני ברכיב קשירות וכך נמצא את 'E.



1



2

אחד מהיתרונות של מציאת 'E זה שניתן באמצעותה למצוא את 'V כפי שניתן לראות באיור 2. אך ההפך אינו נכון כפי שניתן לראות באיור 1.

אלגוריתם [40] מחזיר גרף (ניתן לראות שקיימת הוכחה שגרף זה הוא בעצם עץ), הגרף הזה מייצג את מספר הדרכים המינימלי בין כל זוג קודקודים ברכיב הקשירות כלומר בגרף G.

הבעיה באלגוריתם [40] שהגרף שהוא מחזיר כל קשת שבו מכילה משקל K, והמשקל בעצם מייצג את מספר הקשתות בחתך של הגרף המקורי – G, המטרה שלנו היא מציאת קודקודים מתוך כל הקשתות בחתך שהם יפרקו את רכיב הקשירות בצורה מאוזנת.

חלק מהדרישות שעבור רכיב קשירות גדול יהיה לנו פירוק מאוזן שלו, לכן לבעיה במציאת 'V קטנה אשר יפרק את רכיב הקשירות נוסיף את הדרישה שרכיב הקשירות יפורק בצורה מאוזנת.

הבעיה בבדיקות

נעשה מספר בדיקות, כולן עבור גרפים, אנו צרכים לדעת עבור גרף שנוצר מסט חוקים מהם כל רכיבי הקשירות שלו .
הבדיקות יתנהלו כדלקמן:

1. אנו נשתמש בתוכנות קיימות שמוצאות 'V. אנו נעשה השוואה עבור גרפים זהים מהו ה 'V. ישנם מספר תוכנות מוכנות כגון METIS, PARTY, CHACO , JOSTLE, SCOTCH, GNU עבור תוכנות אלה נצטרך ללמוד מהם באיזה אופן הגרפים אמורים להיות מיוצגים, אך הבעיה המרכזית בכל התוכנות שמוצעות להלן היא שכולן בנויות עבור גרפים לא מכוונים ואילו הפתרון שאנו מציעים יעבוד על גרף מכוון.
2. אנו נבקש למצוא 'V מינימלי.
3. בעיה נוספת שאנו נתקל בבדיקות זה רמת האיזון של הפירוק. עבור רכיב קשירות יכול להיות מספר רב של פירוקים חלקם מאוזנים יותר וחלקם פחות, לכן ההשוואה של 'V תלויה ברמת איזון הפירוק, לכן נעשה בדיקה עבור כל גרף נתון מהו פירוק מאוזן שלו.

4. בדיקה נוספת זאת בדיקת זמני ריצה – לאלגוריתם [40] זמן הריצה ליניארי, אך נעשה בדיקות מול התוכנות הקיימות אשר מוצאות V', וב- [44] אולי זמן הריצה שאנו מציעים לא יעיל לעומתם.

3. תיאור הפתרון

פרויקט זה עוסק בפתרון בעיית SAT, ולכן יש למצוא עבור סט של חוקים האם קיימת השמה מספקת .עבור מציאת מודל מינימלי יש להשתמש באלגוריתמים לפתרון בעיית SAT זמן הריצה יהיה מעריכי כיון שבעיה זו הוכחה כבעיה השייכת למחלקה NP שלמה(קוק-לוין) .

האלגוריתם [40]:

Algorithm 1: Algorithm ModuMin

Input: A positive theory T
Output: A minimal model for T

```
M := ∅ ;
while T ≠ ∅ do
    if There is a clause δ in T violated by M such that |head(δ)| = 1 then
    let X := head(δ); M := M ∪ X ;
    T := Reduce(T, X, ∅) ;
    else
        let G be the super-dependency graph of T;
        Iteratively delete from G all the empty sources ;
        let S be the set of atoms in a source of G ;
        let T_S be the subset of T containing all the clauses from T having only
          atoms from S;
        let X be a minimal model of T_S;
        M := M ∪ X ;
        T := T − T_S; T := Reduce(T, X, S − X);
return M
```

האלגוריתם הנ"ל ממומש בפרויקט של עדי, והוא מוצע כפתרון למציאת מודל מינימלי.

כעת ניתן להבין שהאלגוריתם שמוצע למציאת מודל מינימלי פועל בזמן ריצה מעריכי בגובה רכיב הקשירות הגדול ביותר. בשביל לשפר את זמן הריצה של האלגוריתם אזי עבור source גדול יהיה צורך לפרק אותו קודם ע"י מציאת כמות מינימלית של קדקודים שיכולה לפרק את הרכיב בצורה מאוזנת.

בשלב ראשון לא התמקדנו בפירוק מאוזן, אבל לקחנו אלגוריתם כללי שמחשב חיתוך מינימלי ונותן לנו לכל זוג של קודקודים איזה קודקודים כדאי להוציא כדי להפריד את הרכיב.

הפתרון שלנו בעזרת אלגוריתם [40].

<u>האלגוריתם:</u>

Algorithm 1: Construction(G(V, E), s, N)
  If $N = \{s\}$
    Return.
  Randomly pick a vertex t from $N - \{s\}$.
  $(x, S, T) :=$ S-T MAX-FLOW(G, s, t).
  $(x', T', S') :=$ S-T MAX-FLOW(G, t, s).
  If $x' < x$
    $x := x', S := S', T := T'$
  Add edge (s, t) with weight x to A
  CONSTRUCTION$(G, s, N \cap S)$
  CONSTRUCTION$(G, t, N \cap T)$

באלגוריתם זה [40] נמצא מידע שבאמצעותו נוכל למצוא ׳V מינימלי ועדיין שהפירוק יהיה מאוזן.

תיאור הפתרון המוצע

למבט כולל על הפתרון ראה נספח ה׳ רשימת מודלים.
בשביל להקל על מציאת הפתרון אנו חילקנו אותו למספר שלבים:

1) <u>יצירת גרף מסט החוקים:</u>
ראה נספח ז׳.

2) <u>שלב ראשון – מציאת רכיבי קשירות:</u>

בשביל למצוא את רכיבי הקשירות בגרף אני מריץ אלגוריתם DFS ואז מסדר את הקדקודים על ידי זמן הסיום בסדר יורד, הופך את הגרף, ומריץ DFS על הגרף ההפוך, מתחילים את ריצת האלגוריתם מהצומת בעל זמן היציאה הגבוה ביותר מבין אלו שחושבו. וכאשר האלגוריתם מסיים את הסריקה שהחלה מהצומת הזה ועליו לבחור צומת חדש, בוחרים את הצומת בעל זמן היציאה הגבוה ביותר מבין אלו שנשארו, וכן הלאה.

3) <u>שלב שני יישום אלגוריתם ממאמר [40] למציאת ׳E:</u>

בפרויקט זה אנו מיישמים אלגוריתם ״k-Edge connected component״ [40] שמטרתו היא מציאת k דרכים מקדקוד אחד לקדקוד שני ברכיב קשירות. אנו נשתמש באלגוריתם זה, שבעצם מוצא קשתות להוריד מרכיב הקשירות, כך שיהיה ניתן לפרק את רכיב הקשירות באופן שווה יחסית, לכך שיחפש קודקודים להוריד ושעדיין יפרק את רכיב הקשירות באופן סימטרי יחסית.
הסבר על אלגוריתם [40] ראה נספח ו׳.

4) <u>שלב שלישי קבלת מידע מגרף A בשביל מציאת 'V :</u>

אנו נרצה להסיר את הקדקודים שקשתותיהם נמצאים בחתך(אנו נבחר קדקוד אחד מכל קשת).
קודקודים אלה נחזיר למבנה הנתונים של החוקים .
במצב זה ועבור הקשתות בעלות המשקל המינימלי בגרף(עץ) A נדע שאם נסיר קשת בעלת משקל מינימלי נצליח לפרק את רכיב הקשירות.
הקשת בעלת ה- k המינימלי (ויכול להיות יותר מקשת אחת) תחזיק את הקודקודים a ו b .
אולם, אנו מחפשים את הקדקודים שאם נסיר אותם מרכיב הקשירות נצליח לפרק את רכיב הקשירות בצורה מאוזנת (במקרה שלנו יכולים להיות K קשתות בחתך ועבורם נצטרך לבחור איזה קדקוד כדאי להסיר). דבר שלא משתלב עם הרצון שלנו למצוא מספר מינימלי של קודקודים , כי השיפור של אלגוריתם [1] דורש הצבה של הקדקודים (המשתנים) בסט החוקים, ואין לנו רצון להציב קודקודים "לחינם", כלומר שהם בסוף לא יפרקו לנו את רכיב הקשירות בצורה מאוזנת ויישאר לנו רכיב קשירות קצת יותר קטן מהקודם. לכן נמצא דרך להסיר קודקודים בעלי סיכוי גבוה יותר לפרק את הגרף(סעיף 5).

5) <u>שלב רביעי מציאת 'V כאשר נתון לנו 'E :</u>

בשלב זה של הפרויקט לא התמקדנו בגרף מאוזן אבל לקחנו אלגוריתם [40] כללי שמחשב חיתוך מינימלי ונותן לנו לכל זוג קודקודים איזה קודקודים כדאי להוציא כדי להפריד את הרכיב.
המעבר לאלגוריתם שמסיר רכיב קשירות יפעל בדרך הבא :

- לאחר יצירת גרף A מגרף רכיב הקשירות המקורי, ניצור גרף שני נקרא לו גרף "מיוחד", הגרף ייווצר באופן הבא :

  - עבור כל קדקוד x ניצור קדקוד x0 .
  - ניצור קשת בין x0 ל a (x0->x). עבור כל הקשתות שנכנסות לx, נשנה אותן שיצביעו ל-x0.

  כעת, כאשר נבקש למצוא קודקודים להוריד, ויש לנו a ו b קודקודים שעבורם הקשת היא בעלת המשקל המינימלי בגרף A, אנו נריץ את האלגוריתם למציאת זרימה מקסימלית וחיתוך מינימלי בגרף "המיוחד", כלומר נמצא את כל הקשתות בחתך בין a ל b ונניח ש x נמצא בחתך, אנו נעדיף להסיר קדקוד x אם החתך מחזיק את x ו x0 .
  בכל מקרה אנו נמצא קדקוד להסיר מקשת ממשתנה y0 למשתנה x (כאשר x יועדף להיות y).

- שיפור נוסף שנוסיף ע"מ להעדיף קדקוד להסיר זה להסיר קשת שעבור גודל S (משתנה שמחזיק את גודל S מריצת אלגוריתם הזרימה המקסימלית, S מכיל את כמות המשתנים שבצד של החתך, הקדקוד שממנו מתחילים את ההזרמה), K (גודל החתך), v כל הקדקודים בגרף .

  אם נמצא קדקוד שעבורו יש מינימום $K + \left||S| - \frac{|v|}{2}\right|$ בהרצה על הגרף "המיוחד", זה יבטיח לנו הסרה של קדקוד שמחובר בקשתות להרבה קודקודים, לכן אם נסיר את הקדקוד הנ"ל נפרק את רכיב הקשירות בצורה מאוזנת יחסית.

תיאור הכלים המשמשים לפתרון

אנו משתמשים באלגוריתם למציאת כל רכיבי הקשירות בגרף, אנו יוצרים בעצם סופר גרף אשר מטרתו היא לספק source (רכיב קשירות בגרף) למחלקת החוקים.

אם ה source גדול במיוחד נרצה לפרק אותו אם הוא קטן אין טעם להפעיל אלגוריתם לפירוק (זה אלגוריתם יקר).

בפרויקט זה אנו מיישמים אלגוריתם "k-Edge connected component" שמטרתו היא מציאת k דרכים מקדקוד אחד לקדקוד שני ברכיב קשירות. אלגוריתם זה זרימה מקסימלית באמצעות אלגוריתם למציאת זרימה- Ford Fulkerson method Edmonds Karp algorithm for finding max.

תכנית בדיקות

לפרויקט שלנו קיימת דרישה לייעול מקסימלי וקיצור זמני ריצה של אלגוריתמים קיימים.
כדי להשיג מטרה זו אנחנו מבצעים בדיקות יחידה על שלב ביצירה חדשה של גרף ויצירה של רכיב קשירות.
כדי להגדיר את רמת האמינות של התוכנה, חילקנו את הבדיקות ל-2 חלקים. נקיים בדיקות עבור התוכנה המשותפת של שני הפרויקטים ונעשה בדיקות רק עבור פירוק הגרף.
חלק מהבדיקות שלנו אקטיביות וחלקם פאסיביות כלומר בחלק מהבדיקות נעשה השוואה של זמני ריצה וסוגי 'V בין כמה תוכנות שונות שירוצו מול התוכנה שלנו וחלק מהבדיקות ניצור גרפים שאנו יודעים את ה 'V שלהם ונבדוק את התוצאה של האלגוריתם שלנו.
בנוסף נעשה בדיקות אטומיות של כל מחלקה על מנת למצוא מחלקות שמבזבזות משאבים.
בנספח ח' נציג חלק מהבדיקות שנעשו עד כה.

כאמור הבדיקות שלנו בפרויקט זה מתחלקות ל-2:

1. בדיקות עבור קלטים שהם פסוקיות מהצורה CNF .
2. בדיקות עבור קלטים שהם גרפים.

<u>בדיקות עבור פסוקיות מהצורה CNF:</u>

כאמור פרויקט זה הוא חלק מפרויקט משותף, לכן חלק גדול בהצלחת הפרויקט של עדי יוגדר מהצלחת פרויקט זה, לכן עבור כל קובץ בדיקות שמכיל סט גדול של פסוקיות אנו נרצה, בנוסף למציאת מודל מינימלי, למוצא את רכיב הקשירות הגדול ביותר, ולבדוק עליו מה קורה שמנסים לפרק אותו.

יכולים להיות כמה תוצאות אפשריות:

- ברגע שהצלחנו לפרק את רכיב הקשירות הגדול ביותר באופן מאוזן יחסית אז נצליח לצמצם באופן משמעותי את זמן הריצה המעריכי של האלגוריתם, לזמן ריצה מעריכי אבל קטן משמעותית.
- ברגע שהצלחנו לפרק את רכיב הקשירות אך הפירוק לא היה מאוזן ייתכן שנשאיר רכיב קשירות גדול עדיין, גם אותו ננסה לפרק, וגם הפירוק שלאחריו לא יהיה מאוזן, ככה ברקורסיה (השיטה שמציבה משתנים נעשית ברקורסיה) . מצב זה יגרום לנו לבזבז הרבה עבודה על פירוק בלבד, דבר שיגרום לנו לאבד מזמן הריצה שנרצה לשפר.

בדיקות על הזיכרון ומדידת זמני ריצה:

1. בדיקת הרצת Davis Putnam לבד
2. אלגוריתם [1] + Davis Putnam
3. בדיקת אלגוריתם [1] + אלגוריתם שמפרק את רכיב הקשירות הגדול ביותר.

אנו נרצה לדעת את ההשפעה האמיתית בשימוש הזיכרון ומה ההבדל בין כל השלושה מבחינת זמני ריצה.

חשוב לציין, אנו נשתמש באלגוריתם שלנו ב WASP שבאמצעותו למצוא מודל מינימלי לסט קטן של קלטים, לכן הבדיקות שלנו יהיה מול האלגוריתם WASP שירוץ על סט גדול של חוקים, כך נדע אם באמת הצלחנו למצוא מודל מינימלי בזמן משופר אך אלגוריתם זה לא נותן מענה לבדיקה עבור פירוק רכיב קשירות בגרף.

פה באמת נבחן, אם יש שיפור משמעותי מבחינת זמני הריצה אז התאוריה אלגוריתם [1] + הייעול שלו שהוא פירוק רכיב הקשירות הגדול ביותר יעמוד במבחן התוצאה .

<u>בדיקות עבור קלטים שהם גרפים:</u>

ישנם מספר תוכנות מוכנות למציאת V' כגון METIS, PARTY, CHACO , JOSTLE, SCOTCH, GNU .
אולם תוכנות אלה מתאימות רק לגרפים לא מכוונים, אנו נשתמש בתוכנות עבור בדיקה לאלגוריתם שלנו, אנו נריץ סוגי גרפים שונים את אותם קלטים ונראה אילו תוצאות כל תוכנה נותנת .

סוגי הבדיקות שנעשה:

1. בדיקה על רכיב קשירות בגודל הגרף.
2. בדיקות מול תוכנות קיימות אילו V' נקבל ומה רמת האיזון של הגרף כאשר נוציא את V'. אם ישנה תוכנה שנותנת איזון ברמה טובה נשתמש בה.
3. בדיקה על גרפים בגדלים קטנים אולם שעדיין נוכל לדעת מהו ה V' .
4. בדיקות עבור רכיב קשירות מעגלי, במקרה כזה אלגוריתם [40] יחזיר לנו גרף שמשקל כל קשת מכל קדקוד למשנהו הוא 1, לכן אם ננסה לפרק את רכיב הקשירות נקבל המון רכיבי קשירות שגודלם 1.
5. בדיקות בשביל למצוא V' מינימלי.
6. מדידת זמני ריצה של מציאת V' .

4. סקירת עבודות דומות בספרות והשוואה

קיימים מספר עבודות שיש להם רמת דמיון לפרויקט שלנו ביניהם

(ראו ביבליוגרפיה ערכים 43,44,8,7,41,24,14,9,3,25,4,1,22,18,34)

Finding strong bridges and strong articulation points in linear time:

מאמר [44] מציע פתרון בזמן ריצה ליניארי, בתחילת העבודה על הפרויקט הוצע מאמר זה כפתרון לפירוק רכיב הקשירות הגדול ביותר, למרות שזמן ריצה לינארי ואף מהיר יותר משלנו, תהליך העבודה של [44] הוא שתחילה נהפוך את רכיב הקשירות לעץ שבו כל קדקוד מדורג כרמת החיבור שלו לקדקודים אחרים כלומר בעל השפעה (dominators), ואז בריצה על הקדקודים נדע מכל קדקוד לכל קדקוד מיהו הכי בעל השפעה והכי בעל השפעה זה הקדקוד שנסיר.

הפתרון של מאמר זה ימצא קדקוד בודד אשר אם נסיר אותו מרכיב הקשירות בגרף נצליח לפרק את רכיב הקשירות לגמרי. מקרה זה הוא מקרה פרטני לבעיה שאנו מתמודדים איתה. במקרה שלנו הסיכוי לקבל רכיב קשירות שקדקוד אחד יפרק אותו נמוך ביותר ולכן אלגוריתם זה לא מתאים לפרויקט זה.

NP-completeness of the Planar Separator Problems:

במאמר זה [43] קיימת הוכחה שבעיית פירוק רכיב קשירות באמצעות קודקודים או באמצעות קשתות בצורה מאוזנת לגמרי, כלומר לחצי, היא NP שלמה, ואילו ככל שאנו מוכנים להתפשר על האיזון אנו נצליח לפתור אותה בזמן ליניארי.

יש למאמר [43] קשר ישיר לבעיה שאנו מתמודדים איתה בפרויקט. אנו בפרויקט זה מצד אחד דורשים איזון בפירוק אך אנו מוכנים להתפשר על האיזון לטובת זמן ריצה, כלומר אין לנו בעיה שרכיב הקשירות יישאר אפילו ב 90% מהגודל המקורי שלו אך לא נפסיד זמן ריצה על הפירוק. בסופו של יום מטרת הפרויקט זה שיפור זמני ריצה של [1] לכן כל שיפור בזמן ריצה יהיה הצלחה.

Finding small balanced separators:

דוגמה לעבודה דומה למה שאנו עושים בפרויקט זה במאמר [41], שם מציעים אלגוריתם למציאת חתך בגודל k שמחלק את הגרף לרכיבים קטנים יותר מ αn, $\frac{2}{3} \leq \alpha < 1$. אצלנו רכיבים אלו עדיין יהיו גדולים מידי שכן אנו מבקשים לחלק את רכיב הקשירות הגדול ביותר בצורה מאוזנת יותר, ואפילו לנסות לפרק אותו לכמה שיותר רכיבי קשירות ממש קטנים. במאמר זה גם מוכח שאם נוריד קודקודים אקראיים מהגרף נצליח לפרק אותו, כאשר יש לנו טווח אפשרות זיהוי בגודל $O(k^3\epsilon^{-1}\log(1/\epsilon))$, טווח זה שופר ל $O(k\epsilon^{-1})$. הוכחה זאת יכולה להיות לנו יעילה כאשר ננסה להוריד את זמן הריצה הגדול יחסית שאנו עושים במימוש מאמר [40] אך החיסרון העיקרי (עבורנו) במאמר זה שבתוצאת פירוק רכיב הקשירות לא יינתן לנו רכיב מפורק בצורה מאוזנת.

Davis Putnam:

דוגמה לעבודה דומה למה שאנו עושים בפרויקט זה הוא האלגוריתם של Davis Putnam שזה אלגוריתם שמוצא מודל מינימלי בכך שהוא מתעדף בהצבה של במשתנים ערכי false, אך הוא

מתאים לבעיות SAT ששונות מצורת סט החוקים שאנו נשתמש בהם כיוון שאצלנו תמיד יש מודל לכל סט חוקים רק צריך למצוא מודל מינימלי.

מה שלנו יש להציע הוא זמן ריצה יותר מהיר מזמן הריצה של אלגוריתם זה אך במקרים מסוימים במהלך הפרויקט כנראה שנשתמש באלגוריתם זה כדי לראות אם קיים מודל עבור השמה מסוימת .

WASP :

דוגמה נוספת לעבודה דומה היא התוכנה WASP אשר חלק ממטרותיה היא מציאת מודל מינימלי והיא תוכנה קיימת שנמצאת ב- GitHub, היא מיישמת טכניקות שיוצגו במקור עבור פתרון SAT בשילוב עם שיטות אופטימיזציה.

יהיו לנו כמה שימושים תוכנה זאת :

1. לאחר שניצור גרף ונחזיר source (הכוונה לרכיב קשירות בסופר גרף) נרצה להציב ערכים במשתנים שהוחזרו ב source . ובכך אולי נצליח לייעל את האלגוריתם WASP כי הוא לא ירוץ על כל סט החוקים וכל המשתנים אלה ירוץ רק על המשתנים שנמצאים ב source .
2. נרצה לעשות בדיקות של האלגוריתם שלנו שמוצא מודל מינימלי באמצעות גרף ובזמן ריצה של רכיב הקשירות בגדול ביותר לעומת זמן ריצה של אלגוריתם WASP שיפעל ללא כל השיפורים שלנו. כך נדע בעצם אם הצלחנו לשפר את זמני הריצה באמת של האלגוריתם, ואולי נדע על אילו מקרים כן הצלחנו לשפר ואם לא על כל המקרים מדוע לא.

5. סיכום \ מסקנות

מסקנות עבור הסנכרון של שני הפרויקטים :

בשלב האלפא של הפרויקט המשולב (הפרויקט הזה יחד עם הפרויקט של עדי) הצלחנו לקבל סט של חוקים בצורה CNF להכניס אותם למבנה נתונים וליצור גרף מסט הנתונים הנ"ל.

לאחר יצירת הגרף הראשוני אנו נמצא את כל רכיבי הקשירות בגרף, אלגוריתם למציאת רכיבי הקשירות יחזיר לנו את ה source של רכיבי הקשירות כלומר ה source בסופר-גרף .

אנו רצים על כל source בסופר גרף ובודקים אם הוא source קטן אנו מציבים את המשתנים שמיוצגים ב source בעזרת אלגוריתם Davis Putnam בסט החוקים המקורי.

אחרת אם זה source גדול אנו משתמשים בסאפארטור ע"מ לנסות לפרק את רכיב הקשירות (כלומר source).

אנו כבר למדים שהרצת אלגוריתם [1] שמשתמש ב Davis Putnam אך משתמש בו בצורה יותר יעילה כלומר רק עבור source יותר יעילה מלהריץ Davis Putnam על כל המשתמשים.

אולם, הייעול הזה לא מספיק, כפי שראינו בדוגמת ההרצה למעלה ייתכן מצב שבו רכיב הקשירות הגדול ביותר הוא כמעט בגודל מספר המשתנים, כלומר ריצה על כל source בנפרד לא תייעל יותר מידי אלה תהפוך להרצת Davis Putnam רגיל. נוסיף לכך את הדרישה למצוא 'V מינימלי בגלל שאנו רוצים "לחסוך" בהצבות של משתנים.

המסקנה שלנו שפירוק רכיב הקשירות הגדול ביותר בצורה מאוזנת, באמצעות מציאת 'V מינימלי זה דבר הכרחי שאלגוריתם [1] באמת ייעל זמני ריצה.

<u>מסקנות עבור השיפורים שנעשו מציאת סאפארטור :</u>

עבור מציאת סאפארטור שמשתמש באלגוריתם [40], אלגוריתם [40] משמש ככלי עזר למציאת קודקודים . לכן אלגוריתם [40] הוא אבן הפינה של פרויקט זה.

תהליך פירוק רכיב הקשירות מורכב מכמה אבני יסוד :

1. הסתכלות על כל רכיב כגרף משל עצמו ועבודה רק כל הגרף.
2. הרצת אלגוריתם [40] ויצירת גרף A שישמש לנו ככלי למידע עבור מציאת 'V.
3. החתך בגרף וגודל המינימלי של t (ראה מילון מונחים), באמצעות זה נחליט אילו קדקודים כדאי לנו לבחור כדי להוריד מהגרף.

כל אבני היסוד הנ"ל הם הכרחיים למציאת 'V .

המשך הפרויקט :

- פירוק רכיב הקשירות בצורה מאוזנת יותר – במצב כרגע התמקדנו במציאת 'V קטן, בהמשך נרצה למצוא 'V שיפרק את רכיב הקשירות בצורה מאוזנת .
- שינוי במבנה הנתונים בשביל לשפר זמני ריצה –באלגוריתם שלנו ישנם כמה מבני נתונים שיכולים להשתמש במשאבים רבים, במידה ונצליח למצוא פירוק קטן ומאוזן נשנה את מבני הנתונים הנ"ל בשביל לצמצם בשימוש המשאבים שלנו.

6. נספחים

א. רשימת ספרות \ ביבליוגרפיה

1. F. Angiulli, R. Ben-Eliyahu-Zohary, F. Fassetti, and L. Palopoli. On the tractability of minimal model computation for some cnf theories. Arti_cial Intelligence, 2014. doi: http://dx.doi.org/10.1016/j.artint.2014.02.003 .

2. R. Ben-Eliyahu. A hierarchy of tractable subsets for computing stable models. J. Artif. Intell. Res. (JAIR), 5:27-52, 1996.

3. R. Ben-Eliyahu and R. Dechter. On computing minimal models. Annals of Mathematics and Arti_cial Intelligence, 18:3-27, 1996.

4. R. Ben-Eliyahu-Zohary. An incremental algorithm for generating all minimal models.

Arti_cial Intelligence, 169(1):1-22, 2005.

5. R. Ben-Eliyahu-Zohary and L. Palopoli. Reasoning with minimal models: E_cient algorithms and applications. Arti_cial Intelligence, 96(2):421-449, 1997.

6. N. Bidoit and C. Froidevaux. Minimalism subsumes default logic and circumscription in strati_ed logic programming. In Proceedings of the IEEE symposium on logic in computer science, pages 89-97, June 1987.

7. M. Cadoli. The complexity of model checking for circumscriptive formulae. Inf. Process. Lett., 44(3):113-118, 1992.

8. M. Cadoli. On the complexity of model _nding for nonmonotonic propositional logics. In Proceedings of the 4th Italian conference on theoretical computer science, pages 125-139. World Scienti_c Publishing Co., October 1992.

9. Z. Chen and S. Toda. The complexity of selecting maximal solutions. In Proc. 8th IEEE Int. Conf. on Structures in Complexity Theory, pages 313-325, 1993.

10. M. Davis, G. Logemann, and D. Loveland. A machine program for theoremproving. Communications of the ACM, 5(7):394-397, 1962.

11. J. de Kleer, A. K. Mackworth, and R. Reiter. Characterizing diagnoses and systems. Arti_cial Intelligence, 56(2-3):197-222, 1992.

12. R. Dechter. Constraint processing. Morgan Kaufmann, 2003.

13. C. Drescher, M. Gebser, T. Grote, B. Kaufmann, A. Konig, M. Ostrowski, and T. Schaub. Conict-driven disjunctive answer set solving. KR, 8:422-432, 2008.

14. T. Eiter and G. Gottlob. Propositional circumscription and extended closed-world reasoning are iip2-complete. Theor. Comput. Sci., 114(2):231-245, 1993.

15. M. Gebser, B. Kaufmann, and T. Schaub. Advanced conict-driven disjunctive answer set solving. In IJCAI, 2013.

16. M. Gebser, J. Lee, and Y. Lierler. Elementary sets for logic programs. In Proceedings of the 21st National Conference on Arti_cial Intelligence (AAAI), 2006.

17. M. Gelfond and V. Lifschitz. Classical negation in logic programs and disjunctive

databases. New Generation Computing, 9:365-385, 1991.

18. E. Giunchiglia and M. Maratea. Sat-based planning with minimal-#actions plans and "soft" goals. In AI*IA, pages 422-433, 2007.

19. T. Janhunen, E. Oikarinen, H. Tompits, and S. Woltran. Modularity aspects of disjunctive stable models. Journal of Arti_cial Intelligence Research, pages 813-857, 2009.

20. M. Kalech and G. A. Kaminka. On the design of coordination diagnosis algorithms for teams of situated agents. Arti_cial Intelligence, 171(8):491-513, 2007.

21. H. A. Kautz, D. Mcallester, and B. Selman. Encoding Plans in Propositional Logic. In Proceedings of the Fifth International Conference on the Principle of Knowledge Representation and Reasoning (KR'96), pages 374-384, 1996.

22. L. M. Kirousis and P. G. Kolaitis. The complexity of minimal satis_ability problems. Inf. Comput., 187(1):20-39, 2003.

23. C. Koch, N. Leone, and G. Pfeifer. Enhancing disjunctive logic programming systems by sat checkers. Arti_cial Intelligence, 151(1):177-212, 2003.

24. P. G. Kolaitis and C. H. Papadimitriou. Some computational aspects of circumscription.

J. ACM, 37(1):1{14, 1990.

25. N. Leone, P. Rullo, and F. Scarcello. Disjunctive stable models: Unfounded sets, _xpoint semantics, and computation. Inf. Comput., 135(2):69{112, 1997.

26. V. Lifschitz. Computing circumscription. In IJCAI-85: Proceedings of the international joint conference on AI, pages 121-127, 1985.

27. V. Lifschitz and H. Turner. Splitting a logic program. In ICLP, volume 94, pages 23-37, 1994.

28. J. McCarthy. Circumscription - a form of non-monotonic reasoning. Arti_cial Intelligence, 13:27{39, 1980.

29. J. McCarthy. Application of circumscription to formalizing common-sense knowledge.

Arti_cial Intelligence, 28:89{116, 1986.

30. R. Reiter. A logic for default reasoning. Arti_cial Intelligence, 13(1{2):81-132, 1980.

31. P. Simons, I. Niemela, and T. Soininen. Extending and implementing the stable model semantics. Arti_cial Intelligence, 138(1):181-234, 2002.

32. R. T. Stern, M. Kalech, A. Feldman, and G. M. Provan. Exploring the duality in conict-directed model-based diagnosis. In AAAI, 2012.

33. *Strong Bridges and Strong Articulation Points of Directed Graphs*, Giuseppe F. Italiano Univ. of Rome "Tor Vergata" . Based on joint work with Donatella Firmani, ,Luigi Laura, Alessio Orlandi and Federico Santaroni.

34. Davis Putnam ,http://www.jstor.org/stable/1970289?seq=1#page_scan_tab_contents

35. M. Alviano, C. Dodaro, N. Leone, and Francesco Ricca: **Advances in WASP.** Proceedings of LPNMR (2015). Download Reference

36. M. Alviano, C. Dodaro, J. Marques-Silva, and F. Ricca: **Optimal Stable Model Search: Algorithms and Implementation.** Journal of Logic and Computation (In Press 2015). Download Reference

37. M. Alviano, C. Dodaro, and Francesco Ricca: **Anytime Computation of Cautious Consequences in Answer Set Programming.** Theory and Practice of Logic Programming (2014). Download Reference

38. M. Alviano, C. Dodaro, W. Faber, N. Leone, and Francesco Ricca: **WASP: A Native ASP Solver Based on Constraint Learning.** Proceedings of LPNMR (2013). Download Reference

39. M. Alviano, C. Dodaro, and Francesco Ricca: **Comparing Alternative Solutions for Unfounded Set Propagation in ASP.** Proceedings of AI*IA (2013)

40. *A Simple Algorithm for Finding All k-Edge-Connected Components* :Tianhao Wang,2 Yong Zhang,1,3,* Francis Y. L. Chin,4,5 Hing-Fung Ting,4 Yung H. Tsin,6 and Sheung-Hung Poon7

41. *Finding small balanced separators* : Uriel Feige,Mohammad Mahdian

42.*Graph Theory and Sparse Matrix Computation:* Alan George,John R. Gilbert,Joseph W.H. Liu.

43. NP-completeness of the Planar Separator Problems Junichiro Fukuyama Department of Mathematics and Computer Science Indiana State University.

44. Finding strong bridges and strong articulation points in linear time: Giuseppe F.Italiano Luigi Laura Federico Santaroni.

ב. תכנון הפרויקט

26.7	פגישת היכרות עם רחל ויהודה– הסבר על תיאורית הפרויקט.
15.8	פגישה עם רחל ויהודה – בחירת פרויקט
פגישות כל שבוע עד תאריך סיום הפיתוח	פגישות קבועות בימי רביעי עם צוות הפיתוח לצורך קידום הפרויקט, פיתוח וחלוקת עבודות.
18.9	פגישה שלישית עם רחל ויהודה – חלוקה לצוותים צוות פרויקט על הגרף וצוות פרויקט על החוקים.
19.10	**שלב התנעה** - הגשת טופס התנעה.
22.10	תחילת כתיבת קוד.
15.11	פגישה עם רחל ויהודה עבודה על הצעת הפרויקט.
26.11	**שלב ההצעה** - מסירת נוסח ההצעה, תכנון הפרויקט וניתוח דרישות.
1.2	**שלב האב טיפוס** - מימוש,מחקר, סקירת ספרות, ארכיטקטורה ובדיקות-אלפא.
20.4	**שלב הבנייה** - הגשת מסמך תיכון ומימושו.
[תאריך שיקבע לאחר סיום הפיתוח]	**שלב הבדיקות**- בדיקות של הקוד לבדוק אם הוא משפר זמני ריצה של סטים של חוקים קיימים, השלמת תיעוד/דו"ח טכני.
22.6	**שלב המסירה** – מסירת הפרויקט.

ג. טבלת סיכונים

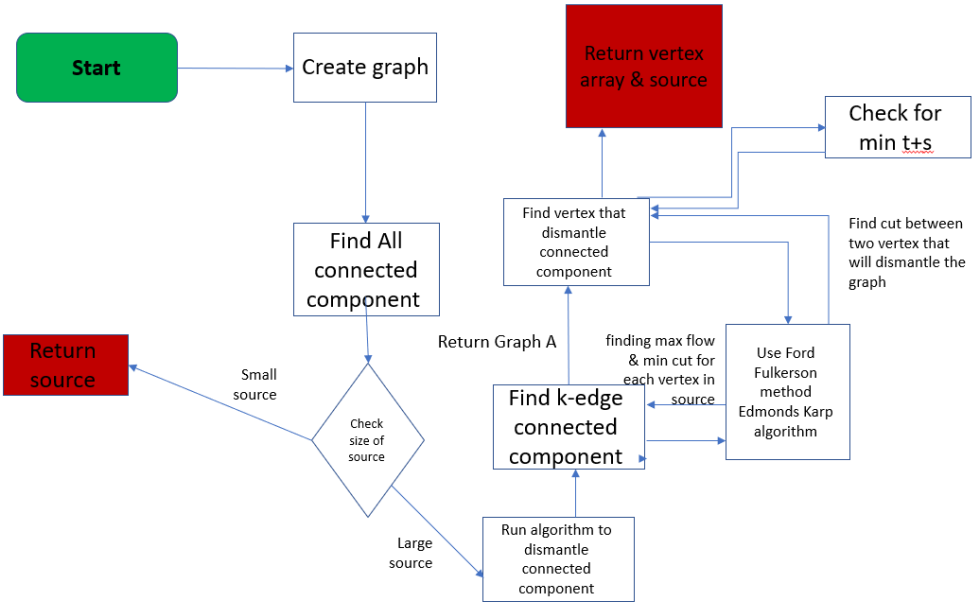
#	הסיכון	חומרה	מענה אפשרי
1	עיכוב של תהליכי פיתוח בבניית מבני הנתונים.	4	שימוש במבני נתונים קיימים אשר נמצאים באינטרנט.

2	הפירוק של רכיב הקשירות לא יהיה מאוזן	2	כל פירוק זה שיפור זמן ריצה וככל שהפירוק פחות מאוזן יותר קל למצוא פירוק לכן יקטין את זמן הריצה, ראה [43]
3	הסנכרון של מחלקת הגרף ומחלקת מבנה הנתונים של החוקים לא יעבוד	9	העבודה על בניית המחלקות תעשה בתיאום מוחלט.
4	עבור בדיקות של כמות גדולה של משתנים לא נמצא מודל מינימלי	7	ניצור גרף של סט חוקים זה וננסה לנתח את רכיב הקשירות ומלה לא נמצא מודל כאשר מפרקים את רכיב הקשירות .
5	בחירת סביבת העבודה ושפת התכנות לא מתאימה להרצת סט גדול של חוקים	3	מספיק שהתאוריה תעבוד אז יהיה ניתן להעביר את הקוד למגוון של שפות תכנות יותר יעלות.
6	הרצת האלגוריתם שמפרק את הגרף יפעל רק עבור רכיבי קשירות מסוימים (כמו רכיב קשירות שהוא עמו מעגל)	5	נריץ מספר גדול של בדיקות ע"מ לתת פתרון עבור כמות גדולה של רכיבי קשירות.
7	הרצת האלגוריתם שלנו על סביבת Linux לא תעבוד כמצופה.	2	לא נוכל להשתמש wasp אך כן נוכל להשתמש באלגוריתמים אחרים שכן הורצו על windows.
8	מטריצת המשקלים שנוצרת מאלגוריתם הזרימה תעמיס מידי על הזיכרון	6	נמצא מבנה נתונים יותר יעיל למשקלי הקשתות או שניצור את המטריצה רק פעם אחת ונעבוד איתה.
9	האלגוריתם הרקורסיבי שפועל למציאת k-edge connected component יגדיל במידה גדולה את זמן הריצה שאנו מנסים להוריד	3	מספיק שהתאוריה תעבוד אז יהיה ניתן למצוא אלגוריתם יותר יעיל לפתרון.
10	הקדקודים שאם נוציא, אמורים לפרק את רכיב הקשירות הגדול ביותר, לא יפרקו אותו בצורה מאוזנת .	2	נמצא דרך תעזור לנו למצוא קודקודים שבעלי סבירות גדולה יפרקו את רכיב הקשירות בצורה מאוזנת. עדיין נשפר את זמן הריצה לאחר הפירוק.
11	התוכנות שמוצאות 'V לא יעזרו לנו לבדיקות כי הן על גרף לא מכוון.	3	נשתמש בבדיקות רק על גרפים קטנים.
12	לא נמצא 'V מינימלי	2	לא לכל גרף יש פירוק .

ד. רשימת\טבלת דרישות

מס' דרישה	סוג	תיאור
1	פלטפורמת מימוש	התוכנה תתבצע בשפת Java בסביבת eclipse.
2	אופן ביצוע	מחלקת החוקים תהיה בסנכרון מלא עם מחלקת הגרף.
3	חלוקת התוכנה	התוכנה תתחלק לשתי כיוונים שונים של עבודה אחד של חוקי התאוריה והשני של גרף המייצג את התאוריה.
4	פונקציונאלי	מבנה הנתונים של הגרף יהיה דינאמי.
5	אופן ביצוע	מבנה הנתונים של הגרף יריץ את האלגוריתם למציאת רכיב הקשירות DFS,וימצא את הגדול ביותר.
6	אפיון טכני	מבנה הנתונים של הגרף יבנה בהתאם למבנה הנתונים של החוקים לפי הפורמט של body→head.
7	אופן ביצוע	המערכת תתחזק מבנה נתונים השומר עבור כל משתנה באיזה חוק הוא נמצא.
8	פונקציונאלי	מבנה הנתונים של הגרף יפרק את רכיב הקשירות הגדול ביותר .
9	בדיקות	נריץ בדיקות על מודלים קיימים ע"מ שנראה שקיים שיפור בזמני הריצה.
10	פונקציונאלי	מחלקת הנתונים של הגרף תחזיר עד שלושה קודקודים בשביל שמערכת החוקים תציב במשתנים הללו.
11	בדיקות	הרצת האלגוריתם שלנו מול אלגוריתם WASP.
12	פונקציונאלי	שילוב אלגוריתם WASP באלגוריתם שלנו.
13	אופן ביצוע	שימוש באלגוריתם k-edge connected component לפירוק רכיב הקשירות והחזרת קודקודים שיפרקו את הגרף.
14	פונקציונלי	לעשות מעקב עבור כל קדקוד מה גודל S (מספר המשתנים לפני החתך המינימלי).
15	בדיקות	הרצה מרובה של בדיקות עבור מקרים שונים של גרפים, הרצה על גדלים שונים של רכיבי קשירות.
16	בדיקות	השוואת 'V מול תוכנות קיימות למציאת 'V.

ה. תרשים מודלים



ו. אלגוריתם למציאת E' – "k-edge connected component"

אלגוריתם זה אשר פועל בצורה רקורסיבית מוצא זרימה מקסימלית באמצעות אלגוריתם למציאת זרימה- Ford fulkerson method Edmonds Karp algorithm for finding max flow , כדי לקבוע את מקסימום זרימה מ s ל t. אם G הוא גרף מכוון, הוא יפעיל גם את האלגוריתם לזרימה מקסימלית לקבוע את הזרימה המקסימלית מ t ל s כי שני ערכי זרימה מקסימלית יכולים להיות שונים מכל כיוון.

לאחר שלב זה, אם G מכוונת, נקבל שתי min_cut (חיתוכים מינימליים), (S, T) ו- (S′, T′).

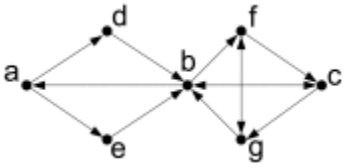
מכיוון שמת החיבור של s ו- t, בגרף מכוון היא המינימום של max-flow (זרימה מקסימלית) בין s ל t ובין t ל- s, הזרימה המינימלית משני אלה, תהיה לדוגמה x, נקצה קשת (s, t) כמו שידמה לנו קישוריות בין s ו t. אנחנו גם נגדיר (S, T) כדי המקביל min-cut.

לאחר מכן, נקצה קשת בין (s, t) עם משקל x נוסף על גרף עזר A. ההליך קורא אז רקורסיבית, תחילה עם S כמו קבוצה של קודקודים שבהם s משמש כמקור, ולאחר מכן עם T כמו קבוצה של קודקודים שבהם t משמש כמקור.

הרקורסיות מסתיימות כאשר S או T מופחת לקדקוד אחד.

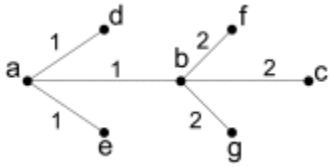
אנו יוצרים עץ A שבעצם מייצג את כמות הדרכים שניתן להגיע מכל קדקוד לשני ברכיב הקשירות.

ציור זה מייצג רכיב קשירות:

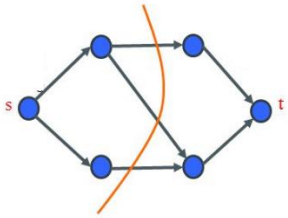


ציור זה מייצג את העץ שנבנה A. לאחר הפעלת אלגוריתם [40]. בציור זה ניתן לראות את מספר הדרכים מכל קדקוד לכל קדקוד, אשר תמיד נבחר את המספר המינימלי.

לדוגמא מקדקוד a לקדקוד c ומקדקוד c לקדקוד a יש רק דרך אחת - זה המינימום בין 1 ל 2.



חשוב לציין שהקשתות בגרף הם חד כיווניות ובעץ הם דו כיווניות.
נסתכל על העץ, אנו מתעניינים בקשתות המינימליות, נבחר זוג קודקודים שעבורם הקשת היא מינימלית k.
נעבור לגרף, עבור קודקודים אלה (שמצאנו מקודם) נריץ אלגוריתם למציאת זרימה מקסימלית.
אמורים להיות k קשתות בחתך של האלגוריתם למציאת זרימה מקסימלית.
אנו נרצה להסיר את הקדקודים שקשתותיהם נמצאים בחתך (אנו נבחר קדקוד אחד מכל קשת).
קודקודים אלה נחזיר למבנה הנתונים של החוקים .



באיור נראה דוגמה לחתך בגרף בין (s,t).

במצב זה ועבור הקשתות בעלות המשקל המינימלי בגרף(עץ) A נדע שאם נסיר את כל K הקשתות שנמצאות בין שני הקדקודים, נצליח לפרק את רכיב הקשירות.

ניתן לראות זאת באיור למעלה של A, עבור קשת (a,b) שהיא בעלת המשקל המינימלי, אם נסיר את b מהגרף המקורי, נצליח לפרק את הגרף המקורי לכמה רכיבי קשירות.

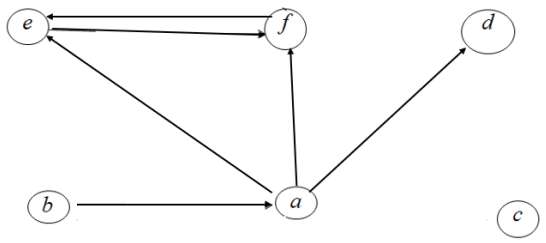
ז. יצירת גרף מסט חוקים נתון

סט החוקים מהצורה $a_1 \wedge \ldots \wedge a_n \rightarrow b_1 \vee \ldots \vee b_m$ (m>0) .
אנו ניצור קשת בין כל מה שב body לכל מה שב head (ראה מילון מונחים לפירוט).
נראה דוגמה ליצירת גרף מסט החוקים ומציאת רכיב קשירות.

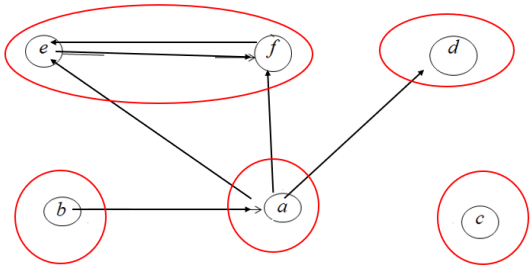
עבור סט חוקים :

1. $a \vee b$
2. $b \rightarrow a$
3. $a \vee c$
4. $a \rightarrow d \vee e \vee f$
5. $e \rightarrow f$
6. $f \rightarrow e$

ניצור את הגרף:



נמצא רכיבי קשירות בגרף:



רכיב הקשירות הגדול ביותר זה {f,e} וכאשר ה source יכול להיות או b או c .

ח. דוגמאות הרצה

דוגמאות עם שימוש בספארטור :

ניקח סט חוקים מדוגמאות שבהם עשו את הבדיקות לתוכנת WASP.

לדוגמה סט חוקים של 830 חוקים ובעלי 200 משתנים.

לשם הנוחות נראה את גודל רכיבי הקשירות בכל הרצה.

```
-----------------------------------------------------------------------------
---------------------------------------
Here are the size of all the connected component in the graph:
sizeof :1

sizeof :1

sizeof :196

sizeof :1

sizeof :1

-----------------------------------------------------------------------------
---------------------------------------
print the source size 1
-----------------------------------------------------------------------------
```

בדוגמה הנ״ל נראה את הסיום של בניית הגרף וחלוקה לרכיבי קשירות שבהם רכיב הקשירות הראשון הוא חייב להיות source .

מכיוון שגודל ה- $\frac{|source|}{|V|} < 0.2$ נציב את ה $source$ בסט החוקים בעזרת אלגוריתם Davis Putnam וניצור גרף מחדש אחרי ההצבה.

```
-----------------------------------------------------------------------------
---------------------------------------
Here are the size of all the connected component in the graph:
sizeof :1

sizeof :196

sizeof :1

sizeof :1

-----------------------------------------------------------------------------
---------------------------------------
print the source size 1
```

באיור למעלה נראה שמכיוון שגודל ה- $\frac{|source|}{|V|} < 0.2$ נציב את ה $source$ בסט החוקים בעזרת אלגוריתם Davis Putnam וניצור גרף מחדש אחרי ההצבה.

```
-----------------------------------------------------------------------------
---------------------------------------
Here are the size of all the connected component in the graph:
sizeof :196

sizeof :1

sizeof :1

-----------------------------------------------------------------------------
---------------------------------------
print the source size 196
use Dismantle the CC method
S is: [-1, 1, -2, 2, -3, 3, -4, 4, -5, 5, -6, 6, -7, 7, -8, 8, -9, 9, -10, 10, -11, 11, -13, 13, -14, 14, -15, 15, -16, 16, -17, 17, -18, 18, -19, 19,
-20, 20, -21, 21, -23, 23, -24, 24, -25, 25, -26, 26, -27, 27, -28, 28, -29, 29, -30, 30, -31, 31, -32, 32, -33, 33, -34, 34, -35, 35, -36, 36, -37,
37, -38, 38, -39, 39, -40, 40, -41, 41, -42, 42, -43, 43, -44, 44, -45, 45, -46, 46, -47, 47, -48, 48, -49, 49, -50, 50, -51, 51, -52, 52, -53, 53, -
54, 54, -55, 55, -56, 56, -57, 57, -58, 58, -59, 59, -60, 60, -61, 61, -62, 62, -63, 63, -64, 64, -65, 65, -66, 66, -67, 67, -68, 68, -69, 69, -70, 70,
-71, 71, -72, 72, -73, 73, -74, 74, -75, 75, -76, 76, -77, 77, -78, 78, -79, 79, -80, 80, -81, 81, -82, 82, -83, 83, -84, 84, -85, 85, -86, 86, -87,
87, -88, 88, -89, 89, -90, 90, -91, 91, -92, 92, -93, 93, -94, 94, -95, 95, -96, 96, -97, 97, -98, 98, -99, 99, -100, 100, -101, 101, -102, 102, -103,
103, -104, 104, -105, 105, -106, 106, -107, 107, -108, 108, -109, 109, -110, 110, -111, 111, -112, 112, -113, 113, -114, 114, -115, 115, -117, 117, -
118, 118, -119, 119, -120, 120, -121, 121, -122, 122, -123, 123, -124, 124, -125, 125, -126, 126, -127, 127, -128, 128, -129, 129, -130, 130, -131,
131, -132, 132, -133, 133, -134, 134, -135, 135, -136, 136, -138, 138, -139, 139, -140, 140, -141, 141, -142, 142, -143, 143, -144, 144, -145, 145, -
146, 146, -147, 147, -148, 148, -149, 149, -150, 150, -151, 151, -152, 152, -153, 153, -154, 154, -155, 155, -156, 156, -157, 157, -158, 158, -159,
159, -160, 160, -161, 161, -162, 162, -163, 163, -164, 164, -165, 165, -166, 166, -167, 167, -168, 168, -169, 169, -170, 170, -171, 171, -172, 172, -
173, 173, -174, 174, -175, 175, -176, 176, -177, 177, -178, 178, -179, 179, -180, -181, 181, -182, 182, -183, 183, -184, 184, -185, 185, -186, 186, -
187, 187, -188, 188, -189, 189, -190, 190, -191, 191, -192, 192, -193, 193, -194, 194, -195, 195, -197, -198, 198, -199, 199, -200, 200, 197]
S size is :389
Number of vertex to remove 1
-----------------------------------------------------------------------------
```

באיור למעלה נראה שמכיוון שגודל ה- $\frac{|source|}{|V|} > 0.2$ נפעיל את האלגוריתם שמחזיר קודקודים לפירוק. לאחר ריצת האלגוריתם שמחזיר קודקודים לפירוק ניתן לראות שיש 195 קודקודים ב S וקודקוד 1 ב-T .
האלגוריתם יחזיר לכן קודקוד 1 שיפרק את רכיב הקשירות. נציב את הקודקוד היחיד הנ"ל במערכת החוקים וניצור גרף חדש.

```
Here are the size of all the connected component in the graph:
sizeof :1

sizeof :194

sizeof :1

sizeof :1

---------------------------------------------------------------------------
---------------------------------------
print the source size 1
```

באיור שלמעלה רואים שהרכיב הגדול ביותר התפרק למספר רכיבים, ביניהם רכיבים עם 194 קודקודים ורכיבים עם קדקוד 1.

```
----------------------------------------
Here are the size of all the connected component in the graph:
sizeof :193

sizeof :1

sizeof :1

sizeof :1

---------------------------------------------------------------------------
----------------------------------------
print the source size 193
use Dismantle the CC method
S is: [161] S size is :1
Number of vertex to remove 1
---------------------------------------------------------------------------
```

באיור למעלה נראה שמכיוון שגודל ה- $\frac{|source|}{|V|} > 0.2$ נפעיל את האלגוריתם שמחזיר קודקודים לפירוק. לאחר ריצת האלגוריתם שמחזיר קודקודים לפירוק ניתן לראות שיש 1 קודקודים ב S וקדקוד 192 ב-T .

האלגוריתם יחזיר לכן קדקוד 1 שיפרק את רכיב הקשירות. נציב את הקדקוד היחיד הנ״ל במערכת החוקים וניצור גרף חדש.

```
----------------------------------------
Here are the size of all the connected component in the graph:
sizeof :132

sizeof :1

sizeof :1

sizeof :1

------------------------------------------------------------------------------
----------------------------------------
print the source size 132
use Dismantle the CC method
S is: [-1, 1, -2, 2, -3, 3, -4, 4, -6, 6, -7, 7, -8, 8, -10, 10, -11, 11, -14, 14, -17, 17, -18, 18, -19, 19, -21, 21, -23, 23, -25, 25, -26, 26, -28,
28, -29, 29, -30, 30, -31, 31, -32, 32, -33, 33, -34, 34, -36, -37, 37, -38, 38, -39, 39, -41, 41, -42, 42, -43, 43, -44, 44, -45, 45, -46, 46, -47,
47, -48, 48, -51, 51, -53, 53, -54, 54, -55, 55, -56, 56, -57, 57, -58, 58, -60, 60, -61, 61, -63, 63, -64, 64, -65, 65, -66, 66, -67, 67, -68, 68, -
69, 69, -70, 70, -73, 73, -74, 74, -78, 78, -80, 80, -81, 81, -83, 83, -86, 86, -87, 87, -88, 88, -90, 90, -92, 92, -93, 93, -95, 95, -96, 96, -97, 97,
-98, 98, -100, 100, -101, 101, -102, 102, -103, 103, -104, 104, -106, 106, -107, 107, -110, 110, -112, 112, -113, 113, -114, 114, -120, 120, -124, 124,
-127, 127, -128, 128, -129, 129, -130, 130, -131, 131, -132, 132, -133, 133, -134, 134, -135, 135, -136, 136, -138, 138, -139, 139, -140, 140, -141,
141, -142, 142, -143, 143, -144, 144, -145, 145, -146, 146, -147, 147, -148, 148, -152, 152, -153, 153, -154, 154, -155, 155, -156, 156, -157, 157, -
158, 158, -162, 162, -163, 163, -164, 164, -167, 167, -170, 170, -171, 171, -173, 173, -174, 174, -175, -178, 178, -181, 181, -182, 182, -184, 184, -
185, 185, -186, 186, -188, 188, -190, 190, -192, 192, -194, 194, -195, -197, 197, 195] S size is :260
S is: [-1, 1, -2, 2, -3, 3, -4, 4, -6, 6, -7, 7, -8, 8, -10, 10, -11, 11, -14, 14, -17, 17, -18, 18, -19, 19, -21, 21, -23, 23, -25, 25, -26, 26, -28,
28, -29, 29, -30, 30, -31, 31, -32, 32, -33, 33, -34, 34, -36, -37, 37, -38, 38, -39, 39, -41, 41, -42, 42, -43, 43, -44, 44, -45, 45, -46, 46, -47,
47, -48, 48, -51, 51, -53, 53, -54, 54, -55, 55, -56, 56, -57, 57, -58, 58, -60, 60, -61, 61, -63, 63, -64, 64, -65, 65, -66, 66, -67, 67, -68, 68, -
69, 69, -70, 70, -73, 73, -74, 74, -78, 78, -80, 80, -81, 81, -83, 83, -86, 86, -87, 87, -88, 88, -90, 90, -92, 92, -93, 93, -95, 95, -96, 96, -97, 97,
-98, 98, -100, 100, -101, 101, -102, 102, -103, 103, -104, 104, -106, 106, -107, 107, -110, 110, -112, 112, -113, 113, -114, 114, -120, 120, -124, 124,
-127, 127, -128, 128, -129, 129, -130, 130, -131, 131, -132, 132, -133, 133, -134, 134, -135, 135, -136, 136, -138, 138, -139, 139, -140, 140, -141,
141, -142, 142, -143, 143, -144, 144, -145, 145, -146, 146, -147, 147, -148, 148, -152, 152, -153, 153, -154, 154, -155, 155, -156, 156, -157, 157, -
158, 158, -162, 162, -163, 163, -164, 164, -167, 167, -170, 170, -171, 171, -173, 173, -174, 174, -175, -178, 178, -181, 181, -182, 182, -184, 184, -
185, 185, -186, 186, -188, 188, -190, 190, -192, 192, -194, 194, -195, -197, 197, 195]
S size is :260
Number of vertex to remove 2
------------------------------------------------------------------------------
```

באיור למעלה נראה שמכיוון שגודל ה- $\frac{|source|}{|V|} > 0.2$ נפעיל את האלגוריתם שמחזיר קודקודים לפירוק. לאחר ריצת האלגוריתם שמחזיר קודקודים לפירוק ניתן לראות שיש 130 קודקודים ב S וקדקוד 2 ב-T .

האלגוריתם יחזיר לכן 2 קודקודים שיפרק את רכיב הקשירות. נציב את הקדקודים במערכת החוקים הנ״ל וניצור גרף חדש.

```
----------------------------------------
Here are the size of all the connected component in the graph:
sizeof :1

sizeof :129

sizeof :1

sizeof :1

sizeof :1

-----------------------------------------------------------------------------
```

באיור שלמעלה רואים שהרכיב הגדול ביותר התפרק למספר רכיבים, ביניהם רכיבים עם 129 קודקודים ורכיבים עם קדקוד 1.

דוגמאות ללא ספאראטור:

```
-----------------------------------------------------------------------------
----------------------------------------
Here are the size of all the connected component in the graph:
sizeof :1

sizeof :1

sizeof :196

sizeof :1

sizeof :1

-----------------------------------------------------------------------------
----------------------------------------
```

באיור למעלה ניתן לראות ריצה ראשונית על כל רכיבי הקשירות.

נציב בסט החוקים כל source שמופיע.

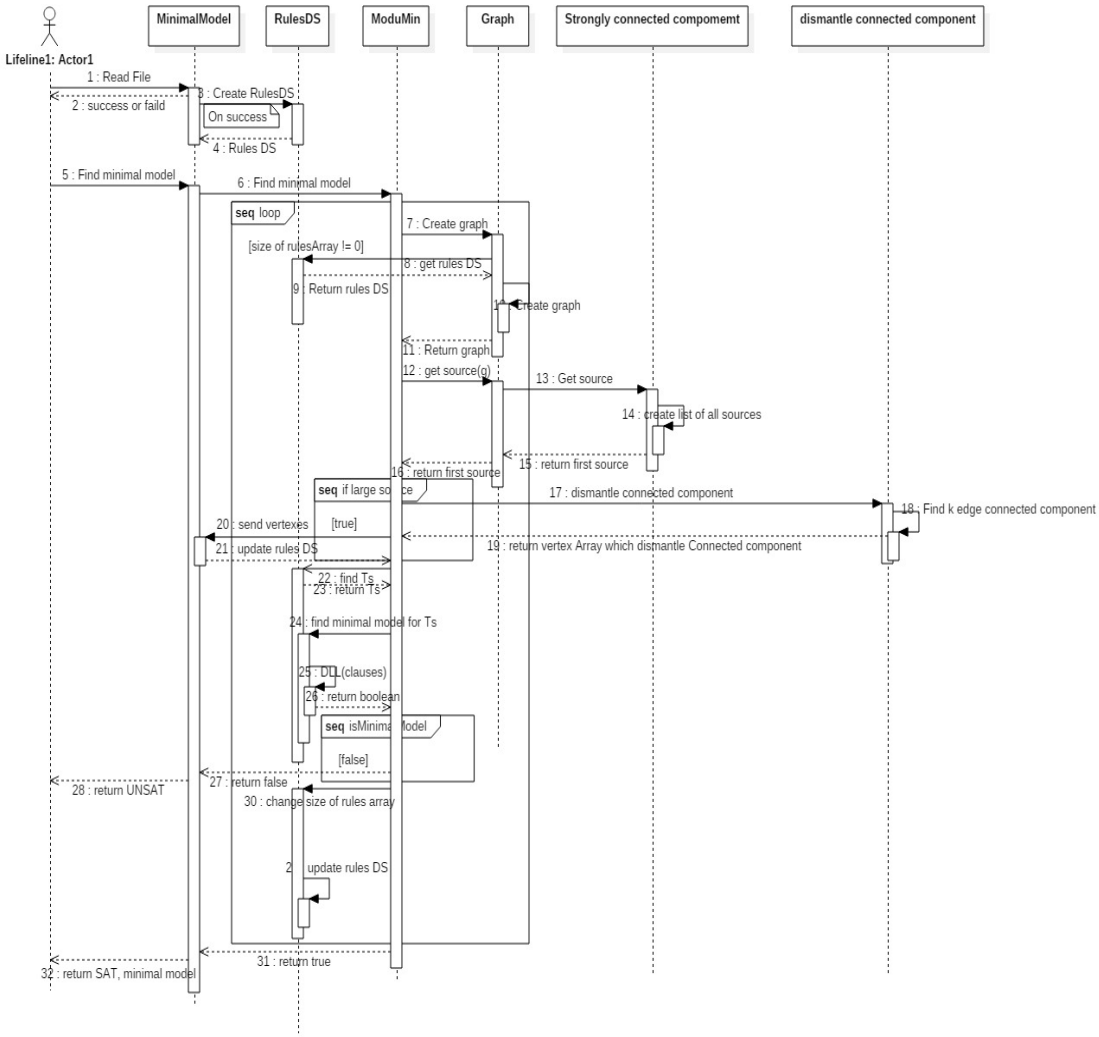
```
-----------------------------------------------------------------------------
----------------------------------------
Here are the size of all the connected component in the graph:

sizeof :196

sizeof :1

sizeof :1

-----------------------------------------------------------------------------
```

באיור למעלה ניתן לראות שאין כל הבחנה בין source גדול לקטן, כלומר ההצבה בחוקים של sourceה תתבצע ללא כל הבחנה.

זמן ריצה של source גדול במיוחד ישפיע הרבה יותר הדוגמאות ללא ספאראטור.

ט. תרשימים

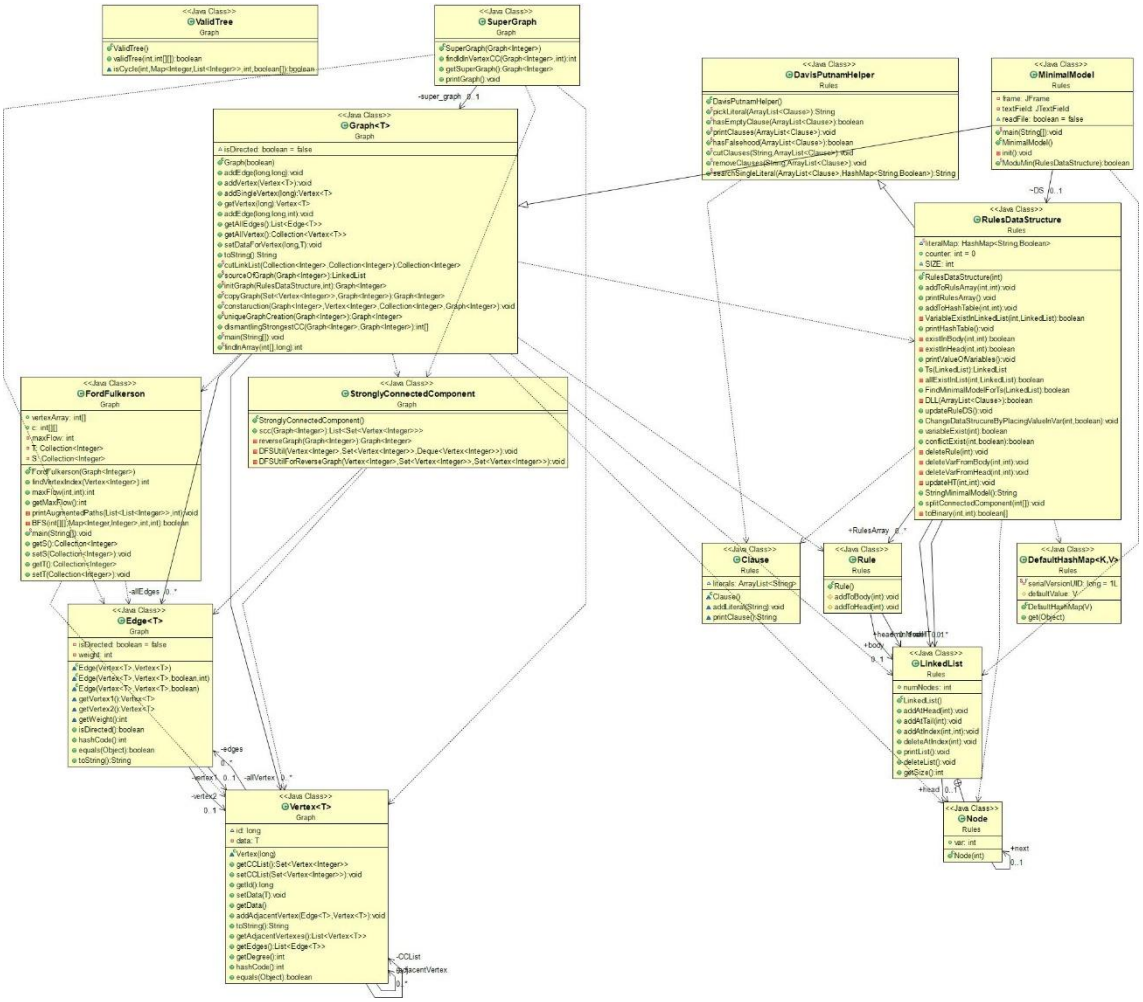
Sequence diagram



1. בדיאגרמה זאת נתאר את התהליך שבו המשתמש שולח קובץ מוגדר של חוקים, מבנה הנתונים של החוקים יודע לקרוא את הקובץ, להמיר את רשימת כל הפסוקיות בקובץ למבנה נתונים דינאמי .

2. מחלקת מבנה הנתונים של החוקים ישלח את מבנה הנתונים של החוקים למחלקת מבנה הנתונים של הגרף.
3. ניצור גרף שייצג את כל החוקים. כאשר נסיים את בניית הגרף נמצא את רכיב הקשירות הגדול ביותר. כעת יש לנו סופר גרף של רכיבי קשירות והראשון בניהם הוא ה source.
4. אם ה source קטן נציב בכל החוקים של המשתנים שמופיעים ב source ואז נעדכן את סט החוקים .
5. אחרת אם רכיב הקשירות גדול נשלח למחלקת dismantle
   5.1 ניצור גרף מרכיב הקשירות
   5.2 נריץ אלגוריתם [40] שיצור לנו גרף A .
   5.3 נחזיר מספר קודקודים שאם נסיר אותם אז נצליח לפרק את רכיב הקשירות.
6. באמצעות סט הקודקודיים הנ״ל נעבור על כל האפשרויות של ההצבות שלהם ונפרק את רכיב הקשירות ואז נעדכן את סט החוקים.
7. אם קיים מודל מינימלי נחזיר אותו אחרת נחזור לסעיף 3 .

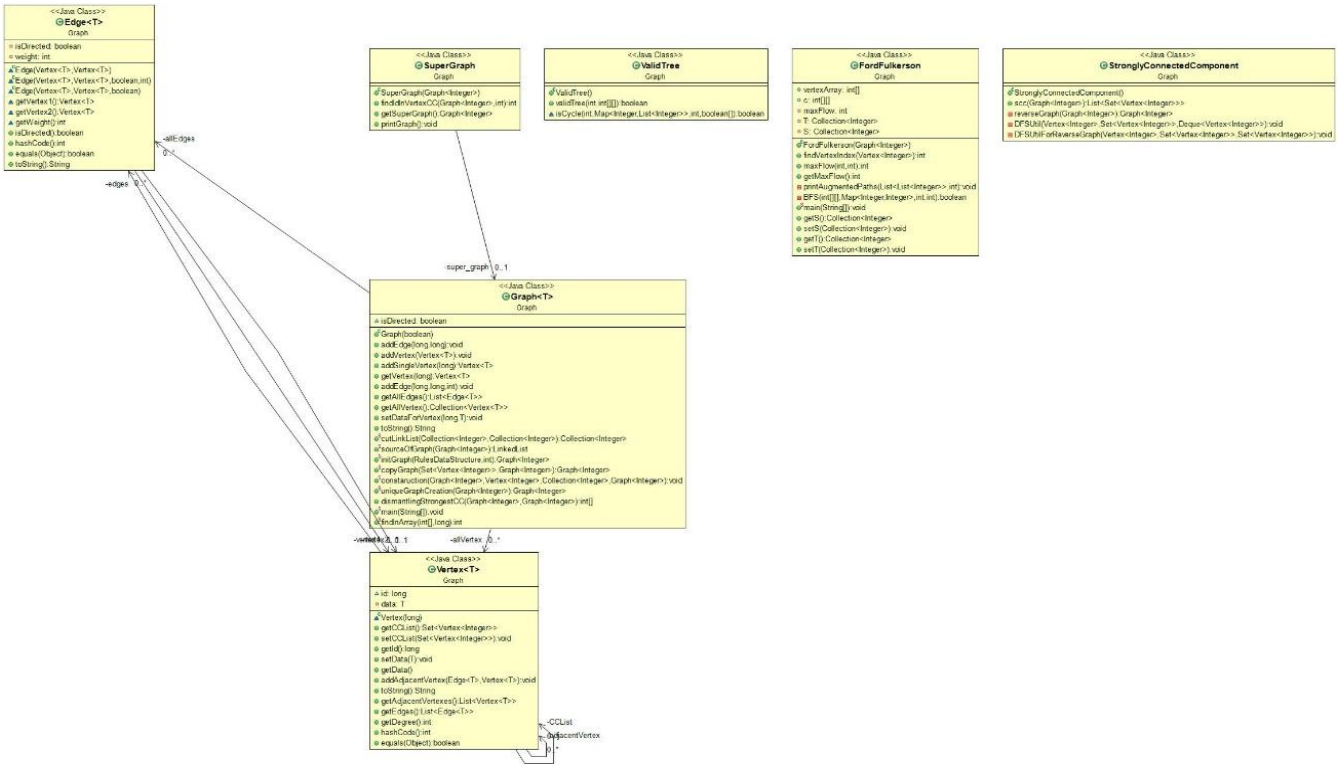
Class diagram



התרשים הבה מייצג את יחסי המחלקות שידועות עבור מחלקת החוקים ומחלקת הגרף.

בנקודת מבט זו ניתן לראות את כל הפונקציות שמשתמשות מבנה הנתונים של החוקים בפונקציות של מבנה הנתונים של הגרף .

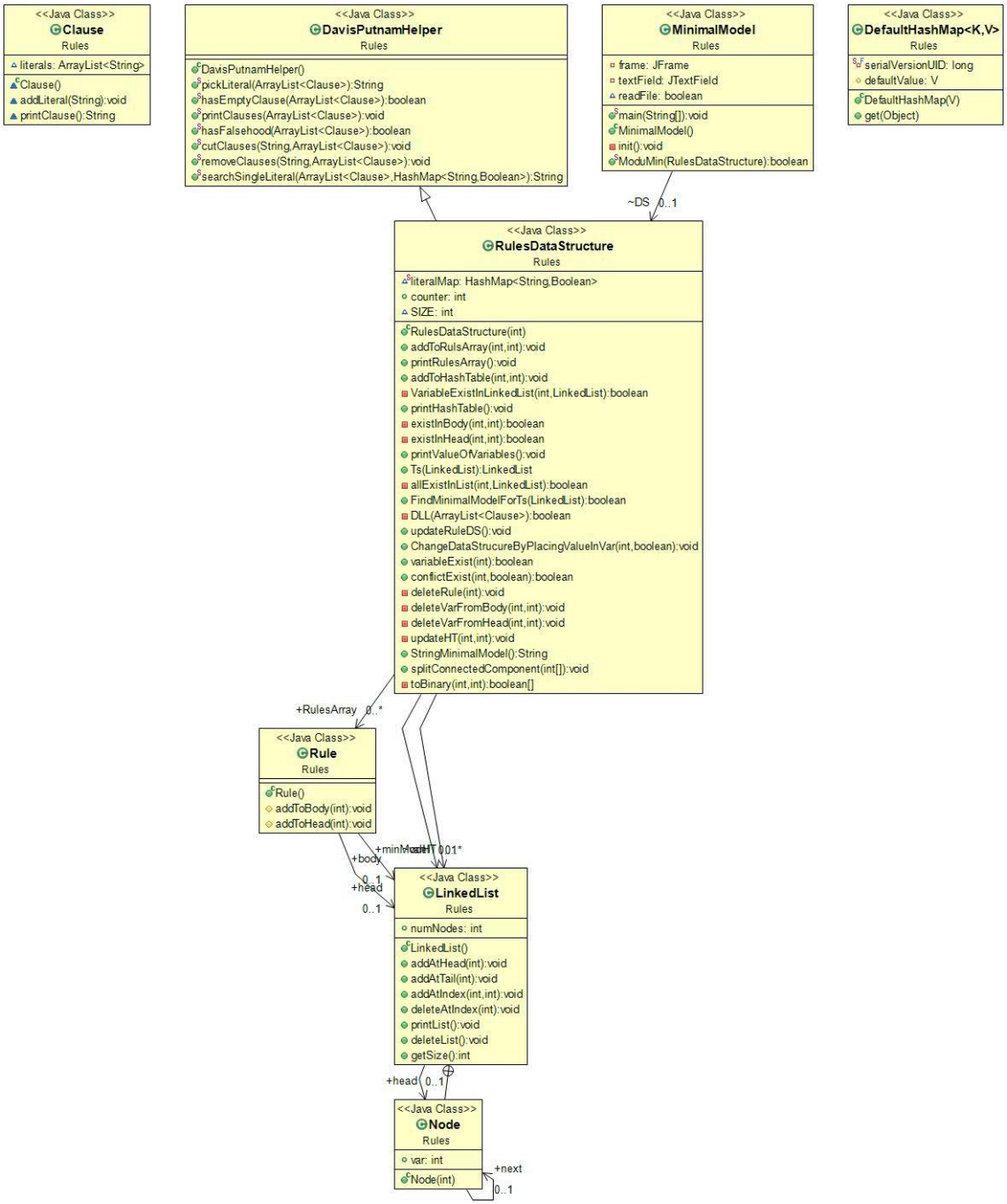
כאן גם ניתן לראות שהבדיקה שמחליטה אם להשתמש בסאפרטור תהיה במחלקת החוקים.



התרשים הבה מייצג את יחסי המחלקות שידועות עבור מחלקת הגרף.

בנקודת מבט זו ניתן לראות את כל המחלקות שמשתמשות אך ורק בפונקציות של מבנה הנתונים של הגרף .

יצירת גרף מכלילה בתוכה יצירת קודקודים ועבור כל קדקוד יש לו רשימה של קודקודים שכנים שלו. לגרף יש בנוסף רשימה של קשתות שמכילות את קדקוד המוצא וקדקוד היעד.



התרשים הבה מייצג את יחסי המחלקות שידועות עבור מחלקת החוקים.

בנקודת מבט זו ניתן לראות את כל המחלקות שמשתמשות אך ורק בפונקציות של מבנה הנתונים של החוקים.