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 חסין יהודה   אישור: תאריך:

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

עבודה משותפת עם הפרויקט של עדי טיירי

הצהרה:

העבודה נעשתה בהנחיית דר׳ חסין יהודה ופרופ' בן אליהו זוהרי רחל במחלקה להנדסת תוכנה, עזריאלי - המכללה האקדמית להנדסה ירושלים ובשיתוף פעולה עם הסטודנט עדי טיירי, כאשר הפרויקט פוצל לשני חלקים אשר כתוצאה מכך כל סטודנט הגיש פרויקט בנפרד.

החיבור מציג את עבודתי האישית ומהווה חלק מהדרישות לקבלת תואר ראשון בהנדסה.

עזריאלי מכללה אקדמית להנדסה ירושלים
Azrieli College of Engineering Jerusalem

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 כפי שמוצגת היא מציאת פירוק קודקודים מינימלי שמפרק רכיב קשירות של גרף מכוון בצורה מאוזנת.

אנו נציע כפתרון את אלגוריתם [45],אשר יקרא FM על שם ממציאיו, כאשר מטרתו היא מציאת מספר מינימלי של קודקודים אשר יהוו פירוק מאוזן לגרף לא מכוון, נשתמש במידע שאלגוריתם זה נותן בשביל למצוא קדקודים שאם נסיר אותם מהגרף נצליח לפרק את רכיב הקשירות הגדול ביותר בגרף מכוון.
הבעיה ניתנת לשימוש למגוון אלגוריתמים , לרבות אלגוריתם [1], אשר משתמש בו כדי להקטין את זמן הריצה.

הבעיה כפי שמוצגת עבור אלגוריתם [1], 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 חישוב מודלים מינימליים באמצעות גרף התלויות 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 אלגוריתם ממאמר [45], מאמר זה מאפשר לנו למצוא מספר קודקודים מינימלי שכדאי 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.

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 אלגוריתם שטרם מומש והתבססות על מחקרים דומים שנעשו.

מילון מונחים , סימנים וקיצורים :

1) V- כמות הקדקודים בגרף.
2) S- בהינתן חתך בין 2 קודקודים בגרף, S מכיל את קודקודים שבצד של הקדקוד שממנו מתחילים את ההזרמה. לכל קדקוד נשמור את גודל S .
3) K- גודל החתך.
4) A – גרף עזר שמתקבל כתוצאה מהרצת אלגוריתם [40], באמצעות הגרף נמצא את E׳, ישנה הוכחה שגרף זה הוא עץ.
5) W - קבוצת קודקודים אשר מוגרלת מתוך קבוצת כל הקדקודים, V בגרף קבוצה זו חייבת להיות קטנה ולא יותר מ 32 קודקודים מסך הקדקודים בגרף כאשר נתחיל בגודל W 2 ונעלה כל פעם פי 2 עד שנגיע ל -32 .
6) A,B– 2 קבוצות שמוגרלות בסיכוי שווה מקבוצת הקדקודים W, תנאי הכרחי זה שאין קשתות בין הקדקודים בקבוצה A לקדקודים בקבוצה B .
7) #AB - זה מספר הפעמים אשר נגריל תתי קבוצות A,B ל- קבוצה W נתונה.
8) איזון לקבוצות S,T – רמת האיזון אחרי שנעשה אלגוריתם זרימה ונמצא חיתוך בין שני קודקודים. זה יקבע את גודל האיזון של הפירוק של הספראטור שהתקבל.
9) קדקוד s – קדקוד שנוסיף בצורה מלאכותית לגרף אשר יהיו קשתות מכוונות ממנו לכל הקדקודים בקבוצה A .
10) קדקוד t - קדקוד שנוסיף בצורה מלאכותית לגרף אשר יהיו קשתות מכוונות אליו מכל הקדקודים בקבוצה B .
11) 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 .
12) מודל – השמה שמספקת את סט החוקים.
13) מודל מינמלי – הוא מודל עבורו מספר ההשמות של ערכי TRUE במשתנים עבור סט של חוקים הוא מינימלי (קיים הסבר מפורט יותר במבוא).
14) רכיב קשירות- בגרף רכיב קשירות הוא אוסף של קודקודים שמכל קדקוד ניתן להגיע לקדקוד אחר.
15) סופר גרף- גרף של רכיבי קשירות בהחלט, כל קדקוד יהיה בעצם רכיב קשירות בגרף המקורי.
16) Source – קדקוד בסופר גרף אשר לא נכנסים אליו קשתות אלה רק יוצאים ממנו קשתות.
17) 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.
18) Source גדול- גודלו צריך לקיים $\frac{|source|}{|V|} > 0.2$
19) E׳ – סאפארטור של קשתות, רשימת קשתות שאם נסיר אותם נפרק את רכיב הקשירות.
20) V׳ – סאפארטור של קודקודים.
21) $t=\left||S|-\frac{|v|}{2}\right|$

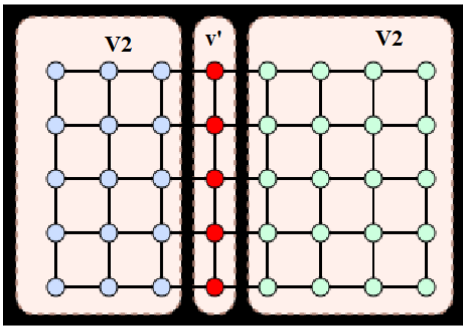
1. מבוא

<u>מציאת פירוק קודקודים מינימלי ומאוזן :</u>

בפרויקט זה נעסוק במציאת מספר מינימלי של קודקודים V׳, אשר אם נסיר אותם מהרכיב הקשירות, רכיב הקשירות יפורק אם זה בצורה מאוזנת או לגמרי.

בעיה זו הוכחה כ-NP שלמה [43], אנו מציעים אלגוריתם [45] אשר פותר אותה בזמן מעריכי בגודל ׳|V.

<u>פירוק קודקודים הגדרה :</u>



בהינתן

v1,v2⊂v

$$v = v1 \cup v' \cup v2$$

$$x \in v1, y \in v2$$

מתקיים :

$$(x,y) \notin E \quad \text{AND} \quad (x,y) \notin E$$

<u>פירוק מאוזן:</u>

V׳ יקרא פירוק מאוזן אם לאחר הפירוק גודל רכיב הקשירות הגדול ביותר בגרף יהיה קטן שווה V $\frac{2}{3}$.

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 מכיוון שללא גרף מכוון לא נצליח לעקוב אחרי הקשר בין הקדקודים למקביליהם המשתנים בסט החוקים.

עבור גרף עם קשתות מכוונות אין בכלל מאמרים או אלגוריתמים ממומשים או תוכנות אשר מציעות פתרון יעיל לפירוק רכיב קשירות בצורה מאוזנת האתגר המרכזי הוא למצוא תיאוריה או מאמר בשביל שנוכל לפרק רכיב כשירות כפי שהוצע לפתרון מאמר [1].

חשוב לציין, הדרישה שהפירוק יהיה מאוזן לגמרי זאת בעיה NP שלמה [43] וככלל שנבקש פירוק פחות מאוזן כך הבעיה תהיה יותר ליניארית.

הפירוק נעשה בכמה שלבים, בהינתן רכיב קשירות גדול שאותו צריך לפרק אנו נשתמש באלגוריתם ממאמר [45], בהינתן גרף שכולו רכיב קשירות האלגוריתם אנו נגריל תת קבוצה W מתוך V. מתוך W נמצא תתי קבוצות A,B . האלגוריתם מעביר את גרף הקלט לבניית גרף עזר כדי לאחסן מידע אודות קישוריות קצה בין כל זוג קודקודים בהתחשב בתתי הקבוצות A,B בזמן O (F(n)), כאשר F זה המורכבות בזמן כדי למצוא את הזרימה המקסימלית בין שני קודקודים בתרשים G ו- |V| = n. אנו נמצא את החיתוך המינימלי אשר מפריד בין קדקוד s , t. מאלגוריתם הזרימה על הגרף העזר הנ"ל נרצה לקבל מידע אשר נמצא V' קטן ככל האפשר ואשר ייתן לנו פירוק מאוזן ככל האפשר.

הפירוק במאמר [1] זאת בעיה NP שלמה מכיוון שהיא תלויה בגודל |V'|, גודל זה משתנה ולא קבוע ויכול להיות גבוה במיוחד במקרים קשים.

<u>שילוב הפרויקט:</u>

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 ($\neg x \vee \neg z \vee \neg w$) 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, במקרים שבו ה-source בגודל הגרף כולו 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.

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 המרכזיות זה מציאת חומר רקע שעליו ניתן יהיה להסתמך כמקור, אין בכלל אלגוריתמים אשר מפרקים גרף מכוון, לכן אי אפשר יהיה לבחון את האלגוריתם מולם.

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

הבעיה בהפרדת רכיב הקשירות

הבעיה הינה למצוא v' קטן אשר אם נסיר אותם מהגרף יפרק רכיב הקשירות ל- $\frac{2}{3}V$ (זה גודל האיזון שנבקש).

אנו מיישמים תאוריה כפי שמוצאת במאמר [45], מטרתו היא מציאת חיתוך מינימלי בין שתי קדקודים s ו t, אשר נמצאים בשתי תתי קבוצות A,B. מאמר זה הוא תיאורטי לגמרי שאין לו יישומים.

אחד מהיתרונות של מציאת חיתוך זה שניתן באמצעותה למצוא את V' כפי שניתן לראות באיור 2 בזמן ריצה O(|E|+|V|). אך לא כל V' ייתן לנו פירוק של G בצורה מאוזנת, לשם כך אנו נחפש V' מינימלי מבין כלל V' שיתקבלו לנו אשר ייתן פירוק מאוזן. אנו נצטרך למצוא איזון בין הדרישה שימצא V' מינימלי ומאוזן לבין הצורך שהאלגוריתם לא ירוץ על כלל האפשרויות של הפירוקים האפשריים להגרלת W נתונה.

מאמר [45] תלוי בהגרלות של W ובהגרלות של A,B לכן יש סבירות קטנה שיכשל כאשר יבחר W אשר ממנו אין אפשרות להגריל B,A כפי שאנו דורשים. בנוסף אנו מתבקשים במאמר להגריל W $2^{|W|}$ פעמים דבר שיבזבז לנו זמן ריצה.

אחת מהדרישות היא שעבור רכיב קשירות גדול יהיה לנו פירוק מאוזן שלו, לכן הבעיה במציאת V' קטנה היא שרכיב הקשירות יפורק 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 ונרצה למצוא לו פירוק בעזרת מספר מינמלי קודקודים אשר עדיין יפרק את G בצורה מאוזנת.

הבעיה מבחינת הנדסת תוכנה

נרצה לבנות אלגוריתם דינאמי אשר יוצר גרף כאשר הוא מקבל סט חוקים משתנה, מבנה הנתונים ימיר את החוקים לפי הפורמט המקובל (ראה מילון מונחים) כאשר יהיה קשת מקדקוד שנמצא ב-body אל קדקוד שמצא ב head. האלגוריתם צריך לתמוך ולנהל את מבנה הנתונים של הגרף בצורה יעילה. התממשקות לרחל

מבנה הנתונים ידע להתמודד עם אלגוריתמים מסובכים שישתמשו בו ועדיין לשמור על זמן ריצה ליניארי.

נממש אלגוריתם אשר ייתן פתרון חד משמעי לבעיה הנתונה, אשר יוכל לעבוד על כל מערכת הפעלה שעובד עליה java.

3. תיאור הפתרון

תיאור הפתרון המוצע

לאחר חיפוש מעמיק בספרות, אמנם יש המון ספרות על גרפים לא מכוונים אין בכלל על גרפים מכוונים, נמצאו מספר אלגוריתמים לגרפים מכוונים למציאת פירוק מינימלי ומאוזן אשר מהם ננסה למצוא פירוק מאוזן ומינימלי לגרפים מכוונים.

בסמסטר קודם נוסה מאמר [40] אשר מוצא קשתות E′ אשר יהוו פירוק לגרף, ניסינו בעזרת מידע שהאלגוריתם נתן למצוא את הקדקודים לפירוק רכיב קשירות.
אלגוריתם זה סיפק תוצאות לא מספקות, אנו אמנם קיבלנו רכיב קשירות מינימלי אך האלגוריתם לא פירק בצורה מאוזנת גרפים מסוגים כגון Grid.
ראה נספח ה׳ על אופן המימוש והתוצאות של האלגוריתם.

נבחר בפרויקט זה אלגוריתם נוסף, אלגוריתם FM :

<u>מבנה אלגוריתם FM:</u>

1. כל עוד $|W|<=32$:
   1.1. הגרל קודקודים מ-V ל-W כאשר הסיכוי להיכנס לקבוצה הוא x
2. כל עוד AB# קטן מ-$\min(200, 2^{|W|})$
   2.1. הגרלת תתי קבוצות A,B
   A,B תתי קבוצות בעלי קודקודים שונים של W
   אנו נגריל קודקודים מ- W בעלי סיכוי שווה להיכנס או לקבוצה A או לקבוצה B .
   בנוסף נוסיף דרישה שלא יהיו קשתות בין הקדקודים בקבוצה A לקדקודים בקבוצה B .
3. יצירת גרף רשת זרימה
4. הרצת אלגוריתם למציאת זרימה מקסימלית וחיתוך מינימלי.
5. קבלת הסאפארטור כאשר יש לנו שתי קבוצות T,S, מהחתך של S,T.

זמן ריצה: $n^{O(1)}2^{O(k\epsilon^{-2}\log(1/\epsilon))}$

כאשר k זה גודל הסאפארטור.
$\epsilon > 0$ זה אחוז אשר מקיים $\frac{|v'|}{|v|}-\epsilon \le \frac{|W \wedge V'|}{|W|} \le \frac{|v'|}{|v|}+\epsilon$
$\alpha > \frac{2}{3}$ זה גודל האיזון שאנו מבקשים לפי [45].
כלומר, האלגוריתם מכוון לפירוק רכיב הקשירות ב- $\frac{2}{3}V$.

פירוט על אופן מימוש האלגוריתם:

מכיוון שאין כמעט בכלל תיאוריות עבור גרפים מכוונים ורובם ככולם על גרפים מכוונים, לקחנו אלגוריתם כללי [45] שמחשב חיתוך מינימלי ובאמצעות החיתוך הנ"ל נמצא אילו קודקודים כדאי להוציא כדי להפריד את רכיב הקשירות.

הפתרון שלנו בעזרת באלגוריתם [45] נקרא לו FM על שם יוצריו.

בפרויקט זה אנו מיישמים אלגוריתם FM [45] שמציעה דרך למציאת V' מינימלי ע"י המרת הבעיה לחישוב החתך המינימלי. זמן הריצה שלו $n^{O(1)}2^{O(k\epsilon^{-2}\log(1/\epsilon))}$
כאשר k זה גודל הסאפארטור (אצלנו רשום בתור |V'|)שאנו דורשים
$\alpha > \frac{2}{3}$ זה גודל האיזון שאנו מבקשים לפי [45].
כלומר, האלגוריתם מכוון לפירוק רכיב הקשירות ב- $\frac{2}{3}V$

יישום האלגוריתם מתחלק לכמה שלבים :

i) הגרלת תת קבוצה W :
בגלל שגודל |V'| אינו ידוע אנו נרוץ על W בגודל שונה בשביל שנוכל ככה להגדיל את גודל |V'| שיתקבל לאחר מכן. ככה גם בהתאמה יגדל זמן הריצה.
W מהווה תת קבוצה של V שגודלה יכול לנוע בין 4 קודקודים ל- 32 בלבד.
אנו נתחיל בבחירה של גודל מקסימלי |W|=4 ועד |W|=32 כאשר נכפיל את הגודל המקסימלי האפשרי ל-W ב-2. עבור כל קדקוד אנו נגריל מספר בין 0 ל-1 ואם הוא קטן מ-0.1 אז הוא יכנס לקבוצה W.

ii) הגרלת תתי קבוצות A,B :
A,B תתי קבוצות בעלי קודקודים שונים של W. אנו נגריל קודקודים מ-W בעלי סיכוי משתנה בין 0.25 ל-0.75 להיכנס או לקבוצה A או לקבוצה B.
הסיכוי משתנה ע״מ שגודל A ו-B כי נרצה שהם יהיו בערך באותו גודל.
בנוסף נוסיף דרישה שלא יהיו קשתות בין הקדקודים בקבוצה A לקדקודים בקבוצה B.

הערה 1: את הגרלת A,B דורשים ב- [45] לבצע 2^W פעמים ע״מ שנקבל V' אידיאלי, אצלנו באלגוריתם משיקולים של זמן ריצה נבצע בין המינימום של 2^W ו-ערך בין 20-200 פעמים ונראה שככל שנגריל את A,B **יותר** פעמים ככה נקבל V' **קטן** יותר ושנותן פירוק יותר מאוזן.

הערה 2: אם נגדיל את מספר האיטרציות על W ככה נגדיל גם AB# בהתאמה.

iii) יצירת גרף רשת זרימה :
אלגוריתם למציאת זרימה מקסימלי מחזיר חיתוך על קשתות שאם נסיר אותם נפרק את G. בשביל למצוא פירוק קודקודים ניצור רשת זרימה שבאמצעותה נדע אילו קודקודים כדאי לפרק.

תהליך יצירת רשת זרימה:

בהינתן שתי תתי קבוצות A,B אנו ניצור גרף באופן הבא :

1) ניצור באופן מלאכותי 2 קודקודי עזר s,t (ראה מילון מונחים) קודקודים אלו נוצרים כי בהמשך נריץ אלגוריתם למציאת זרימה מקסימלית וחיתוך מינימלי על הגרף החדש שנוצר.
2) נקבע משקלים לקשתות :
   a. משקל קשת שבין וקדקוד s לכל אחת מהקדקודים ב-A יהיה |V|.
   b. משקל קשת שבין הקדקודים ב-B לקדקוד t יהיה |V|.
3) עבור כל קדקוד אשר **לא** נמצא ב-A,B אנו נבצע :
   a. עבור כל קדקוד x ניצור קדקוד x1 , x2 בגרף החדש אשר כל הקשתות שנכנסות ל-x יצביעו מעכשיו ל-x1 משקל כל קשת הוא |V| .
   b. כל הקשתות אשר יוצאות מ-x מעתה יצאו מ- x2 משקל כל קשת הוא |V|.
   c. ניצור קשת עם משקל 1 בין x1 ל- x2 .

רשת זרימה זאת נוצרת בזמן ריצה של O(V+E) ומטרתה היא שכאשר נריץ אלגוריתם זרימה בהמשך נוכל בעזרת הרשת להצביע על הקדקודים שיהיו V' המינימלי ומציאת קודקודים שאותם נרצה למצוא זרימה מהם.

iv) <u>הרצת אלגוריתם למציאת זרימה מקסימלית וחיתוך מינימלי :</u>
השתמשתי באלגוריתם מ-[46] עם מספר רב של שינויים
Ford Fulkerson method Edmonds Karp algorithm למציאת זרימה מקסימלית
וחיתוך מינימלי זמן הריצה שלו $O(VE^2)$.
התוצאה מהרצת האלגוריתם זה שהתקבלו שתי קבוצות T,S.
כל הקדקודים שבצד s של החיתוך שייכים ל-S והשאר שייכים ל-T.
אנו נוסיף דרישה ש- |T|>0.1 וגם |S|>0.1 , דרישה זאת הינה הכרחית מכיוון שאם גדלי |S| ו |T| יחסית מאוזנים כך גם גודל V' שיתקבל מהם יגרום לפירוק מאוזן של רכיב הקשירות כלומר G.
הקוד עבר שינוי רב מכיוון שהשתמש במטריצות בשביל לנהל את משקלי הקשתות, שיניתי את המטריצה לרשימות מקושרות דו ממידיות.

v) <u>קבלת V' כאשר יש לנו שתי קבוצות T,S :</u>
כאשר יש לנו 2 קבוצות שמייצגות חיתוך ה-V' יתקבל מהקדקודים שעל הקשתות שבין S ל- T.

הפתרון המוצע עבור הפרויקט המשותף:

1) <u>יצירת בסיס נתונים דינמי לגרף :</u>

בסיס הנתונים לגרף נלקח מקוד פתוח ב- GitHub [46].

2) <u>יצירת גרף מסט החוקים :</u>
ראה נספח ו'.

3) <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. זמן הריצה למציאת כל רכיבי הקשירות $O(E+V)$

4) <u>שלב שני יישום אלגוריתם ממאמר [45] למציאת V' :</u>
ראה פירוט בסעיף קודם.

5) <u>שמירת כל V' שהתקבל ומציאת המינימום :</u>
כחלק מהדרישה של [1] אנו נרצה להציב את המשתנים ש-V' מייצג אותם פעולה שכבר אמרנו שתהיה $O(2^{|v'|})$, לכן נרצה למצוא את V' המינימלי ע"מ לשמור על זמני הריצה קטנים ככל האפשר. בבדיקות נראה ש-V' מינימלי לא בהכרח ייתן פירוק מאוזן.

שימוש בפרויקט זה באלגוריתם "חישוב מודלים מינימליים באמצעות גרף התלויות":

הפרויקט של עדי טיירי "חישוב מודלים מינימליים באמצעות גרף התלויות" בפתרון בעיית SAT, ולכן יש למצוא עבור סט של חוקים האם קיימת השמה מספקת או לא.

אנו משתמשים בצורת סט חוקים של CNF ובנוסף לכך עבור כל clause לא קיים מצב בו כל המשתנים הם בשלילה, אם כך כל סט חוקים בצורה זו הוא SAT אך מטרתנו היא למצוא מודל מינימלי עבור סט החוקים, ז"א אנו רוצים לקבל ערך אמת מכל החוקים ע"י כמה שפחות הצבות של true במשתנים הקיימים.

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 [1]:

Algorithm 1: Algorithm ModuMin

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

$M := \emptyset$;
while $T \neq \emptyset$ **do**
  if *There is a clause* δ *in* T *violated by* M *such that* $|\text{head}(\delta)| = 1$ **then**
    let $X := \text{head}(\delta)$; $M := M \cup X$;
    $T := \texttt{Reduce}(T, X, \emptyset)$;
  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;
    $M := M \cup X$;
    $T := T - T_S$; $T := \texttt{Reduce}(T, X, S - X)$;
return M

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

ניתן לראות שהאלגוריתם שמוצע למציאת מודל מינימלי [1], 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. אנו נשתמש באלגוריתם FM ע"מ למצוא את הקודודים שיהוו פירוק לרכיב קשירות (source) גדול.

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).

אנו מיישמים אלגוריתם FM בשביל למצוא V' שנותן פתרון אידיאלי לבעיה ל [1]

בפרויקט זה מבנה הנתונים של הגרף מיוצג באמצעות קוד פתוח מ-github.

תכנית בדיקות

הבעיה בבדיקות

נעשה מספר בדיקות, על גרפים שיצרנו אותם מראש ועל סט חוקים מהצורה CNF.
אנו נרצה 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 ומירב התוכנות המוצעות ישנות ולא נתמכות יותר, לכן יהיה קשה להפעיל אותן וכמו כן להבין לעומק איך הן עובדות.
2. אנו נבקש למצוא V' מינימלי, לכן ניצור גרפים מסובכים ופשוטים אשר נדע מה הפירוק המינימלי שלהן וננסה לראות אם האלגוריתם שאנו מציעים ימצא את הפירוק הנ"ל.
3. נעשה בדיקות עבור גרפים רנדומליים שעבורם FM מסיקה שאין V' מינימלי שיפרק אותם, כלומר לא עובדת עליהם.
4. 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.
5. בדיקה נוספת זאת בדיקת זמני ריצה – לאלגוריתם [45] זמן הריצה תלוי ברמת האיזון שנרצה, אך נעשה בדיקות מול התוכנות הקיימות אשר מוצאות V', וב- [44] אולי זמן הריצה שאנו מציעים לא יעיל לעומתם.
6. בדיקת מאמץ שתנוע מסט קטן לסט גדול של חוקים.
7. בדיקת מאמץ עבור יחס שונה של חוקים למשתנים, נתחיל מיחס גבוה ולאט לאט נקטין את כמות המשתנים לחוקים.
8. בדיקה שימוש בזיכרון עבור גדלי AB# שונים.

9. בדיקה עבור |W| קבוע עבור כל הריצה.
10. בדיקה זמן ריצה עבור כל חלק באלגוריתם.
11. בדיקה שימוש בזיכרון עבור כל חלק באלגוריתם.
12. בדיקת זמן ריצה עבור גדלי AB# שונים.
13. בדיקה משולבת של זמן ריצה, שימוש בזיכרון במציאת V' מינימלי עבור כל AB#.
14. בדיקה משולבת של זמן ריצה, שימוש בזיכרון במציאת רכיב הקשירות המקסימלי עבור כל AB#.
15. בדיקה עבור כל AB# ו-5 V' המינימלים מה רכיב הקשירות המקסימלי שהתקבל.
16. ציור גרפים רלוונטיים וטבלאות עבור כל הבדיקות המוזכרות.

תהליך הבדיקות

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.
בתרשימים נציג חלק מהבדיקות שנעשו עד כה.

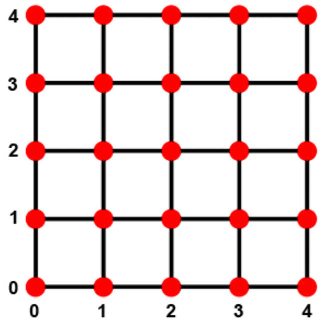
כאמור הבדיקות שלנו בפרויקט זה מתחלקות ל-2 :
1. בדיקות עבור קלטים שהם גרפים.
2. בדיקות עבור קלטים שהם פסוקיות מהצורה CNF .

<u>בדיקות עבור קלטים שהם גרפים:</u>

ישנם מספר תוכנות מוכנות למציאת V' כגון :
METIS, PARTY, CHACO , JOSTLE, SCOTCH, GNU, Kahip.
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.

<u>סוגי הבדיקות שנעשו :</u>

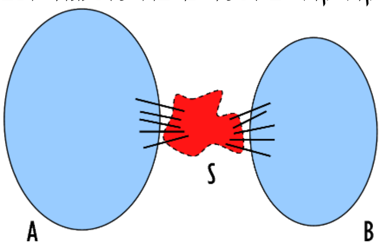
1. בדיקות עבור גרפים מסוגים שונים :
(ראה טבלה עבור הפלט שלהם)
   a. Grid – גרף לא מכוון שאנו יודעים שגודל הפירוק שלו חייב להיות $\sqrt{V}$ לפי [47] כפי שניתן לראות האלגוריתם מפרק את הגרף בצורה מאוזנת בעזרת אחד מ-5 ה-V' המינימלים שנקבל.



תוצאה רצויה:

תוצאה:
דווקא ב'V המינימלי הגרף לא יפרק את הרכיב בצורה הכי אידיאלית.

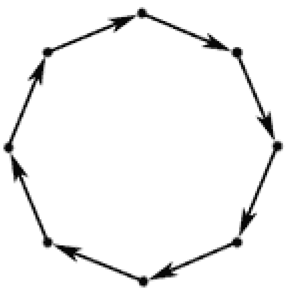
b. גרף שניצור בצורה מלאכותית משתי רכיבי קשירות שונים אשר מחברים ביניהם 4 קודקודים אשר נרצה שאלגוריתם שלנו ימצא אותם.



תוצאה רצויה:
האלגוריתם ימצא את הקדקודים אשר ב-S וכך נקבל 2 רכיבי קשירות מאוזנים.

תוצאה:
האלגוריתם מוצא את הקדקודים ב-S אך לא בכל הרצה .
האלגוריתם נותן פירוק מאוזן אם קיבלנו קודקודים ב-S או לאו.
אם הפירוק מאוזן ועדיין לא קיבלנו את הקודקודים ב-S המסקנה היא שרכיבי הקשירות שנבחרו לא מספיק מסובכים בשביל שהאלגוריתם יעדיף את הקודקודים ב-S.

c. רכיב קשירות מעגלי שכל קדקוד יש לו קשת אחת בלבד ואין קודקוד שנכנסים אליו 2 קשתות.



תוצאה רצויה:

האלגוריתם יחזיר קודקוד אחד בלבד אשר יפרק את רכיב הקשירות לגמרי

תוצאה:

האלגוריתם מוצא |V'| =1 שמפרק את רכיב הקשירות המעגלי לגמרי. ניתן לראות את ההרצה בדוגמאות הרצה.

d. בדיקת מאמץ – נבנה גרף בעל 200-2000 חוקים בקפיצות של 200 ונמצא את גודל הפירוק המקסימלי ואת גודל V' ונשנה את היחס בין החוקים למשתנים.

תוצאה רצויה:
האלגוריתם ימצא את V' המינימלי .

תוצאה:
האלגוריתם מוצא V' המינימלי אך פועל ממש לאט כאשר אנו עוברים את 1000 חוקים וביחס 4.1. ראה נספח

e. בדיקת מאמץ – נבנה גרף בעל 200-2000 חוקים בקפיצות של 200 ונמצא את גודל הפירוק המקסימלי ונשנה את היחס בין החוקים למשתנים.

תוצאה רצויה:
האלגוריתם מוצא את גודל רכיב הקשירות המקסימלי מבין 5 V' המינימלים.

תוצאה:
האלגוריתם את גודל רכיב הקשירות המקסימלי של רכיב הקשירות הגדול ביותר מבין 5 V' המינימלים אך פועל ממש לאט כאשר אנו עוברים את 1000 חוקים וביחס 4.1.

f. בדיקה עבור גרף רנדומלי

תוצאה רצויה:

האלגוריתם ימצא V' אך לא ייתן פירוק אידיאלי לפי FM, כי לגרפים רנדומליים יכול להיות שאין V' קטן.

תוצאה:

האלגוריתם מוצא V' גדול ובגודל רכיב הקשירות כולו, כלומר ל-G.

להלן תוצאות הבדיקות:

סוג הגרף	|V|	Min |V'|	Max |CC|	Run time	Memory usage
Grid 5X5	25	4	16	3000millisec	0.982673645019535MB
מעגלי	20	1	1	953millisec	0.978729248046875MB
הרכבה של 2 רכיבי קשירות	62	5	29	5621millisec	0.989746513957546MB

Max |CC| - גדול רכיב הקשירות המקסימלי לאחר הפירוק.

Min |V'| - גודל הפירוק המינימלי.

<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 המעריכים 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 וגודל V' :

נריץ 20 פעם קובץ אקראי של 100 משתנים ו-500 חוקים ונבדוק את הדברים הבאים :

a) בדיקת זמן הריצה ב- millisecond לעומת מספר החזרות על ההגרלה של A,B (#AB) השונים [5,10,20,50,100,200]
   - ראה תרשימים Run Time Test :

תוצאה:
ניתן לראות שככל שאנו מגדילים את מספר החזרות ככה גדל זמן הריצה, תוצאה שהייתה צפויה. מה שלא צפוי זה זמן הריצה הגדול במיוחד. האלגוריתם שמיושם ב-[1] שמבקש פירוק בשביל לקצר זמן ריצה לא יוכל להשתמש באלגוריתם שאנו מציעים מכיוון שהוא מגדיל את זמן הריצה באופן משמעותי. לכן צריך לפרק את האלגוריתם לשלבים ולבדוק מה צורך את הכי הרבה זמן ריצה.

b) פירוק האלגוריתם לחלקים ע"מ לבודד את השלב שצורך את זמן הריצה המרבי :
עבור כל שלב בבניית האלגוריתם ניצור טבלה עבור כל #AB (ראה מילון מונחים).
   - ראה תרשימים פירוק האלגוריתם לחלקים.

תוצאה:
ניתן לראות בבירור שהחלק שצורך הכי הרבה זמן ריצה זה השימוש באלגוריתם זרימה Ford Fulkerson על כל חלקיו. הקפיצה בזמני הריצה מעידה על כך שזה בעיה של שימוש בזיכרון ולא בזמני ריצה.

c) בדיקות על יעילות הפירוק :
ניצור גרף אשר יעיד על גודל V' בהגדלת טבלה #AB :
   - ראה תרשימים גרף של תלות מספר הרצות #AB לגודל ה-V' המינימלי שיתקבל.

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

תוצאה:
ניתן לראות שככל שאנו מגדילים את AB# ככה נקבל V' קטן יותר. תוצאה שתואמת לאלגוריתם FM וטובה לפרויקט של " חישוב מודלים מינימליים באמצעות גרף התלויות" [1].

d) בדיקות על יעילות הפירוק:
עבור כל עבור כל AB# ניצור טובלה שבה 5 הפירוקים המינימלי ביותר שהתקבלו עבור AB# נתון . נמצא עבור כל V' מינימלי את גודל רכיב הקשירות הגדול ביותר לאחר הפירוק.
תוצאה:
a. כפי שראינו בגרף בסעיף (3) ניתן לראות שככל שאנו נגדיל את AB# ככה נקבל V' יותר קטן ובהתאמה גם 5 V' המינימלים יהיו יותר קטנים בהתאמה.
b. V' קטן אינו בהכרח מבטיח את הפירוק הכי יעיל, מאוזן. אנו רואים מתוצאות הבדיקה שהפירוק האידיאלי ביותר אינו דווקא מגיע מה-V' הכי קטן.
התוצאות לגבי איזה V' ב-5 המינימלים הכי כדאי לנו לבחור הן לא חד משמעיות. לכן יידרש חקירה נוספת לגבי איזה V' נבחר.
c. ניתן לראות שעבור AB# קטן יכול שנקבל פירוק יותר מאוזן.

e) בדיקה עבור גודל W=16 קבוע מה יעילות הפירוק:
עבור AB# משתנה נראה מה גודל רכיב הקשירות הגדול ביותר לאחר הפירוק.
- ראה נספח בדיקת יעילות הפירוק כאשר |W|=16 קבוע

תוצאה
ניתן לראות שהגבלת |W| משפיעה מאוד על הפירוק שנקבל, הוא עדיין נשאר גדול לא משנה איזה AB# נבחר.

f) בדיקה מציאת |V'| מינימלי כאשר AB# משתנה ו- |W|=16 קבוע (ראה בנספח את הגרף)

תוצאה
ניתן לראות שהגדלת AB# באמת מקטינה את |V'|, כפי שהתאוריה [45] מוכיחה.

g) בדיקה שמטרתה לקבל מידע כאשר AB=200# קבוע ונמצא |V'| מינימלי עבור כל W.
- ראה תרשימים AB=200# קבוע ונמצא |V'| מינימלי עבור כל W

תוצאה
בכל איטרציה של W גודל |W| שנמצא נשאר קבוע, מכאן שאלגוריתם מוצא פעם או פעמיים קבוצות W אחרות. מכן מסיק שגודל |V'| המינימלי יהיה זהה, מכיוון שגודל W משפיע על גודל |V'|.

h) נריץ בדיקה שמטרתה לקבל מידע כאשר AB=200# קבוע ונמצא גודל רכיב קשירות מקסימלי.
- ראה תרשימים AB=200# קבוע ונמצא גודל רכיב קשירות מקסימלי.

תוצאה

בכל איטרציה של W גודל |W| שנמצא נשאר קבוע, מכאן שאלגוריתם מוצא פעם או פעמיים קבוצות W אחרות. גודל רכיב הקשירות המקסימלי משתנה כל פעם, אבל הוא נשאר באותו סדר גודל.

2. בדיקות כנגד אלגוריתם Metis :

Metis זה אלגוריתם שפועל על גרף לא מכוון, הוא מיוחד מהבחינה שאתה יכול לבקש לכמה חלקים לפרק את הגרף והאלגוריתם מפרק את הגרף בזמן ריצה קצר משמעותית מזמן הריצה של האלגוריתם שלי, ובפחות זיכרון.

ניתן לראות בנספחים את דוגמאות הרצת Metis על גרף מסוג Grid אשר גרם לאלגוריתם [40] אשר מימשנו גם להיכשל.

אולם התוכנה **לא** מחזירה אילו קודקודים היא משתמשת לפירוק ולכן אלגוריתם [1] אולי לא יוכל להשתמש בה כראוי כי היחס בין החוקים לא נשמר במצב כזה.

תוצאה:

ראה נספח ז׳

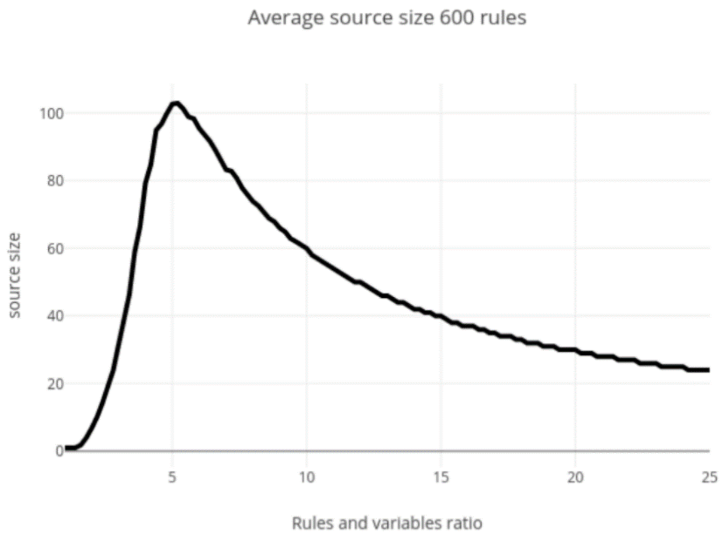
עבור גרף בגודל 5x5 כאשר ביקשתי מהאלגוריתם לפרק ל-2 חלקים.

Metis מפרק את הגרף הלא מכוון לחלקים כמעט שווים (מכיוון שיש מספר אי זוגי של קודקודים), מגרפים בעלי מספר זוגי הוא מפרק בדיוק מספר זוגי של קודקודים.

באותו אופן ניתן היה לבקש ממנו לחלק את הגרף ל-25 חלקים ואז הוא היה מחזיר כל קדקוד בנפרד.

3. בדיקה יחס משתנים וחוקים לגודל IV׳ו:

בבדיקה זו עבור כל הרצה נלקחו ממוצע כל ה sources ועבור כל יחס נוצרו 300 קבצים ומהם נלקח ממוצע גם כן וזהו הגרף .
ניתן לראות כי באזור יחס 5 גודל ה source הוא כגודל כל המשתנים בסט החוקים.



4. סקירת עבודות דומות בספרות והשוואה

קיימים מספר עבודות שיש להם רמת דמיון לפרויקט שלנו ביניהם

(ראו ביבליוגרפיה ערכים 43,44,8,7,45,24,14,9,3,25,4,1,22,18,34,39)

אלגוריתם "k-Edge connected component" [40]:

מטרתו היא מציאת k דרכים מקדקוד אחד לקדקוד שני ברכיב קשירות וכך נמצא את E'.

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.

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.

The k-Separator Problem:

בהינתן גרף בעל כיוון קדקוד משוקלל $G = (V, E, w)$ ומספר שלם חיובי k, הבעיה היא מציאת תת קבוצה של קודקודים בעלי משקל מינימלי, אשר הסרתם מובילה לגרף שבו הגודל של כל רכיב קשירות הוא קטן או שווה ל-k.

במאמר זה מראים שניתן למצוא פירוק בזמן פולינומיאלי לגרפים מסוימים, לשאר הגרפים ככל שאנו מחפשים k יותר מאוזן ככה הבעיה היא NP שלמה.

יש למאמר זה משמעות רבה עבור פרויקט זה כי מאמר [1] מבקש פירוק מאוזן מצד אחד, אך גם פירוק הגרף למלא רכיבי קשירות קטנים יעזור לפרויקט " חישוב מודלים מינימליים באמצעות גרף התלויות".

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, $\frac{2}{3} \leq \alpha < 1$. 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 $O(k\epsilon^{-1})$ כאשר K זה גודל הסאפארטור שאנו מבקשים, כלומר אם נוריד קצת מדרישת האיזון . הוכחה זאת יכולה להיות לנו יעילה.
החיסרון העיקרי (עבורנו) במאמר זה בזמן הריצה שתלוי בגודל האיזון שאנו מבקשים ולכן יכול להיות גדול מידי עבור מאמר שדורש לשפר זמני ריצה.

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 " חישוב מודלים מינימליים באמצעות גרף התלויות" 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.

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] *The k-Separator Problem:* Walid Ben-Ameur, Mohamed-Ahmed Mohamed-Sidi, and Jos´e Neto

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

[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.

[45] *Finding small balanced separators:* Uriel Feige Mohammad Mahdian

[46] https://github.com/mission-peace/interview

[47] *Graphs Excluding a Fixed Minor have Grids as Large as Treewidth, with Combinatorial and Algorithmic Applications through Bidimensionality:* Erik D. Demaine∗ MohammadTaghi Hajiaghayi∗

5. סיכום \ מסקנות

המסכנות מתהליך הבדיקות:

בפרויקט זה מומשו שתי אלגוריתמים למציאת V' מינימלי שנותן פירוק מאוזן.

1) **אלגוריתם "k-Edge connected component" [40]**:
במהלך הפרויקט נוסה קודם שימוש באלגוריתם [40] שמוצא E' (פירוק בעזרת קשתות).
עבור מציאת V' 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' .

במהלך תהליך הבדיקות על גרף מסוג Grid בגודל 4X4 וגם 5X5, ניתן היה לראות שהאלגוריתם נכשל בכל פעם מחדש.
התוצאה העיקרית שלו זה שהוא היה מוריד קדקוד אחד או שני קודקודים מהפינה של גרף ה Grid מה שמשאיר את רוב הגרף כרכיב קשירות שלא פורק בהצלחה.
אולם האלגוריתם מצא V' מינימלי אך הוא נתן פתרון מספק לדרישת האיזון של רכיב הקשירות לאחר הפירוק , דרישה שבאה עבור הפרויקט של " חישוב מודלים מינימליים באמצעות גרף התלויות" ממאמר [1]. לבסוף זנחנו את השימוש באלגוריתם הנ"ל [40].

2) **האלגוריתם של הפרויקט הנוכחי נוצר מהמאמר [45] (FM)**:

מסכנות מהפרויקט:

FM נותן תוצאות יפות מבחינת גודל ה-V' שמתקבל כאשר #AB מספיק גדול, מבחינת גודל רכיב הקשירות המקסימלי לאחר שהוצאו V' מהגרף. תוצאה זאת עומדת בקנה אחד לאלגוריתם [45] (FM) , ב-FM מספר הפעמים שנצטרך להגריל מספרים לקבוצה A ו B זה $2^{|W|}$ וכאשר נגריל מספר פעמים כנ"ל נקבל את התוצאה האידיאלית, כלומר V' מינימלי שמפרק בצורה מאוזנת.
חיסרון מרכזי שמשפיע על הפרויקט של " חישוב מודלים מינימליים באמצעות גרף התלויות" זה בעיית זמן ריצה.
זמן הריצה של האלגוריתם, כאשר #AB (כזכור שם מתקבלים ה-V' הקטנים ביותר) גדול במיוחד. החלק שהכי משפיע על זמן הריצה הארוך הוא האלגוריתם למציאת זרימה מקסימלית ומציאת החתך בגרף.

מסקנות מהמימוש:

1) האלגוריתם כולו נכתב ב-JAVA ועל סביבת עבודה Windows, אולי היה מקום לכתוב את האלגוריתם ב-Python ועל סביבת- Ubuntu ע"מ לקבל זמני ריצה מהירים יותר, אותם סביבות שעליהם התוכנות האחרות נכתבו עליהם ושקל יותר להשוות מולם.
2) בגלל שאנו מממשים אלגוריתם שלא מומש, יש הרבה ניסויים בתהליך כתיבת הקוד, קביעות שלקחנו ע"מ לנסות ולהוריד זמני ריצה, לדוגמה אנו החלטנו שנרוץ על |W| מ-4 פעמים ועד 32, דבר שלפי האלגוריתם במאמר נכון חלקית ובשלבי הניסויים ראינו שעבור |W|=32 האלגוריתם פשוט לא יעבוד כאשר אנו נרוץ על 1000 משתנים בקובץ ה-CNF ואנו נצטרך להקטין את הקבוע שבאמצעותו מגרילים משתנה לקבוצה W.

3) האלגוריתם שמבזבז זמן ריצה, כלומר אלגוריתם הזרימה אמור לרוץ ב- $O(V * E^2)$, מסקנה המימוש שנעשה בפרויקט זה לאלגוריתם הוא לא האידיאלי ביותר ויש מקום לשפרו.
4) מסקנה נוספת שנלקחה מהבדיקות עבור זמן הריצה הארוך במיוחד לאלגוריתם הזרימה היא שזמן הריצה הארוך נגרם עקב שימוש יתר בזיכרון.
5) שאר זמני הריצה (חוץ משל אלגוריתם הזרימה) תואמים לזמני הריצה ב [45].

<u>מסקנות עבור הסנכרון של שני הפרויקטים:</u>

בשלב זה של הפרויקט המשולב (הפרויקט הנ"ל והפרויקט של " חישוב מודלים מינימליים באמצעות גרף התלויות") הצלחנו לשלב את מבנה הנתונים של החוקים ומבנה הנתונים של הגרף במלואו האלגוריתם המשולב עובד ומוצא מודל מינימלי.
כאשר היחס בין המשתנים לחוקים הוא באזור 5-4 אז שם זמן הריצה של אלגוריתם למציאת מודל מינימלי הוא יותר גדול.

כאשר יוצרים קבצים רנדומליים עם סיכוי 50% למשתנה חיובי או שלילי אזי ברוב המקרים קיים source אחד שהוא ממש גדול וכמה sources בודדים . וכשיוצרים את אותם קבצים עם יחס 5-4 אז בממוצע גודל רכיב הקשירות הכי גדול הוא פשוט כל המשתנים באותו הקובץ.

פירוק רכיב הקשירות לא עוזר לצמצם את זמן הריצה אלא להפך וזאת מכיוון שעבור אלגוריתם DP מה שגורם לו לרוץ יותר לאט זה כמות הקריאות הרקורסיביות שתלויות ביחס בין המשתנים לחוקים ולא גודל ה source.
המסקנה המתבקשת היא שהאלגוריתם לפירוק רכיב הקשירות גוזל משאבים רבים וזאת מבלי שאפילו הציב משתנה בחוקים.

לאחר ניסויים רבים כנגד אלגוריתמים כמו WASP ו-DP ניתן לראות בבירור שהפתרון עם האלגוריתם לפירוק רכיב קשירות, כלומר האלגוריתם ל" חישוב מודלים מינימליים באמצעות גרף התלויות" עם השיפור של פירוק רכיב הקשירות הגדול ביותר, נותן תוצאה פחות טובה מאשר האלגוריתם שרץ ללא מציאת פירוק לרכיב קשירות.

ניתוח זמני ריצה

זמני ריצה כפי שמוזכרים ב-[45] :

זמני ריצה ממוצעים אשר הורצו 50 פעמים על אותו קובץ (האלגוריתם הוא רנדומלי ולכן ייתן כל פעם זמן ריצה שונה) וחושב להם ממוצע
בפרויקט זה אנו מיישמים אלגוריתם [45] שמציעה דרך למציאת V' מינימלי ע"י המרת הבעיה לחישוב החתך המינימלי. זמן הריצה שלו $n^{O(1)}2^{O(k\epsilon^{-2}\log(1/\epsilon))}$
כאשר k זה גודל הסאפארטור

$\epsilon > 0$ זה אחוז אשר מקיים $\frac{|v'|}{|v|} - \epsilon \leq \frac{|W \wedge V'|}{|W|} \leq \frac{|v'|}{|v|} + \epsilon$
כלומר יחס הקדקודים שב-W וגם הספארטור חלקי הקדקודים ב-W , צריך להיות שווה עד כדי ϵ ליחס הקדקודים של הסאפארטור חלקי כל הקדקודים.

$\alpha > \frac{2}{3}$ זה גודל האיזון שאנו מבקשים לפי [45].

כלומר, האלגוריתם מכוון לפירוק רכיב הקשירות ב- $\frac{2}{3}V$

מסקנה: האלגוריתם הוא מעריכי בגודל 'V שאנו מוצאים. במקרה שאין 'V קטן אז האלגוריתם ירוץ הרבה מאוד זמן.
יישום האלגוריתם מתחלק לכמה שלבים:

1) <u>הגרלת תת קבוצה W:</u>
בגלל שגודל I'VI אינו ידוע אנו נרוץ על W בגודל שונה בשביל שנוכל ככה להגדיל את גודל I'VI שיתקבל לאחר מכן. ככה גם בהתאמה יגדל זמן הריצה.
W מהווה תת קבוצה של V שגודלה יכול לנוע בין 4 קודקודים ל- 32 בלבד.
אנו נתחיל בבחירה של גודל מקסימלי IWI=4 ועד IWI=32 כאשר נכפיל את הגודל המקסימלי האפשרי ל-W ב-2.
אנו נעשה תמורה רנדומלית למערך W וניקח כל פעם ל-W את מספר הקדקודים שיכולים להיות W מקסימלי להיות W.

2) <u>הגרלת תתי קבוצות A,B:</u>
A,B תתי קבוצות בעלי קודקודים שונים של W. אנו נגריל קודקודים מ-W בעלי סיכוי שבין 0.25 ל- 0.75 להיכנס או לקוצה A או לקבוצה B. בנוסף נוסיף דרישה שלא יהיו קשתות בין הקדקודים בקבוצה A לקדקודים בקבוצה B.
את הגרלת A,B דורשים ב- [45] לבצע 2^W פעמים ע״מ שנקבל 'V אידיאלי, אצלנו באלגוריתם משיקולים של זמן ריצה נבצע בין 20-200 פעמים ונראה שככל שנגריל את A,B יותר פעמים ככה נקבל 'V קטן יותר ושנותן פירוק יותר מאוזן.

3) <u>יצירת גרף רשת זרימה:</u>
לפירוט איך נעשה ראה נספח ד'.
רשת זרימה זאת נוצרת בזמן ריצה של O(V+E) ומטרתה היא שכאשר נריץ אלגוריתם זרימה בהמשך נוכל בעזרת הרשת להצביע על הקדקודים שיהיו 'V המינימלי ומציאת קודקודים שאותם נרצה למצוא זרימה מהם.

4) <u>הרצת אלגוריתם למציאת זרימה מקסימלית וחיתוך מינימלי:</u>
השתמשתי באלגוריתם מ-[46] עם מספר רב של שינויים
Ford Fulkerson method Edmonds Karp algorithm למציאת זרימה מקסימלית וחיתוך מינימלי זמן הריצה שלו $O(VE^2)$.
התוצאה מהרצת האלגוריתם זה שהתקבלו שתי קבוצות T,S.
כל הקדקודים שבצד s של החיתוך שייכים ל-S והשאר שייכים ל-T.
אנו נוסיף דרישה ש- ITI>0.1 וגם ISI>0.1, דרישה זאת הינה הכרחית מכיוון שאם גדלי ISI ו ITI יחסית מאוזנים כך גם גודל 'V שיתקבל מהם יגרום לפירוק מאוזן של רכיב הקשירות כלומר G.

5) <u>קבלת 'V כאשר יש לנו שתי קבוצות T,S:</u>
כאשר יש לנו 2 קבוצות שמייצגות חיתוך ה-'V יתקבל מהקדקודים שעל הקשתות שבין S ל- T.

6) <u>שמירת כל 'V שהתקבלו ומציאת המינימום:</u>
כחלק מהדרישה של [1] אנו נרצה להציב את המשתנים ש-'V מייצג אותם פעולה שכבר אמרנו שתהיה $O(2^{|v'|})$, לכן נרצה למצוא את 'V המינימלי ע״מ לשמור על זמני הריצה קטנים ככל האפשר. נמצא 'V מינימלי $O(n)$.
בבדיקות נראה ש-'V מינימלי לא בהכרח ייתן פירוק מאוזן. רק בבדיקות נפעיל אלגוריתם מיון שעובד על זמן ריצה מקסימלי $O(n^2)$ ע״מ שאולי בעתיד נרצה להשתמש לאו דווקא ב-'V הכי קטן אלה בזה שייתן פירוק הכי טוב.

6. מערכת לניהול 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	https://drive.google.com/open?id=1RQO3z8ZgcdkRnnPwGLwMtpSEwTGBeQXi

7. נספחים

א. 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.
14.6	**שלב המסירה** – מסירת הפרויקט.

ב. 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	המימוש של אלגוריתם [45] למציאת פירוק טוב לא תעבוד בזמן ריצה שמתאים לאלגוריתם [1]	8	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 .

ג. רשימת\טבלת דרישות

מסי 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 head→body.
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	מימוש תיאוריה [1] שמוצאת V' מינימלי ושמפרק בצורה מאוזנת
14	פונקציונלי	להריץ אלגוריתם זרימה על s,t ע"מ למצוא חיתך מינימלי בינהם
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'.

ד. יצירת גרף רשת זרימה

בהינתן שתי תתי קבוצות A,B אנו ניצור גרף באופן הבא :

4) ניצור באופן מלאכותי 2 קודקודי עזר s,t (ראה מילון מונחים) קודקודים אלו נוצרים כי בהמשך נריץ אלגוריתם למציאת זרימה מקסימלית וחיתוך מינימלי על הגרף החדש שנוצר.
5) נקבע משקלים לקשתות :
   a. משקל קשת שבין קודקוד s לכל אחד מהקודקודים ב-A יהיה |V|.
   b. משקל קשת שבין הקודקודים ב-B לקודקוד t יהיה |V|.
6) עבור כל קודקוד אשר **לא** נמצא ב-A,B אנו נבצע :
   a. עבור כל קודקוד x ניצור קודקוד x1 , x2 בגרף החדש אשר כל הקשתות שנכנסות ל-x יצביעו מעכשיו ל-x1 משקל כל קשת הוא |V| .
   b. כל הקשתות אשר יוצאות מ-x מעתה יצאו מ- x2 משקל כל קשת הוא |V|.
   c. ניצור קשת עם משקל 1 בין x1 ל- x2 .

ה. $\alpha > \frac{2}{3}$ זה גודל האיזון שאנו מבקשים לפי [45].

ו. כלומר, האלגוריתם מכוון לפירוק רכיב הקשירות ב- $\frac{2}{3}V$

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

בפרויקט זה אנו **יישמנו גם** 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.

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′).

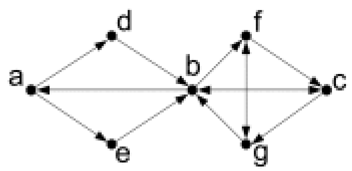
מכיוון שמת החיבור של s ו- t, בגרף מכוון היא המינימום של max-flow (זרימה מקסימלית) בין s ל t ובין t ל- s, הזרימה המינימלית משני אלה, תהיה לדוגמה x, נקצה קשת (s, t) כמו שידמה לנו קישוריות בין s ו t. אנחנו גם נגדיר (S, T) כדי המקביל min-cut.

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.

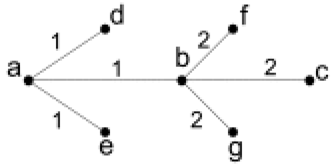
אנו יוצרים עץ A שבעצם מייצג את כמות הדרכים שניתן להגיע מכל קדקוד לשני ברכיב הקשירות.

ציור זה מייצג רכיב קשירות:

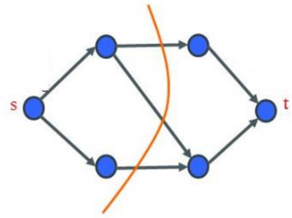


ציור זה מייצג את העץ שנבנה A. לאחר הפעלת אלגוריתם [40]. בציור זה ניתן לראות את מספר הדרכים מכל קדקוד לכל קדקוד, אשר תמיד נבחר את המספר המינימלי.

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.

הסבר על אלגוריתם [40] ראה נספח ה'.

<u>שלב הבא הוא 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).

<u>שלב הבא הוא 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 :

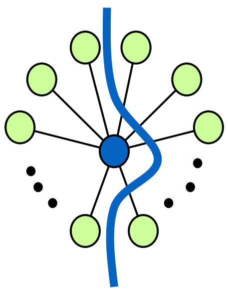
- לאחר יצירת גרף A מגרף רכיב הקשירות המקורי, ניצור גרף שני נקרא לו גרף "מיוחד", הגרף ייווצר באופן הבא :

  - 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.

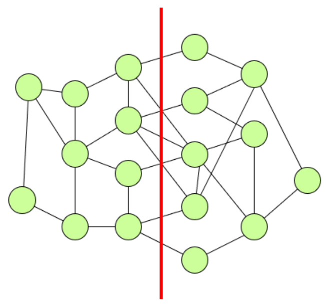
  כעת, כאשר נבקש למצוא קודקודים להוריד, ויש לנו a ו b קודקודים שעבורם הקשת היא בעלת המשקל המינימלי בגרף A, אנו נריץ את האלגוריתם למציאת זרימה מקסימלית וחיתוך מינימלי בגרף "המיוחד", כלומר נמצא את כל הקשתות בחתך בין a ל b ונניח ש x נמצא בחתך, אנו נעדיף להסיר קדקוד x אם החתך מחזיק את x ו x0 .
  בכל מקרה אנו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



.1



.2

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

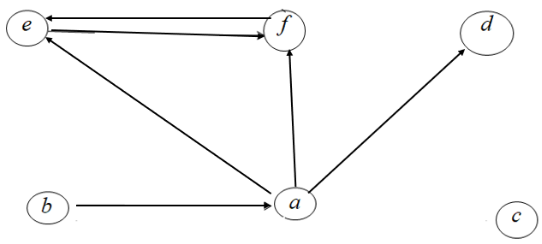
ו. יצירת גרף מסט חוקים נתון

סט החוקים מהצורה $a_1 \wedge \ldots \wedge a_n \to b_1 \vee \ldots \vee b_m$ (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.

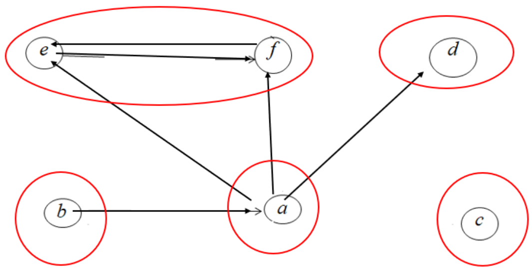
עבור סט חוקים :

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

ניצור את הגרף:



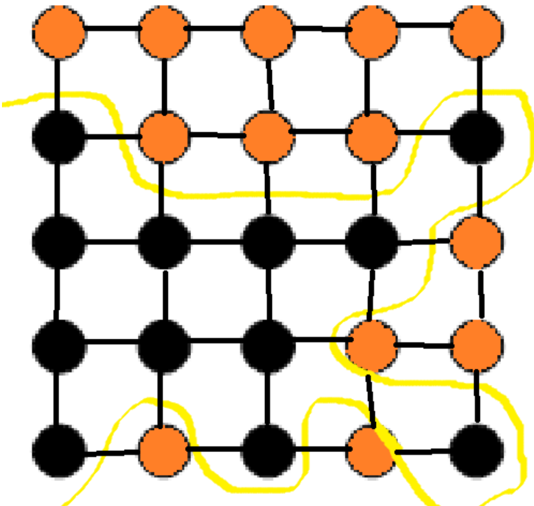
נמצא רכיבי קשירות בגרף:

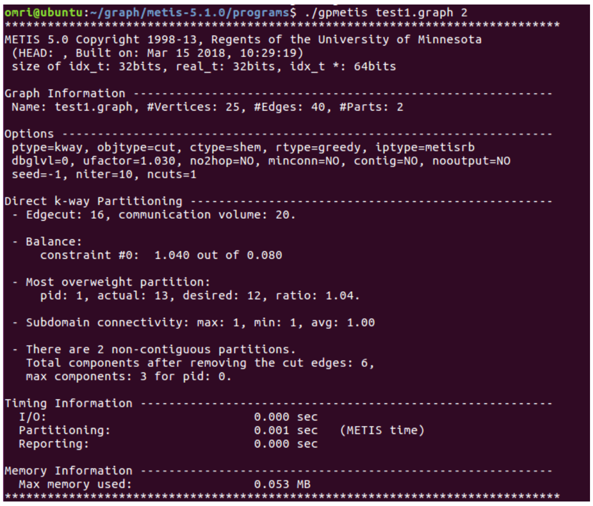


רכיב הקשירות הגדול ביותר זה {f,e} וכאשר ה source יכול להיות b או c .

ז. השוואת Metis לאלגוריתם שלנו:

דוגמא עבור גרף Grid :





```
omri@ubuntu:~/graph/metis-5.1.0/programs$ ./gpmetis test1.graph 2
******************************************************************************
METIS 5.0 Copyright 1998-13, Regents of the University of Minnesota
 (HEAD: , Built on: Mar 15 2018, 10:29:19)
 size of idx_t: 32bits, real_t: 32bits, idx_t *: 64bits

Graph Information ---------------------------------------------------------
 Name: test1.graph, #Vertices: 25, #Edges: 40, #Parts: 2

Options -------------------------------------------------------------------
 ptype=kway, objtype=cut, ctype=shem, rtype=greedy, iptype=metisrb
 dbglvl=0, ufactor=1.030, no2hop=NO, minconn=NO, contig=NO, nooutput=NO
 seed=-1, niter=10, ncuts=1

Direct k-way Partitioning -------------------------------------------------
 - Edgecut: 16, communication volume: 20.

 - Balance:
     constraint #0:  1.040 out of 0.080

 - Most overweight partition:
     pid: 1, actual: 13, desired: 12, ratio: 1.04.

 - Subdomain connectivity: max: 1, min: 1, avg: 1.00

 - There are 2 non-contiguous partitions.
   Total components after removing the cut edges: 6,
   max components: 3 for pid: 0.

Timing Information --------------------------------------------------------
  I/O:                                0.000 sec
  Partitioning:                       0.001 sec   (METIS time)
  Reporting:                          0.000 sec

Memory Information --------------------------------------------------------
  Max memory used:                    0.053 MB
******************************************************************************
```

נבחר גרף בצורת Grid בגודל 5x5 ונבקש לפרק ל-2 קבוצות שונות.

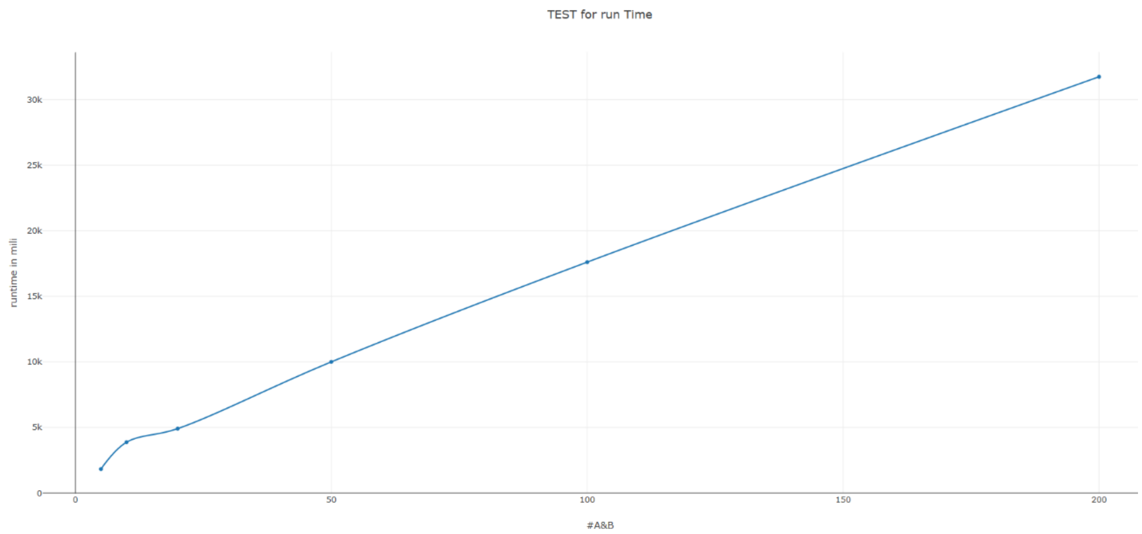
נסמן בצבעים שונים כל קבוצה שונה לאחר הפירוק.

ניתן לראות ש Metis מוצא פירוק מאוזן (לגמרי) לגרף בזמן קצר בצורה משמעותית מהאלגוריתם שלי ובשימוש בפחות זיכרון.

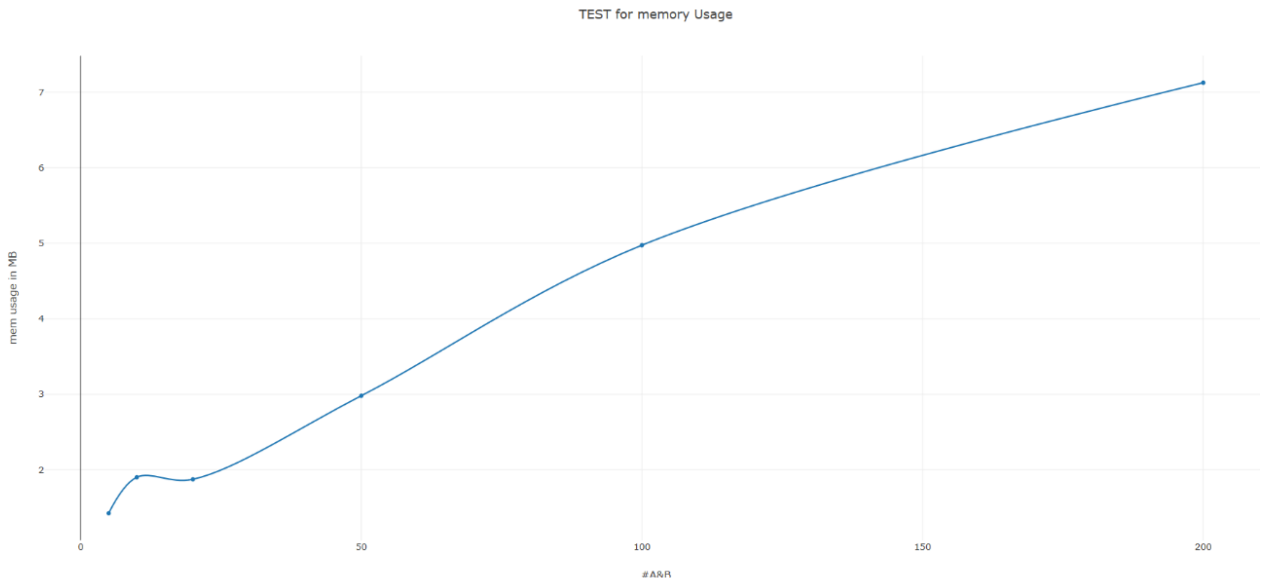
ניתן היה באותו אופן לבקש לפרק ל-|V| קבוצות ואז כל קודקוד היה קבוצה בפני עצמו.

תרשימים

Run time Test



בדיקה לשימוש בזיכרון



בדיקה לשימוש בזיכרון, זמן ריצה וגודל 'V מינימלי

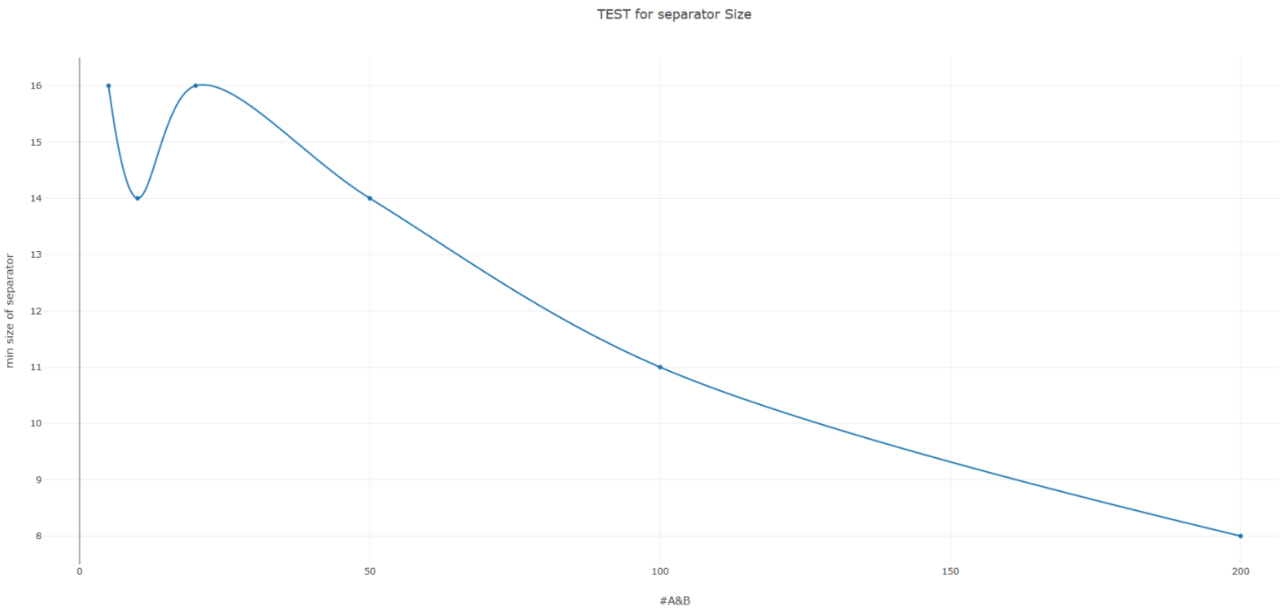
#(A B)	Memory Usage In MB	Run Time In millisec	Min separator size
5	1.4254531860351562 MB	1827 millisec	16
10	1.9015274047851562 MB	3875 millisec	14
20	1.874786376953125 MB	4910 millisec	16
50	2.9805374145507812 MB	10002 millisec	14
100	4.9737091064453125 MB	17610 millisec	11
200	7.124916076660156 MB	31747 millisec	8

חישוב זמן ריצה ופירוק האלגוריתם לחלקים

Table test for Run Time for every #AB

#	#AB run	Runtime for lottery w	Runtime for lottery A an B	Run Time to create flow chart	Runtime for Ford Fulkerson	Runtime for contains	Total runtime
1	5	62 millisec	1 millisec	0 millisec	1764 millisec	0 millisec	1827 millisec
2	10	94 millisec	0 millisec	0 millisec	3766 millisec	0 millisec	3860 millisec
3	20	63 millisec	0 millisec	0 millisec	4814 millisec	15 millisec	4892 millisec
4	50	29 millisec	0 millisec	0 millisec	9642 millisec	16 millisec	9687 millisec
5	100	78 millisec	0 millisec	0 millisec	17579 millisec	15 millisec	17672 millisec
6	200	31 millisec	25 millisec	0 millisec	31154 millisec	25 millisec	31235 millisec

גרף של תלות מספר הרצות AB# לגודל ה-'V המינימלי שיתקבל



בדיקות עבור יעילות הפירוק:

עבור כל AB# נחשב את 5 ה'V המינימלי ונמצא כמה מה גודל רכיב הקשירות הגדול ביותר לאחר הפירוק.

Table test for dismantle efficiency when #AB=5

#	seperator size	Percentage of dismantle	largest CC size	All connected component after_dismantle
1	16	71.0 %	29	29 28 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
2	18	79.0 %	21	21 4 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
3	20	76.0 %	24	24 22 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
4	21	59.0 %	41	41 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
5	21	52.0 %	48	48 5 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

Table test for dismantle efficiency when #AB=10

#	seperator size	Percentage of dismantle	largest CC size	All connected component after_dismantle
1	14	41.0 %	59	59 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
2	17	51.0 %	49	49 12 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
3	19	60.0 %	40	40 5 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
4	20	73.0 %	27	27 3 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
5	20	50.0 %	50	50 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

Table test for dismantle efficiency when #AB=20

#	seperator size	Percentage of dismantle	largest CC size	All connected component after_dismantle
1	16	30.0 %	70	70 1 1 1 1 1 1 1 1 1 1 1 1 1 1
2	16	43.00000000000001 %	57	57 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
3	16	35.0 %	65	65 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
4	18	52.0 %	48	48 6 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
5	20	48.0 %	52	52 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

Table test for dismantle efficiency when #AB=50

#	seperator size	Percentage of dismantle	largest CC size	All connected component after_dismantle
1	14	43.00000000000001 %	57	57 14 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
2	15	41.0 %	59	59 8 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
3	15	36.0 %	64	64 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
4	16	42.00000000000001 %	58	58 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
5	16	40.0 %	60	60 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

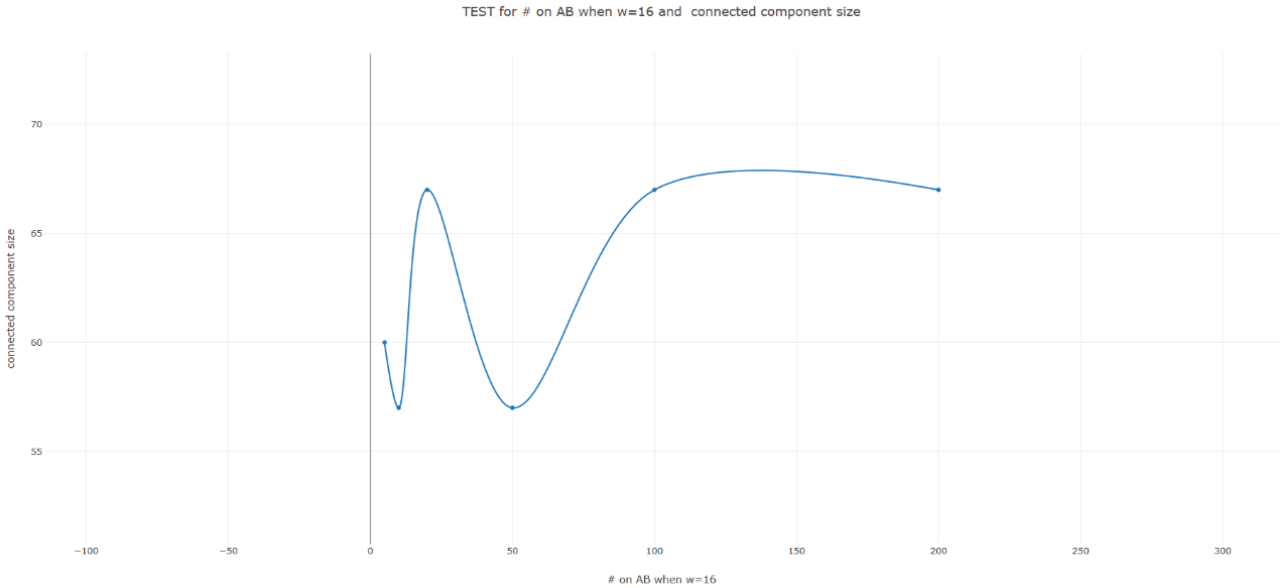
Table test for dismantle efficiency when #AB=100

#	seperator size	Percentage of dismantle	largest CC size	All connected component after_dismantle
1	11	38.0 %	62	62 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
2	13	33.0 %	67	67 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
3	14	35.0 %	65	65 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
4	14	32.0 %	68	68 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
5	14	43.00000000000001 %	57	57 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

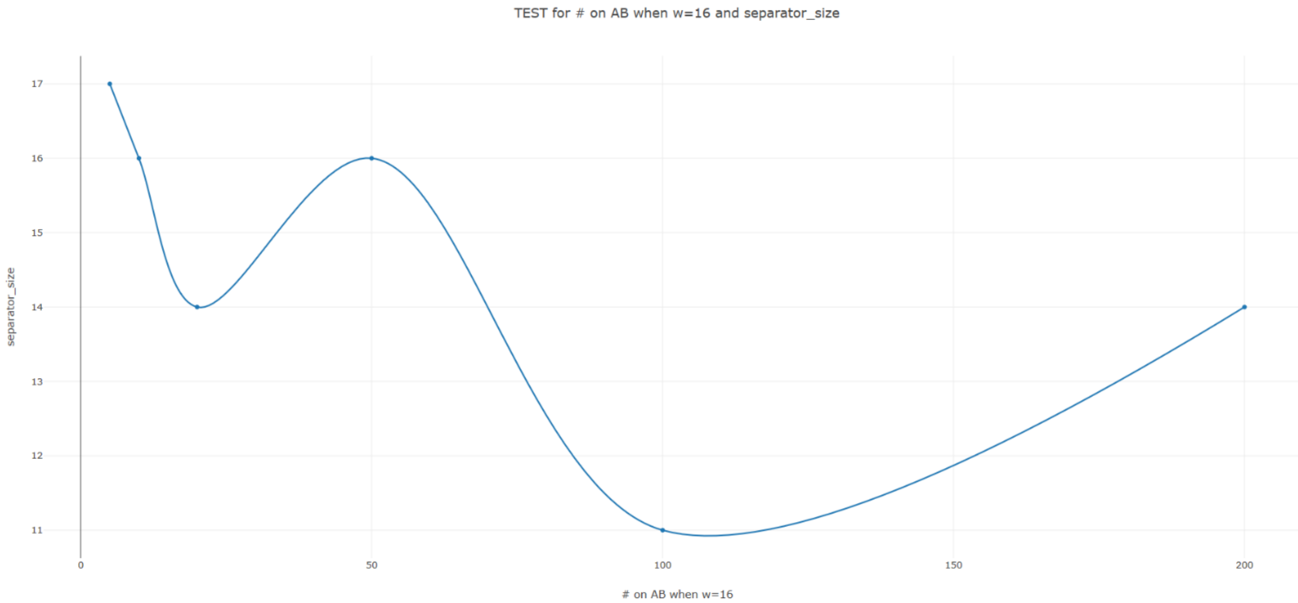
Table test for dismantle efficiency when #AB=200

#	seperator size	Percentage of dismantle	largest CC size	All connected component after_dismantle
1	8	28.0 %	72	72 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
2	9	32.0 %	68	68 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
3	10	34.0 %	66	66 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
4	10	27.0 %	73	73 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
5	11	36.0 %	64	64 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

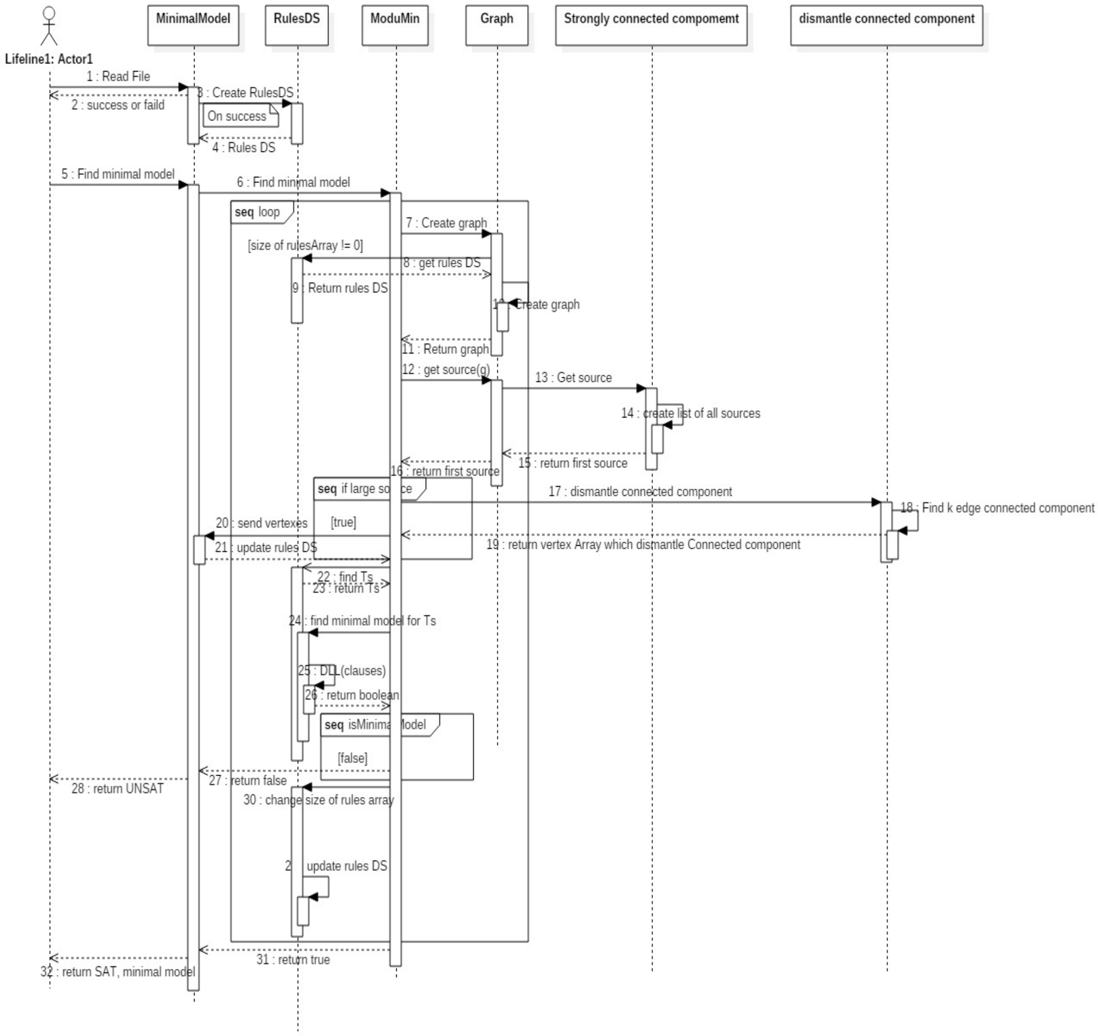
בדיקת יעילות הפירוק כאשר |W|=16 קבוע



בדיקה מציאת |V'| כאשר #AB משתנה ו- |W|=16 קבוע



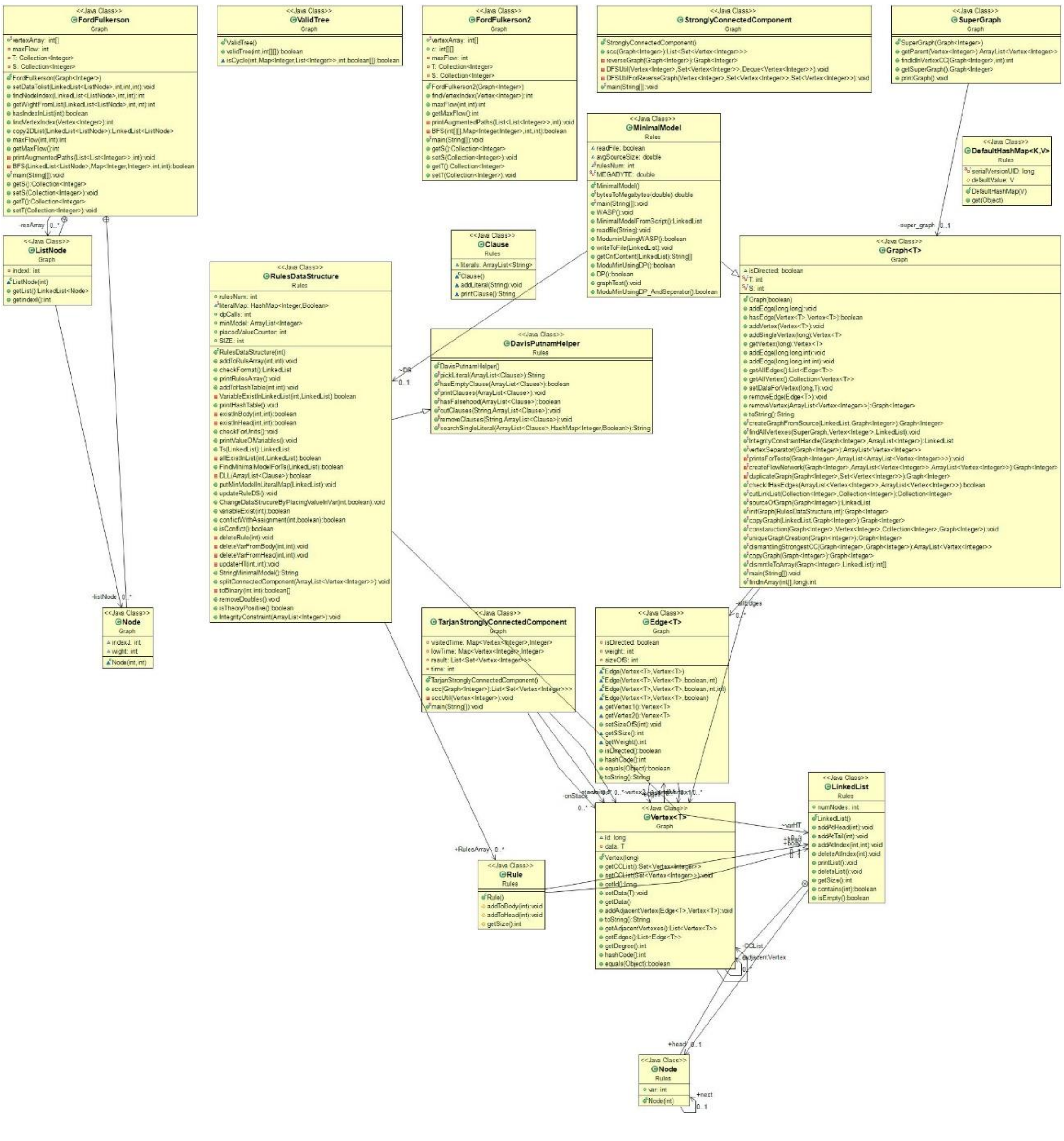
Sequence diagram



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 .

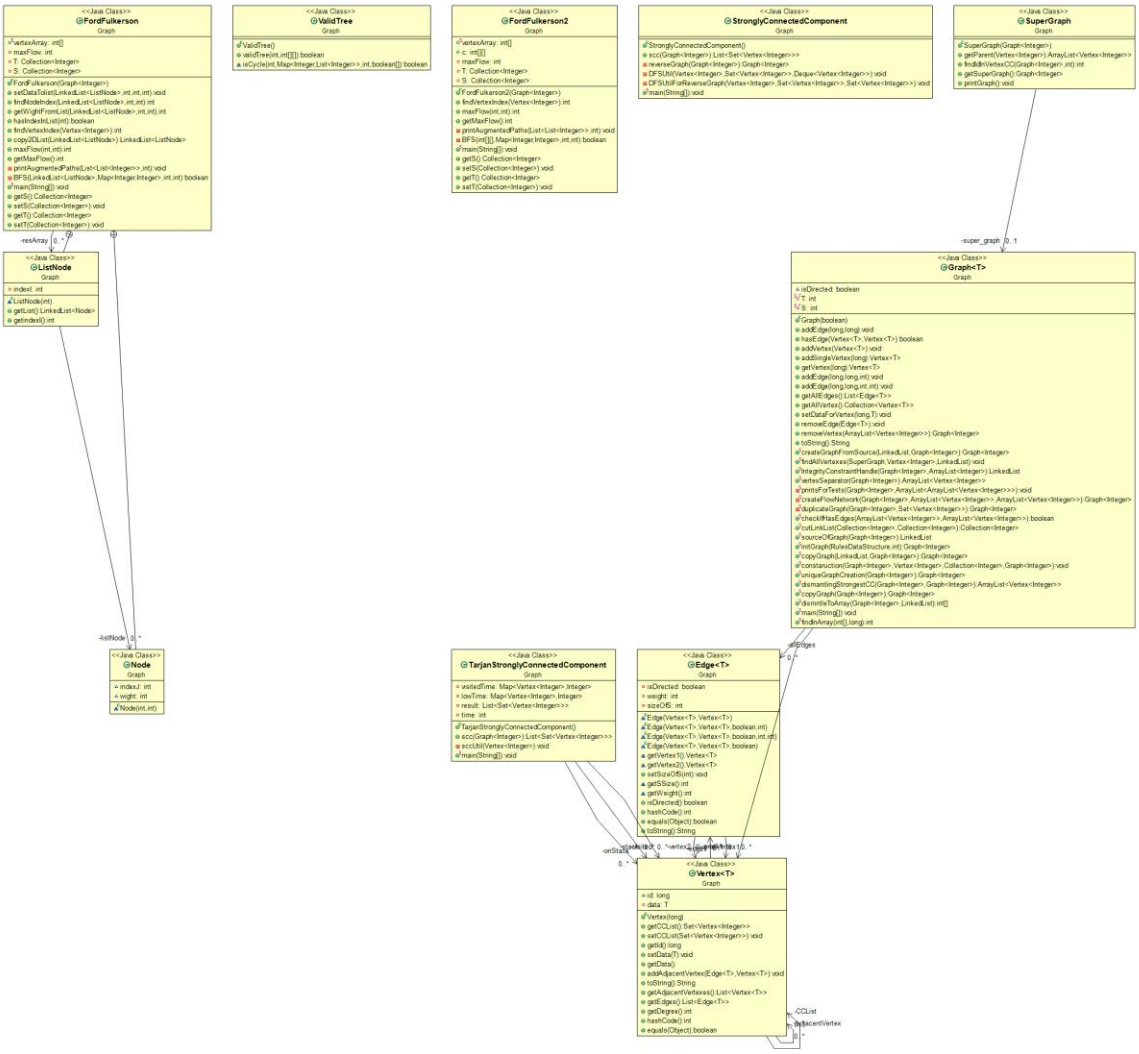
Class diagram

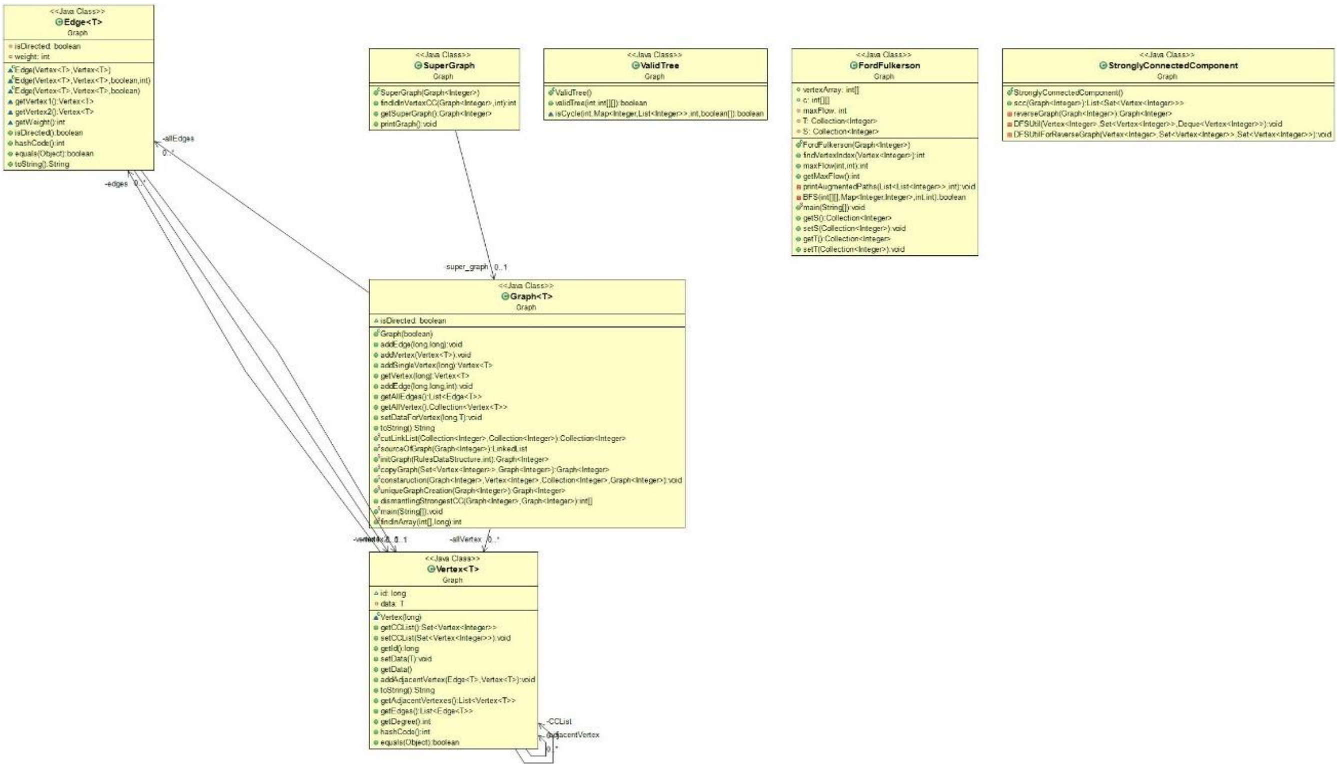


התרשים הבה מייצג את יחסי המחלקות שידועות עבור מחלקת החוקים ומחלקת הגרף.

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.

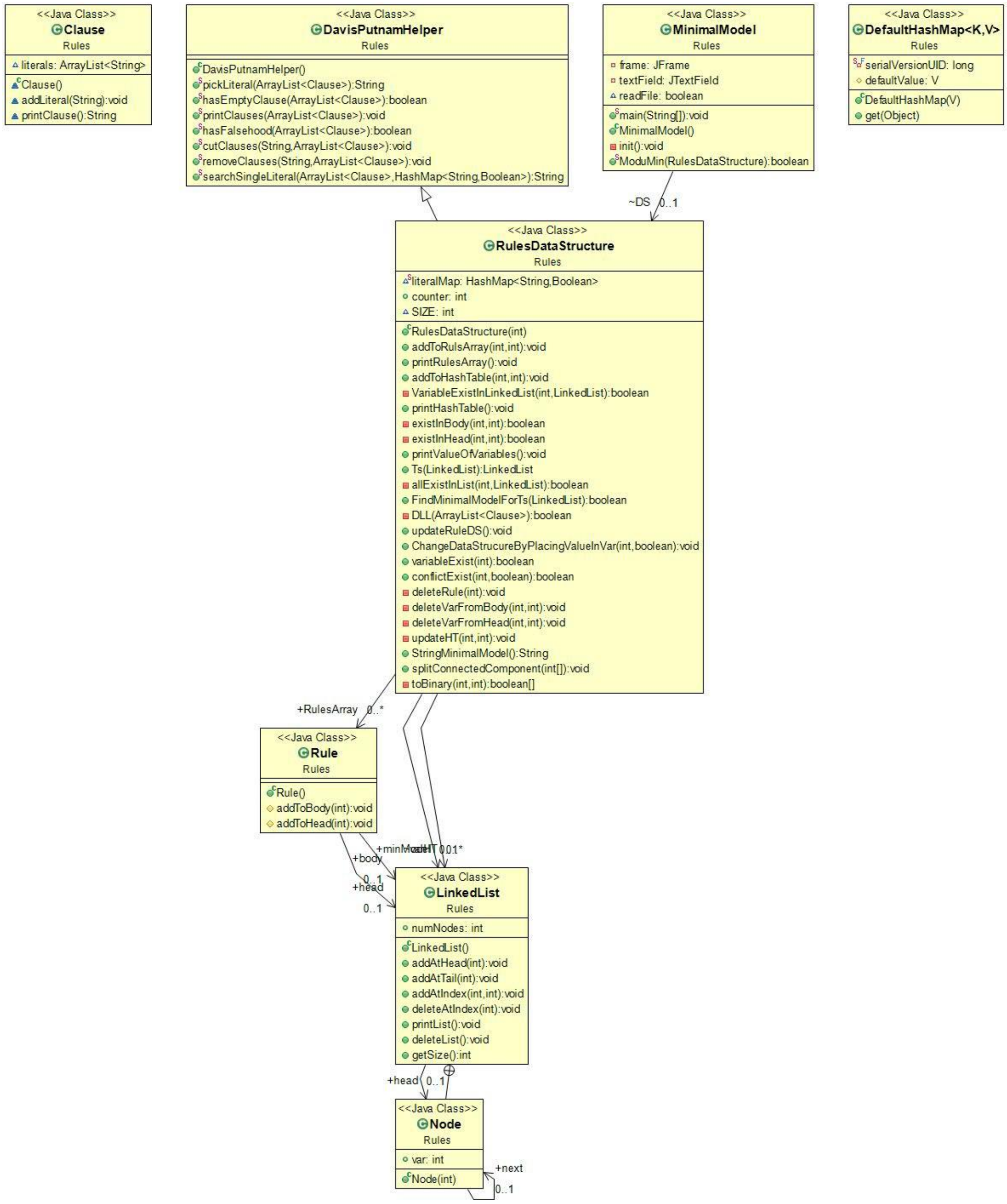




התרשים הבה מייצג את יחסי המחלקות שידועות עבור מחלקת הגרף.

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.

Azrieli
College of Engineering
Jerusalem

עזריאלי
מכללה אקדמית להנדסה
ירושלים

Software Engineering Department

Final project

By: Omri Mizrahi

Minimal balanced node separator

Supervisor: Dr. Yehuda Hassin

Approved by the Supervisor: Dr. Yehuda Hassin **Date:**

Approved by the Projects' Coordinator: Dr. Assaf B. Spanier **Date:**

July 2018

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

Summary

As part of the final project in the Department of Software Engineering at Azrieli collage of engineering (JCE), I was assigned the task of implementing a project within a of 400 hours.
The project I will be doing is a research project that was carried out in cooperation with the student Adi Tyree and our mentors Dr. Yehuda Hassin and Prof. Rachel Ben Eliyahu Zohari.

In Adi's project we will deal with part of the Rules data structure . In this project we will deal with the graphical part of the algorithm, The data of Adi Tyree's project.

The problem given the SAT positive formula is to find a minimum model for the formula.
Recently [1] the output algorithm has been proposed a minimal model that depends on the size of the largest binding component. Each formula can be translated into a graph.
We will implement the algorithm as suggested in the article [1]. The goal of the algorithm [1] is to find a minimum model for a set of CNF-shaped rules while improving the runtime of the algorithm. Finding the minimum model will be done by constructing a graph for the set of rules in a given order, and the run time of the algorithm depends on the size of the graph's bonding components, so if we can reduce the size of the binding elements in the graph, we can reduce the running time of [1].

To decompose the binding element, we will use the theory from article [45]. This article allows us to find a minimum number of vertices that should be lowered so that the binding element, the size of an algorithm [1] depends on it.

It turns out that in many difficult problems, the whole graph is one large binding component. We will propose to solve the algorithm [40] with the goal of finding arcs that will decompose the graph, and we will use the information that this algorithm gives to find vertices if we remove them from the graph we will be able to dismantle the large binding component most in the graph, thus reducing the running time of the algorithm [1].

Management of the two projects requires a lot of organization and responsibility, so we need to have two key points to deal with, one is synchronization between the various projects, so that we can achieve the two algorithms will work together weekly work obligation and the creation of a management system of the two projects together, the second part is working closely with the mentors because the project in part is the realization of a theory and based on similar studies done.