האוניברסיטה הפתוחה
המחלקה למתמטיקה ולמדעי המחשב
החטיבה למדעי המחשב

פרויקט מתקדם במדעי המחשב

מימוש אלגוריתמים לשמירת הפרטיות בגרפים ורשתות חברתיות

מנחה: פרופ' אהוד גודס

סמסטר ב' 3127

מגיש:
גלעד קינן- ת"ז: 6:7:1498

פרויקט מתקדם זה הוגש כחלק מהדרישות לקבלת תואר #מוסמך .M.Sc(במדעי המחשב#*
באוניברסיטה הפתוחה

תוכן עניינים

1. מבוא 4
הגישה הנאיבית 4
גישת הקיבוץ 4
גישות המאפשרות שינויים בגרף 5
מטרת הפרויקט 6
היקף הפרויקט 6
2. תיאור כללי 7
התמקדות במוצר 7
סביבת המערכת 7
3. עיצוב המערכת 10
ארכיטקטורה 10
ממשק משתמש 11
טכנולוגיות ותלויות 14
בדיקות המערכת 14
4. הרצת התוכנית 16
פקודות להרצת המערכת 16
דוגמאות הרצה 16
יעילות האלגוריתמים על ידי זיהוי צמתים דומים עם אותה דרגה 16
יעילות האלגוריתמים על פי מודל אנטרופיה 18
5. חשיבות הפרויקט 20
6. רשימת מקורות 21
7. נספחים 22
דיאגרמת מחלקות 22
תיאור תהליכים מרכזיים 27
דוגמת הרצת K-Degree במעקב אחר צומת 2630 לחישוב Entropy 33
דוגמת הרצת K-Symmetry במעקב אחר צומת 2630 לחישוב Degree Similarity 37
דוגמת הרצת K-Symmetry במעקב אחר צומת 2630 לחישוב Entropy 38

1. מבוא

רשתות חברתיות הן חלק מרכזי בחיינו ואפשר למצוא אותן במגוון רחב של תחומים כגון רשת ארגונית בעבודה או רשת חברתית ליצירת קשרים בין אישיים, רשת חברתית בד"כ מכילה מידע אישי על אינדיבידואלים ועל הקשרים של האינדיבידואל עם אינדיבידואלים אחרים או עם קבוצות של אינדיבידואלים, משתמשים ברשת חברתית מעלים תוכן באופן רצוני אך אינם מודעים מי בעלי הגישה למידע זה ואיזה שימוש יעשה במידע הפרטי על ידי בעלי המידע, המחקר בנושא בעיות בפרטיות ברשתות חברתיות הוא צעיר יחסית והוא מתמקד בעיקר בשני מקרים עיקריים של בעיות אבטחה [12].

א. **פרצות בפרסום מידע** – במקרה זה התוקף מעוניין להשיג מידע פרטי על אינדיבידואלים מתוך רשתות חברתיות שמפורסמות בצורה פומבית. פרסום המידע ברשת חברתית מיועד לצורך כריית מידע אך תיתכן פגיעה בפרטיות כתוצאה מפרסום המידע.

ב. **פרצות בשיתוף מידע** – בעל המידע מעוניין לשתף מידע של רשת חברתית לצורך מחקר אבל מעוניין לשמור על פרטיות המידע של האינדיבידואלים ברשת. במקרה זה, בעל המידע צריך לספק מנגנון אבטחה בכדי לשמור על פרטיות המידע של האינדיבידואלים.
כאשר בעל הרשת חברתית מעוניין לשתף או לפרסם את הרשת בצורה פומבית הסיכון שמידע פרטי יזלוג יעלה באופן ניכר.

בעבודה זו התמקדתי בפרצה מהסוג השני –"פרצות בשיתוף מידע" שבו בעל המידע מעוניין לשמור על פרטיות המידע של האינדיבידואלים ברשת, מידע פרטי שנמצא ברשת חברתית שונה לעומת מידע פרטי בטבלה רלציונית (עם מאפיינים על משתמש בשורה), מכיוון שברשת חברתית ישנו מידע עשיר יותר לגבי האינדיבידואל כגון קשרים בין אינדיבידואלים.
כאשר בעל רשת חברתית מפרסם מידע יש צורך בבחירת שיטה מתאימה לביצוע אנונימיות למידע אשר תיקח בחשבון את מאפייני הרשת מבחינת זיהוי מידע רגיש שיש ברשת, זיהוי איזה מידע מקדים יש לתוקף ומידת השימושיות של המידע כפי שהוזכר בסעיף הקודם.
השיטות העיקריות לביצוע אנונימיות לטבלה רלציונית הם הכללה (Generalization) ופיזור/הסטה (Perturbation).
בשיטת ההכללה מסירים ספציפיות מהמידע ומקבצים מידע לקבוצה כאשר כל הצמתים בקבוצה הם בעלי אותו מידע מוכלל.
בשיטת הפיזור/הסטה מבצעים שינוי במידע (על ידי טכניקות שונות) בכדי לקבל התפלגות אחידה מהמידע.
בדומה לשיטות אבטחה על טבלאות רלציוניות, גם בשיטות אבטחה ברשת חברתיות זהות במידה ניכרת, נסקור בקצרה כמה מן הקטגוריות העיקריות:

הגישה הנאיבית

הגישה הנאיבית לשמירה על פרטיות המידע היא על ידי פרסום גרסה של הרשת החברתית לאחר הסרה של מזהים מייחדים או החלפתם ב- Ids שונים.
גישה זו אינה מבטיחה שמירה על פרטיות המידע מכיוון שתוקף יכול לזהות אינדיבידואלים מתוך המידע במידה ויש לו רקע מקדים של מספר מזהים לא מייחדים כמו למשל תאריך לידה, מיקוד וסטטוס משפחתי ובעזרתם יכול להצליב טבלה אנונימית עם טבלאות לא אנונימיות אחרות ובכך למצוא התאמה של אחד לאחד לשורה מתאימה בטבלה האנונימית. [14]
לכן, גישה זו אינה מספקת בכדי לשמור על פרטיות המידע ויש צורך בביצוע גישה נוספת בנוסף לגישה הנאיבית.

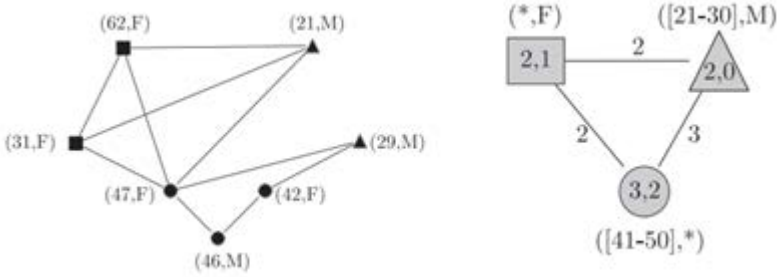
גישת הקיבוץ

בגישה זו מבצעים קיבוץ של צמתים וקשרים לקבוצות (Partitions) ומבצעים הכללה מסוגים שונים את הקיבוץ:

א. קיבוץ צמתים לצומת אחת על ידי הכללת מאפייני הצמתים בכדי להסתיר מידע של צומת כלשהי.
ב. קיבוץ קשרים על ידי הסרתם מהגרף המקורי ויצירת קשרים בין קבוצות (Partitions) בכדי להסתיר מידע של קשרים, על הקשר שנוצר מאפיינים על ידי מספר הקשתות שהוכללו.

דוגמא של גרף מקורי ואותו הגרף לאחר ביצוע אלגוריתם לפי גישת הקיבוץ, ראה איור 1.

איור 1 – גרף מקורי וגרף לאחר ביצוע Clustering



באיור 1 ניתן לראות שני גרפים, בצד שמאל הגרף המקורי כאשר צומת מסומנת על ידי ריבוע, משולש או עיגול מלא, קשר בין צמתים מסומן ע״י קו. לאחר ביצוע תהליך של קיבוץ מקבלים את הגרף הימני שבו מקבצים את כל הצמתים מסוג ריבוע לצומת אחת (נראית כריבוע), באותו האופן מקבצים את כל הצמתים בצורת משולש או עיגול לפי הסוג צומת. בתוך הצומת המאוחדת מסמנים את מספר הצמתים שקובצו ואת מספר הקשרים שקובצו, כך למשך בצומת ריבוע המאוחדת קובצו שני צמתים והיה קשר ביניהם שקובץ. באותו האופן, היה בגרף המקורי שני צמתים בצורת משולש ולא היה ביניהם קשר לכן בצומת משולש המאוחד יהיה רשום 2,0. במידה ויש קשר בין שני צמתים מסוג שונה אז מסמנים זאת בקשר בין הצמתים המאוחדים, לדוגמא ישנם שני קשרים שונים בין צמתי ריבוע לצמתי עיגול, לכן בקשר בין צומת ריבוע המאוחד לצומת עיגול המאוחד יהיה קשר שבו רשום 2.

חסרונות של גישת הקיבוץ היא שהגרף שמתקבל הוא יותר קטן משמעותית מהגרף המקורי מכיוון שמאחדים צמתים לצומת אחת ומסירים קשרים בתוך הקבוצה, כמו כן מאחדים קשרים בין קבוצות לקשר אחד. לכן, שינוי זה מסתיר מידע על מבניות הגרף בתוך הקבוצות של הגרף המקורי מה שפוגע בשימושיות הגרף בצורה ניכרת.

גישות המאפשרות שינויים בגרף

בכדי לשמר את השימושיות בגרף המתקבל מנסים בגישה זו לבצע שינויים קלים בגרף המקורי בכדי לקבל גרף חדש אנונימי.
שינויים אלו יכולים להיות הוספת צומת, הסרת צומת, הוספת קשר, הסרת קשר.

הסוגים השונים בגישה זו:

א. K-Anonymity – ביצוע שינויים של הוספה/הסרה של צמתים או קשרים בצורה אופטימלית בכדי לקבל קבוצות בגודל K של קבוצת המאפיינים אותו מנסים להכליל.
בכך לתוקף אין דרך לזהות צומת מסוימת מבין k-1 צמתים אחרים בגרף שמתקבל.
בעבודה זו אסקור כמה מאמרים המסבירים שיטות בגישה זו. [11][9][8][7][6][5]

ב. Randomized – ביצוע שינויים רנדומליים על ידי פיזור/הסטה (Perturbation) בדומה לגישות דומות שנעשות על טבלה רלציונית. [13]

- הוספה/הסרה רנדומלית – מוסיפים x צמתים (לא בהכרח k) ״שקריים״ שלא היו קיימים בגרף המקורי ולאחר מכן מסירים x צמתים ״אמתיים״ שהיו קיימים בגרף המקורי. מספר הצמתים הכולל בגרף נשמר.
- החלפה רנדומלית – בוחרים בצורה רנדומלית x זוגות של צמתים שעומדים בתנאי:
For each (t, w), (u, v): (t, v) and (u, w) does not exist in original graph).
ומחברים את הזוגות בצורה הבאה: (t,v), (u,w), אסטרטגיה זו שומרת על הדרגה של כל צומת.

שינויים רנדומליים אינם מבטיחים שמירת פרטיות בצורה מלאה של K-Anonymity אלא מבטיחים זאת בצורה הסתברותית מכיוון שבשיטה הרנדומלית מבצעים שינויים על הוספה של ״רעש״ כלשהו ולא בהכרח ליצירת קבוצות בגודל k זהה, אלא בגודל שונה שמשתנה בין הרצות [13].

מטרת הפרויקט

מטרת הפרויקט המסכם הוא לפתח תוכנית אשר תממש שניים מהאלגוריתם שסוקרו בעבודה המסכמת. בעבודה המסכמת הוצגו מספר קטגוריות של אלגוריתמים לביצוע אנונימיזציה. אנו נתרכז באלגוריתמים מהקטגוריה הראשונה. מתוך אותם אלגוריתמים פיתחתי שניים מהם שארחיב עליהם בהמשך העבודה.

היקף הפרויקט

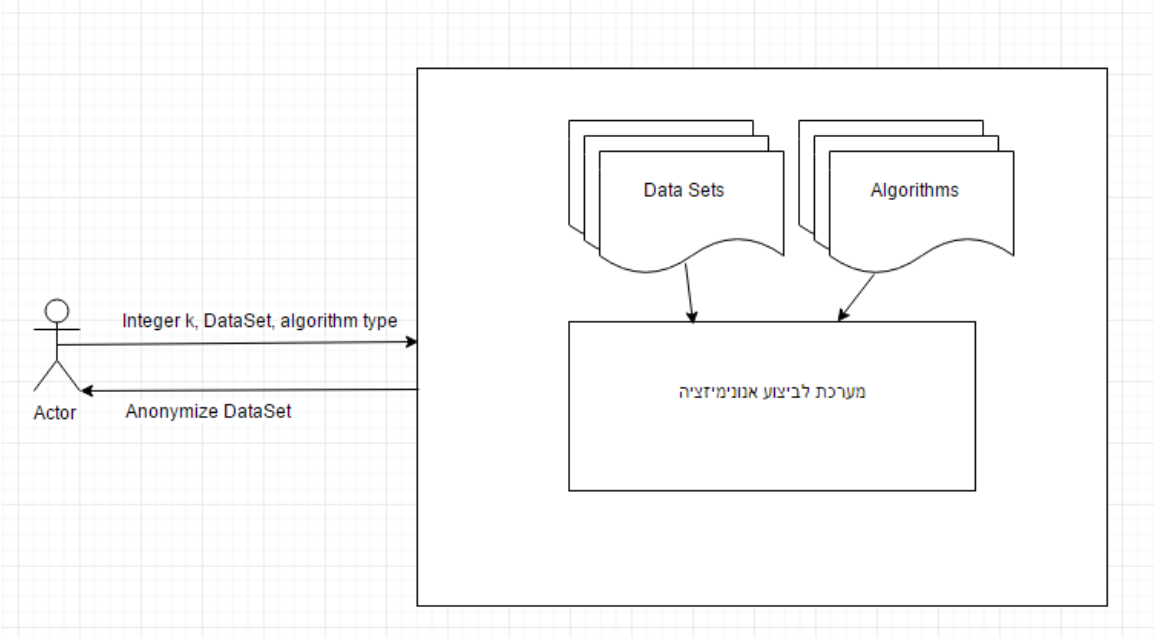
היקף התוכנית הוא לממש את תהליך ביצוע האנונימיזציה של שני אלגוריתמים שונים ולהציג את התוצאות של האלגוריתמים ומדדי ביצועים בהתאם.

2. תיאור כללי

חלק זה יתאר גורמים מרכזיים המשפיעים על התוכנית ועל דרישות התוכנית.
אתאר את הרקע לדרישות אלו אשר יפורט בהרחבה בפרק שלאחריו.

התמקדות במוצר

איור 2 - תיאור סכמתי של המערכת Context diagram.



סביבת המערכת

משתמשי המערכת

המערכת מתקשרת עם המערכות או אנשים הבאים:

- Actor – משתמש אשר יתקשר עם המערכת דרך מסך GUI.
  יבצע את הפעולות הבאות:
  - בחירת מאפיינים שונים לביצוע אנונימיזציה.
  - ביצוע אנונימיזציה על פי המאפיינים שבחר.

מאגרי מידע

המאגרי מידע הם רשתות חברתיות אשר נלקחו מהאתר האקדמאי של אוניברסיטת סטנפורד לצורך מחקרי בלבד.
השוני בין המאגרי מידע נועד בכדי לבדוק את ביצועי התוכנית בעיבוד כמויות מידע שונות.

Facebook circles
מאגר מידע זה מורכב ממידע על מעגלי חברויות מתוך אפליקציית פייסבוק.
כל מספר מתאר משתמש בפייסבוק. כל שורה מתארת שני מספרים שיש להם קשר חברי בפייסבוק.
לדוגמא 1 0 מתאר שמשתמש 0 חבר של משתמש 1 ברשת החברתית פייסבוק.

ArXiv Collaboration network
מאגר מידע זה מתאר האם יש קשר בין מחברים של מאמרים בקטגוריה General Relativity and Quantum Cosmology ברשת arXiv.
כל מחבר מתואר על ידי מספר כלשהו. כל שורה מתארת קשר שבו שני מחברים חיברו מאמר ביחד.

Wiki votes
מאגר מידע זה מכיל מידע על הצבעות תמיכה של משתמשי ויקיפדיה לבחירת מנהל האתר.

כל משתמש מתואר על ידי מספר כלשהו. כל שורה מתארת שני מספרים שיש להם קשר שבו המספר הראשון הצביע למספר השני בכדי שיהיה מנהל בויקיפדיה.

טבלת מידע על מאגרי המידע שיושמו בפרויקט

שם המאגר	לינק	כמות צמתים	כמות קשרים
Facebook circles	https://snap.stanford.edu/data/egonets-Facebook.html	4039	88234
Arxiv Collaboration network	https://snap.stanford.edu/data/ca-GrQc.html	5242	28980
Wiki votes	https://snap.stanford.edu/data/wiki-Vote.html	7115	103689

אלגוריתמים

1. K-Degree Generalization [5]

מבוסס על מאמר מאת Liu and Terzi שבו הם מציגים שמאוד קל לתוקפים להשיג את הדרגה (מספר הקשתות) של הצומת שהם רוצים לתקוף ולכן הם מניחים שיש לתוקף ערך זה כתנאי מקדים.
במאמר [5] הם מסבירים איך לשנות את הגרף על ידי הוספה או הסרה של צמתים לבנייה של גרף אנונימי חדש, כך שלכל צומת בעל דרגה x כלשהו (מספר טבעי או אפס), ישנם עוד k-1 צמתים אחרים ברשת בעלי דרגה x.
בכדי לשמור על שימושיות של הרשת (utility), הם מראים איך משיגים גרף אנונימי במספר מינימלי של שינויים מהרשת המקורית.

לינק למאמר - Towards identity anonymization on graphs. Link

2. K-Symmetry [11]

מבוסס על מאמר מאת Wu, Xiao, Wang, He and Wang שבו הם מציגים מודל חדש שנקרא k-symmetry, בכדי למנוע התקפות לזיהוי אינדיבידואלים בגרף.
במודל זה כל צומת בגרף זהה במבנה הקשרים שלו ל- k-1 צמתים אחרים בגרף.
הרעיון הכללי במודל זה הוא שינוי השיטה הנאיבית כך שלכל צומת v, קיימים לפחות k-1 צמתים אחרים שמשמשים כ-"תמונה" של v תחת אוטומורפיזם של הגרף.
הגדרה לא פורמלית של אוטומורפיזם היא פרמוטציה של צמתי הגרף כך שהפרמוטציה שומרת על קשרי הצמתים המקוריים אך בשינוי סדר כלשהו שאינו מפר קשרים אלו.

לינק למאמר - K-Symmetry model for Identify Anonymization in Social Networks. Link

שני מאמרים אלו נסקרים בפירוט בעבודה המסכמת.

טבלת השוואה בין האלגוריתמים השונים שממומשים במערכת

שאלות	K-Symmetry	K-Degree Generalization
ידע מקדים שיש לתוקף	התוקף יודע שילובים שונים של סביבות של כמה מטרות	הדרגה של צומת המטרה
תיאור הבעיה שהאלגוריתם פותר	תוקף יכול לדעת מראש שילובים שונים של סביבות בקלות על ידי שילובים של כמה רשתות חברתיות, לכן יש צורך לא רק להגן על הדרגה אלא גם אל הסביבה של המטרה.	המטרה יכולה להיות מזוהה מתוך רשת חברתית גם כאשר מורידים מזהים ייחודים.

הפתרון	הרעיון הכללי באלגוריתם הוא לשנות את הרשת כך שכל צומת v יהיו לפחות עוד k-1 צמתים אשר משרתים כתמונה של האוטומורפיזם של הרשת. אוטומורפיזם של הרשת היא פרמוטציה על צמתי הרשת אשר שומרת על הקשרים בין הצמתים.	כל צומת ברשת חברתית G תהיה דומה מבחינת הדרגה לעוד k-1 צמתים אחרים ברשת.

האלגוריתמים השונים פותרים בעיות שונות, כאשר K-Degree מבצע אנונימיזציה כך שכל צומת תהיה דומה ל-k-1 צמתים אחרים מבחינת דרגה. אלגוריתם K-Symmetry פותר בעיה כללית יותר בכך שכל צומת תהיה עם עוד k-1 צמתים שונים ב-"חלוקה" שונה של הגרף, כל שכל חלוקה כזאת שומרת על תכונות דומות בין חברי החלוקה כך שלא ניתן להבחין מבין הצמתים בחלוקה.

תיאור קלט ופלט התוכנית

התוכנית מקבלת כקלט:

- פרמטר k כלשהו אשר מהווה את המדד למידת האנונימיזציה לפי שיטת K-Anonymity, ככל ש-K יהיה גדול יותר כך מאובטח יותר אך ייתכן והשימושיות תיפגע.
- סוג ה- Dataset של נתוני רשת חברתית שאותו נרצה לבצע בו אנונימיזציה.
  ההבדל בין הדוגמאות של ה- Datasets הם במספר הצמתים ובמספר הקשרים שהם מכילים.
  ה-Datasets הועתקו כדוגמאות של רשתות חברתיות והסבר מפורט יותר לגבי הרכבן יובא בהמשך.

- סוג האלגוריתם שהמשתמש יהיה מעוניין להריץ.
  האלגוריתמים שבחרתי הם K-Degree ואלגוריתם K-Symmetry שיהיו זמינים בתוכנית זו.
  תיאור מפורט של האלגוריתמים נמצא בעבודה המסכמת.

התוכנית תחזיר כפלט:

- את אותו ה- Dataset שהמשתמש בחר אך בצורה אנונימית לפי הפרמטר k ובהתאם לסוג האלגוריתם שברצונו להריץ.
- ביצועי האלגוריתם מבחינת זמן ומדדים שונים של שימושיות שאותם אפרט בהמשך.

הנחות

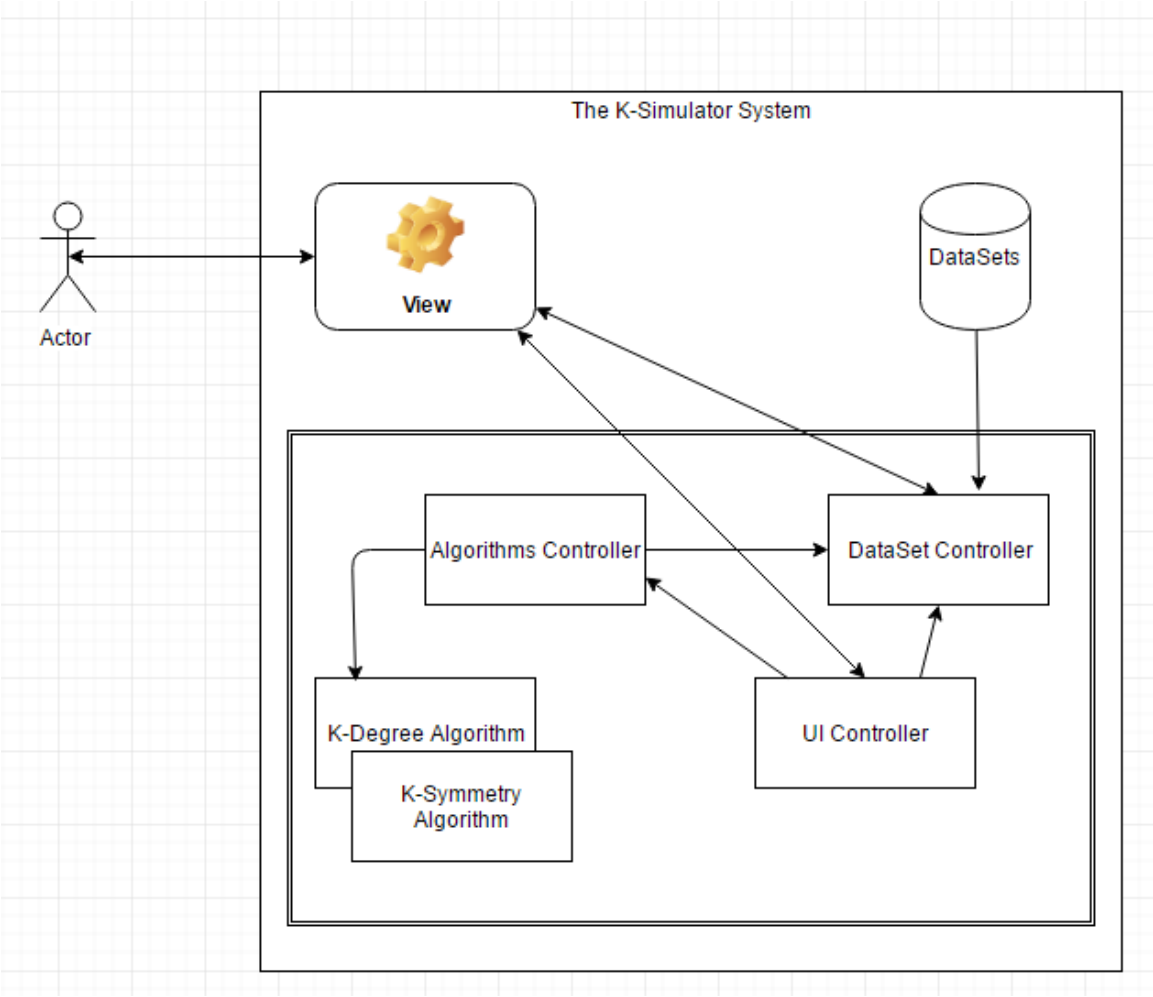
- למערכת אין אינטגרציות עם ממסדי נתונים כלשהם לשמירה או גיבוי נתונים.
- למערכת אין צורך בזיהוי המשתמש או להגדיר חוקי שימוש כלשהם.
- המערכת אינה לימודית, כלומר ההנחה היא שהמשתמש מכיר את הנושא ואין צורך ללמד אותו את האלגוריתמים.

3. עיצוב המערכת

ניתוח ועיצוב המערכת לפי מתודולוגיה של מונחה עצמים (Object-Oriented)

ארכיטקטורה

איור 3 – דיאגרמת ארכיטקטורה של המערכת



המערכת מורכבת משלושה חלקים מרכזיים:

- Model – תפקידו של חלק זה הוא לתאר את המודל ולשמור את אבני הבניין שבהם נשתמש במערכת.
  המודל שנרצה לעבוד איתו הוא רשת חברתית אשר בנויה מצמתים וקשרים.
  ובנוסף יטען את מאגרי המידע וישמור אותם בזיכרון.

- View – תפקידו של חלק זה הוא לתאר את הרכיבים הגרפיים שיתקשרו עם המשתמש.
  ליצור את האירועים שיקרו בעקבות לחיצות המשתמש ובקבלת תוצאות חדשות לעדכן את הרכיבים הגרפיים בהתאם.
  הממשק למשתמש יושם באמצעות ספריית Swing.

- Controller – תפקידו של חלק זה הוא לבצע את תהליכי החישוב בהינתן מודל מסוים או קבלת אירוע לחיצה מה- View.
  ישנם כמה סוגי לוגיקה שונות:

- Dataset Controller – תפקידו של חלק זה הוא לטעון את מאגרי המידע מקבצים למבנה נתונים במודל אשר נוכל לעבד אותו.

- Algorithms Controller – תפקידו של חלק זה הוא בהינתן קונפיגורציה מסוימת מה- view עליו לתרגם זאת להבנת המאגר מידע שבו יש להשתמש והאלגוריתם שיש להפעיל ויצירת model מתאים.

- UI Controller - תפקידו של חלק זה הוא לדעת לתאם בין הרכיבי ה- UI השונים מבחינה אסינכרונית כאשר נקראים אירועים על ידי לחיצות המשתמש והתאמת הפלט מה- controller השונים לעדכון ה- view בהתאם.

תיאור מפורט של המחלקות ותרשימי הזרימה נמצא בנספחים.

ממשק משתמש

ממשק משתמש מורכב משני אזורים מרכזיים:

- אזור הקונפיגורציה
- אזור התוצאות

תיאור תרחיש של בחירת קונפיגורציה ובחינת התוצאות:

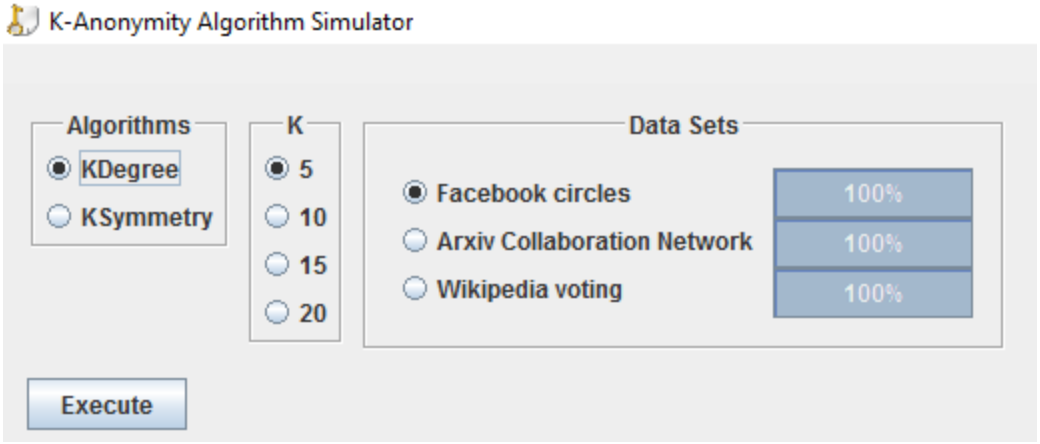
מספר	ישות	קלט, פעולה, פלט
1	המשתמש	בחירת סוג האלגוריתם לביצוע
2	המשתמש	בחירת גודל פרמטר k
3	המשתמש	בחירת סוג המאגר נתונים
4	המשתמש	הפעלת האלגוריתם לפי הקונפיגורציה שנבחרה.
5	appFrame	קבלת הקונפיגורציה והעברתה לקונטרולר.
6	algorithmController	קבלת הקונפיגורציה והעברת הנתונים למימוש של האלגוריתם שנבחר.
7	algorithmImpl	הרצת האלגוריתם לפי הקלט שנבחר והחזרת מאגר נתונים אנונימי.
8	appFrame	קבלת מאגר נתונים אנונימי להצגה

אזור הקונפיגורציה

באזור זה המשתמש יבחר את הקונפיגורציה לצורך הרצת האלגוריתם.
לקונפיגורציה תהיה בחירת מחדל שהיא אלגוריתם K-Degree.
סוגי בחירות אפשריות:

- בחירת סוג האלגוריתם – במערכת קיימים שני סוגי אלגוריתמים שמומשו במסגרת הפרויקט.
- בחירת גודל פרמטר k – לצורך פשטות ומניעת בדיקת קלט ניתן לבצע בחירה מבין הגדלים הבאים: 5, 10, 15, 20.
- בחירת מאגר מידע – במערכת קיימים שלושה מאגרי מידע שנטענו לזיכרון בהפעלת המערכת.
לכל מאגר מידע ישנו חיווי אשר יראה כמה אחוז הוא נטען לזיכרון (מקסימום יגיע ל-100%), אינדיקציה זו מאפשרת למשתמש להבין מתי ניתן יהיה לבצע הפעלה לאלגוריתם.

איור 3 – מסך בחירת המאפיינים להרצה



מכיוון שלא הגיוני לבצע בחירה של כמה מאגרי מידע באותה הפעלה או כמה אלגוריתמים באותה הרצה, הבחירה תהיה מאופשרת על ידי radio buttons המאפשרים בחירה יחידנית של כל סוג פרמטר.

אזור התוצאות

באזור זה המשתמש יוכל להשוות ולנתח את תוצאות ההרצה.
בעת טעינת מאגרי המידע לזיכרון יטען בנוסף מסך תוצאות לבחינת מאגר המידע האנונימי.
אזור זה מחולק לטאבים לפי סוג מאגר המידע בכדי שיהיה יותר נח להשוות הרצות מאותו המאגר.

תוצאות הרצה של אלגוריתם k-Degree בנויות משלושה אזורים שונים:

טבלת מיפוי דרגות

Degree	Vertices
111	10
110	5
109	10
108	5
106	10
105	5
104	5
103	5
102	15
101	5
100	20
99	10
98	10
97	10
96	5
95	5
94	15
93	5
92	10
91	5
90	15

טבלה זו מכילה שתי עמודות אשר ניתנות למיון.
- Degree – עמודה זו מכילה טור של דרגות במאגר.

- Vertices – עמודה זו מכילה את מספר הצמתים שנמצאו לאותה דרגה.

לדוגמא השורה {5, 104} מציינת שנמצאו חמישה צמתים שונים עם דרגה 104.

<u>טבלת פרטנית על כל צומת</u>

Vertex	Degree	Degree similarity	Vertices
1003	104	20.0	[1340, 1184, 1613, 1612, 1059, 1851, 2543, 1331, ...
1006	148	20.0	[1341, 1063, 1613, 1612, 1336, 993, 1214, 1335, 1...
1017	154	20.0	[1187, 1341, 1185, 1184, 1181, 1612, 1336, 1457, ...
1024	93	20.0	[1458, 1898, 1017, 1336, 3437, 1214, 1456, 1610, ...
1032	86	20.0	[1066, 1582, 1020, 1261, 1382, 1018, 596, 1215, 1...
1049	88	20.0	[1340, 1184, 1534, 3437, 2347, 1059, 1653, 1730, ...
1059	171	20.0	[1341, 1340, 1185, 1184, 1180, 1613, 1612, 993, 1...
1075	91	20.0	[1340, 1460, 1184, 353, 3716, 3437, 2347, 2622, 1...
1076	148	20.0	[1341, 1340, 1185, 1184, 1181, 1613, 1612, 993, 1...
1086	211	20.0	[596, 1457, 1214, 1335, 1456, 2543, 1211, 916, 91...
1104	116	20.0	[1462, 1583, 1060, 1578, 1611, 1576, 1575, 1298, ...
1107	148	20.0	[1341, 1460, 1184, 1613, 1458, 1612, 1457, 1335, ...
1124	136	20.0	[1341, 1184, 1580, 1180, 1458, 1214, 1335, 1610, ...
1126	202	20.0	[1341, 1336, 1335, 1456, 2543, 1211, 1331, 1339, ...
1128	103	20.0	[1341, 1185, 1184, 2394, 1612, 993, 1610, 1059, 1...
1132	145	20.0	[1341, 1185, 1184, 1580, 1613, 1612, 1336, 1214, ...
1153	146	20.0	[1341, 1185, 1184, 1580, 1181, 1336, 1457, 1214, ...
1156	105	20.0	[472, 1612, 1456, 1610, 1730, 2543, 1211, 1331, 1...
1172	119	20.0	[907, 1341, 1580, 1613, 993, 1214, 1335, 1456, 16...
1173	121	20.0	[1460, 1580, 1180, 1613, 1214, 1335, 1577, 1610, ...
1184	164	20.0	[1341, 1340, 1185, 1181, 1613, 1612, 993, 1335, 1...

טבלה זו מכילה את המידע הבא:

- Vertex – צומת כלשהי שנמצאה במאגר, המספר מציין את ה- ID של הצומת.
- Degree – הדרגה של הצומת
- Degree similarity – ההסתברות שתוקף יסיק את הצומת בהינתן דרגה, לדוגמא 20 מציין שלתוקף יש 20% להסיק שזהו הצומת בהינתן שיש לתוקף מידע על דרגת הצומת, כלומר ישנם עוד צמתים בעלי דרגה 104 בגרף האנונימי.
- Vertices – רשימת השכנים של הצומת.

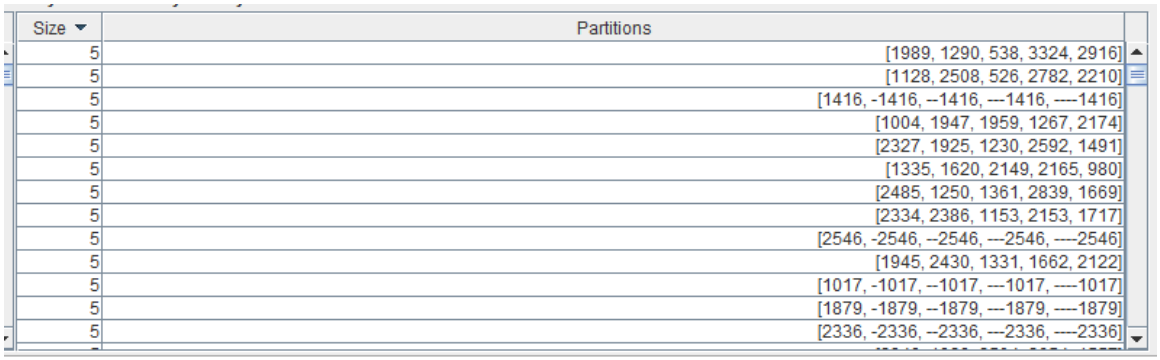
<u>טבלת מדדים</u>

Total Vertices	4039
Vertices added (%)	0 (0.00%)
Total Edges	95960
Edges added (%)	7726 (8.76%)
Duration	1.38sec

- Total Vertices – כמות הכוללת של צמתים במאגר.
- Vertices Added (%) – אחוז הצמתים שנוספו יחסית למאגר המקורי.
- Total Edges – כמות הכוללת של קשרים במאגר.
- Edges Added (%) – אחוז הקשרים שנוספו יחסית למאגר המקורי.
- Duration – הזמן שלקח לאלגוריתם לרוץ, הזמן מוצג ביחידות של שניות.

בעת הרצה של אלגוריתם k-Symmetry נוסיף טבלה נוספת לבחינת התוצאות בכדי לראות את גודל החלוקות.

איור 4 – טבלת החלוקות לפי גודל



Size	Partitions
5	[1989, 1290, 538, 3324, 2916]
5	[1128, 2508, 526, 2782, 2210]
5	[1416, -1416, --1416, ---1416, ----1416]
5	[1004, 1947, 1959, 1267, 2174]
5	[2327, 1925, 1230, 2592, 1491]
5	[1335, 1620, 2149, 2165, 980]
5	[2485, 1250, 1361, 2839, 1669]
5	[2334, 2386, 1153, 2153, 1717]
5	[2546, -2546, --2546, ---2546, ----2546]
5	[1945, 2430, 1331, 1662, 2122]
5	[1017, -1017, --1017, ---1017, ----1017]
5	[1879, -1879, --1879, ---1879, ----1879]
5	[2336, -2336, --2336, ---2336, ----2336]

הטבלה מכילה שתי עמודות:

- Size – גודל החלוקה מבחינת כמות הצמתים.
- Partitions – הצמתים השונים הנמצאים בחלוקה זו.

טכנולוגיות ותלויות

אמינות

- מאגרי המידע המקוריים והאנונימיים יטענו לזיכרון.

זמינות

- המערכת תהיה זמינה עד אשר המשתמש יסגור את התוכנה.

תחזוקתיות

- המערכת תיכתב בשפת JAVA אשר קלה יותר לתחזוקה.

ניידות

- המערכת תוכל לרוץ בכל מערכת הפעלה אשר מכילה Java runtime environment 1.7.

תלויות תוכנה

- Java 1.7 – שפת תכנות.
- Swing – ספריה אשר כתובה ב- Java אשר מספקת מסכי GUI.
- Spring framework – תשתית ל-MVC .
- Maven – ספריה לניהול התלויות ובניית הקוד.
- Log4j – ספריה אשר מספקת שירותי כתיבה ללוג.
- Junit – ספריה אשר מספקת שירותי כתיבת טסטים.

כלי פיתוח

- IDEA InteliJ - סביבת פיתוח.

בדיקות המערכת

<u>בדיקות ידניות</u>

בהרצת בדיקות ידניות נוכל לדעת שהאלגוריתם החזיר תוצאה כפי שציפינו.

- בדיקת אלגוריתם K-Degree
  - האלגוריתם לא משנה את המספר הסופי של צמתים.
  - האלגוריתם יכול לשנות את המספר הסופי של קשרים.
  - בטבלת המיפוי מדרגה לצומת, עמודת כמות הצמתים חייבת להיות גדולה או שווה לפרמטר k אשר הוכנס כקלט.
- בדיקת אלגוריתם K-Symmetry

- האלגוריתם יכול לשנות את המספר הסופי של צמתים.
- האלגוריתם יכול לשנות את המספר הסופי של קשרים.
- בטבלת המיפוי של גודל חלוקה לצמתים, עמודת גודל החלוקה "Size" חייבת להיות גדולה או שווה לפרמטר k אשר הוכנס כקלט.

<u>בדיקות אוטומטיות</u>

- בדיקת אלגוריתם K-Degree
  - ניצור גרף רנדומלי בגודל כלשהו.
  - נריץ את האלגוריתם עם k=5
  - נבדוק שבגרף האנונימי לא כל צומת בגרף יש דרגה לפחות 5.
  - נבדוק שמספר הצמתים בגרף האנונימי שווה למספר הצמתים בגרף המקורי.

- בדיקת אלגוריתם K-Symmetry
  - ניצור גרף רנדומלי בגודל כלשהו.
  - נריץ את האלגוריתם עם k=2
  - נבדוק שמספר הצמתים הסופי הוא 10.
  - נבדוק שגודל כל חלוקה בגרף האנונימי הוא לפחות 2.
  - ניצור גרף רנדומלי בגודל כלשהו.
  - נריץ את האלגוריתם עם k=3
  - נבדוק שמספר הצמתים הסופי הוא 18.
  - נבדוק שגודל כל חלוקה בגרף האנונימי הוא לפחות 3.

- בדיקת קלאס עזר DegreeUtil
  - ניצור גרף רנדומלי בגודל כלשהו כך שלא קיים קשר בין "1" ל- "2"
  - נבדוק שמתודה האם קיים קשר תחזיר "לא" האם יש קשר בין "1" ל-"2"
  - ניצור גרף רנדומלי בגודל כלשהו כך שכן קיים קשר בין "1" ל- "2"
  - נבדוק שמתודה האם קיים קשר תחזיר "כן" האם יש קשר בין "1" ל-"2"
  - ניצור גרף רנדומלי בגודל כלשהו.
  - נריץ מיון לפי דרגות בצורה יורדת.
  - נבדוק שהוקטור שמתקבל ממוין.

4. הרצת התוכנית

פקודות להרצת המערכת

הקוד מאוחסן ב-GitHub בצורה פומבית בכתובת הבאה:
https://github.com/Keinang/K-Anonymity

בכדי להריץ את התוכנית יש לבצע את הפעולות הבאות:

- הורדת הקוד למחשב לוקאלי
  git clone https://github.com/Keinang/K-Anonymity.git

- קמפול הקוד - mvn clean install

- הרצה המערכת -
  בתיקייה "target" תיווצר קובץ K-Anonymity-1.0-SNAPSHOT.jar
  Java -jar K-Anonymity-1.0-SNAPSHOT.jar

דוגמאות הרצה

נריץ את כל השילובים האפשריים של האלגוריתמים על מאגר מידע "Facebook circles".
השורה הראשונה מציינת שלא הופעל אלגוריתם על המאגר המידע, כלומר הוא לא אנונימי.

טבלת הרצות על מאגר מידע Facebook Circles

Data Set	Algorithm	K	Total vertices	Vertices Added (%)	Total Edges	Edges Added (%)	Duration (sec)
Facebook circles	Original Graph	0	4039	0	88234	0	0
Facebook circles	K-Degree	5	4039	0	95322	8.03	3.16
Facebook circles	K-Degree	10	4039	0	106184	20.34	3.18
Facebook circles	K-Degree	15	4039	0	102015	15.62	1.47
Facebook circles	K-Degree	20	4039	0	123893	40.41	2.71
Facebook circles	K-Symmetry	5	4386	8.59	181487	105.69	11.4
Facebook circles	K-Symmetry	10	5162	27.8	458512	419.65	34.28
Facebook circles	K-Symmetry	15	5988	48.25	867771	889.49	101.62
Facebook circles	K-Symmetry	20	6967	72.49	1433005	1524.10	134.85

בהרצות הנ"ל ניתן לראות שככל ש-k גדל, נוספו יותר קשרים בגרף ולאלגוריתם לוקח יותר זמן לסיים.

יעילות האלגוריתמים על ידי זיהוי צמתים דומים עם אותה דרגה

מכיוון שלא ניתן להשוות בין שני האלגוריתמים מכיוון שהם שונים בפתרון הבעיה שהם פותרים, אנסה להגדיר מידת שימושיות ואעקוב אחר צומת כלשהי ואבדוק את מידת הפרטיות שלה, כלומר כמה היא ניתנת לזיהוי מהגרף האנונימי לאורך הרצות שונות.
את ההרצות אריץ על Facebook circles.
הגדרת שימושיות – הדרגה של הצומת, כלומר כמה צמתים שונים יש בגרף עם אותו מספר דרגה.
צומת מעקב – 2630

מדידות עם k משתנה בעזרת אלגוריתם k-Degree

Algorithm	K	Degree of Vertex 2630	How many	Re-identify

			vertices with the same degree	
Original Graph	0	173	6	16.67%
K-Degree	5	178 (+2.89%)	5	20% (+3.34%)
K-Degree	10	189 (+9.24%)	10	10% (-6.67%)
K-Degree	15	180 (+4.04%)	15	6.67% (-10%)
K-Degree	20	191 (+10.4%)	20	5% (-11.67%)

ניתן להבחין שככל ש- K גדל, כך קשה יותר לזהות את הצומת '2630 מבחינת הדרגה.

מדידות עם k משתנה בעזרת אלגוריתם k-Symmetry

Algorithm	K	Degree of Vertex 2630	Vertices in the same orbit	Re-identify
Original Graph	0	173	6	16.67%
K-Symmetry	5	338 (+195.37%)	6	16.67% (0%)
K-Symmetry	10	610 (+352.6%)	12	8.83% (-8.83%)
K-Symmetry	15	875 (+505.75%)	18	5.55% (-11.12%)
K-Symmetry	20	1143 (+660.69%)	24	4.16% (-12.51%)

ניתן להבחין שככל ש- K גדל, כך קשה יותר לזהות את הצומת '2630 מבחינת הדרגה.

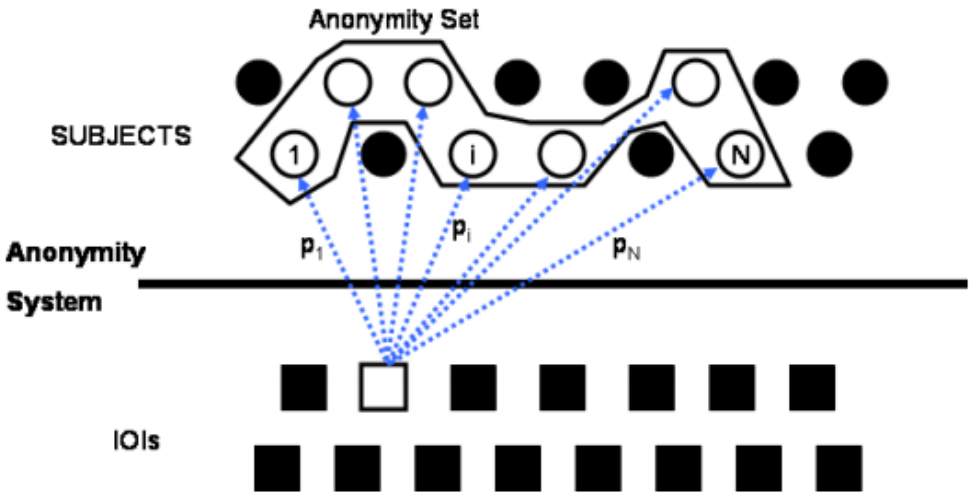
יעילות האלגוריתמים על פי מודל אנטרופיה

מערכות רבות ניתנות למידול במונחים של אי-קישוריות. המשמעות של אי-קישוריות של שניים או יותר איברים היא שבתוך המערכת איברים אלו לא יותר ולא פחות קשורים מאשר ידע מקדים כלשהו מחוץ למערכת.

על פי מודל ממאמר של קלאודיה דיאז [15] האיברים במערכת נקראים Items of Interest. אנונימיזציה מוגדרת לפי היכולת לקשר בין IOI לבין Subject כלשהו. בדרך זו ניתן לתאר את האנונימיזציה (אי-קישוריות) של IOI כלשהו ל- Subject כלשהו ובנוסף לתאר את האנונימיזציה של Subject כלשהו להיות מקושר ל-IOI כלשהו.

Anonymity Set היא קבוצה של Subjects אשר יכולה להיות מקושרת ל- IOI כלשהו עם הסתברות גדולה מ-0, ככל שקבוצה זו תהיה גדולה יותר כך האנטרופיה תהיה גדולה יותר ונהנה מאנונימיות גדולה יותר.

<u>איור 5 – תיאור המודל לפי אנטרופיה</u>



איור 5 מתאר בפשטות את המודל לאנונימיות כאשר המטרה היא להחביא את הקשר בין IOI ל- Subject. ואת קבוצת ה- Anonymity Set שיכולה להיות מקושרת ל- IOI מסוים.

הקונספט של שימוש באנטרופיה נותן יכולת לכמת את האי וודאות של משתנה מסוים.

נגדיר ב- p_i את ההסתברות ש- Subject יהיה מקושר ל- IOI כלשהו.

בצורה נאיבית נגדיר ש-Subject יהיה מקושר ל- IOI במידה ויש לו את אותה דרגה. ניתן להבחין שאפיון זה הוא נאיבי מכיוון שהוא לא בהכרח נכון מכיוון שהאלגוריתמים מוסיפים קשרים לצמתים ולכן דרגתם משתנה, בנוסף ניתן לקשר על ידי השוואת מאפיינים על הקשר או על הצומת.

נגדיר את האנטרופיה של קבוצת ה- Anonymity Set כ- $H(\mathrm{X}) = -\sum_{i=1}^{N} p_i \log_2(p_i)$

X היא קבוצת ה- Subjects בעלת הסתברות מעל 0 כמקושרת לאותו ה- IOI. N הוא מספר הכולל של איברים במערכת ולא רק בקבוצה מכיוון שלאחר תהליך האנונימיזציה מוסיפים ומורידים קשרים ולכן אין לנו וודאות שאותו צומת תישאר עם אותו מספר דרגות, לכן תמיד האנטרופיה המקסימלית היא לפי המספר הכולל של דרגות.

בכדי לדעת כמה טובה האנטרופיה של ה- Anonymity Set ניתן לנרמל זאת לפי סקלה של [0..1], נגדיר את האנטרופיה המקסימלית שניתן להגיע אליה ב- Anonymity Set שהיא - $H_M = \log_2 N$

כעת נגדיר את הדרגה של האנונימיות ב- Anonymity Set כך - $d = \frac{H(X)}{H_M}$

כפי שניתן לראות מהנוסחה, הדרגה של האנונימיות מתקבלת על ידי חלוקת האנטרופיה של ה- Anonymity Set באנטרופיה המקסימלית של ה- Anonymity Set, חישוב זה נותן לנו מדד כמה אנונימיות השגנו לאחר הפעלת האלגוריתם לאותו IOI.

<u>פסאודו קוד לחישוב אנטרופיה</u>

1. For each vertex from the original graph (Iterating IOIs) do
   a. Keep vertex's degree → original degree
   b. Get anonymized set size from the anonymized graph → anonymized_set
   c. sum → 0
   d. For each vertex in the anonymized_set do
      i. 1.0 /anonymized_set.size() → pi
      ii. pi * log (pi) → append to sum
      iii. -1 * sum → hX
      iv. log(total_vertices) → hM
      v. hX / hM → entropy (v)

באלגוריתם עוברים על כל הצמתים בגרף המקורי ועל כל אחד מהם מקבלים את דרגת הצומת, לכל צומת שעוברים באיטרציה מחשבים סט צמתים שדומים לו בגרף האנונימי מבחינת דרגה .
Anonymized_set, כעת, לכל צומת מתוך סט אנונימי זה מחשבים הסתברות למציאת הצומת (מתפלג שווה בין כל הצמתים על ידי *anonymized_set.size()* ומחשבים סכום של כל ההסתברויות של כל הצמתים בסט.

האנתרופיה הסופית היא סכום *hX* חלקי ה *hM* - שהיא ההסתברות המקסימלית למציאת צומת כלשהי מכלל הצמתים בגרף האנונימי שהתקבל.

<u>דוגמא על צומת 2630 בעלת דרגה 173 במאגר מידע Facebook Circles</u>

Algorithm	K	Entropy
Original Graph	0	0.216
K-Degree	5	0.277
K-Degree	10	1.0
K-Degree	15	1.0
K-Degree	20	1.0

Algorithm	K	Entropy
Original Graph	0	0.216
K-Symmetry	5	0.083
K-Symmetry	10	0.0
K-Symmetry	15	0.126
K-Symmetry	20	0.281

תמונות הרצה מלאות נמצאות בנספחים.

ניתן להבחין מהתוצאות שבהרצת K-Degree ככל ש-k גדל כך האנטרופיה גדלה, כלומר השגנו יותר אנונימיות. ב- k=10 קיבלנו שהאנטרופיה היא 1.0 (מקסימלית) והיא נשארת כך ככל ש-k גדל.

לעומת זאת, בהרצת אלגוריתם K-Symmetry ככל ש-k גדל האנטרופיה לא גודלת ועל אף יורדת, זאת מכיוון שהגדרנו שההסתברות ש- Subject מסוים יהיה קשיר ל- IOI רק כאשר יש לשניהם את אותה הדרגה, אך באלגוריתם זה לא מוודאים שיש לפחות k צמתים שונים עם אותה דרגה, אלא מוודאים שמספר הצמתים באותה חלוקה הוא לפחות k צמתים, כלומר חישוב האנטרופיה במקרה זה צריך להיות שונה ולהתאים לבעיה שפותרת האלגוריתם ולכן ההשוואה אינה נכונה במקרה זה.

5. חשיבות הפרויקט

נושא זה של שמירת פרטיות המידע של רשתות חברתיות נמצא בתהליך מתמשך של מחקר ועניין. בעבודה זו אני ממחיש מדוע הגישה הנאיבית אינה מבטיחה פרטיות למידע ולכן לא מומלצת לשימוש, כמו כן עבודה זו סוקרת אלגוריתמים לשמירת פרטיות המידע מקטגוריה של K-Anonymity ומציינת קטגוריות נוספות.
מכיוון שרשתות חברתיות הם יותר מורכבות מאשר מאגרי מידע טבלאיים ישנו אתגר רב לשמור על פרטיות המידע ברשתות חברתיות, לא רק על ידי השאלת האלגוריתמים שנלמדו על מאגר מידע טבלאי אלא גם ביישומם על רשתות חברתיות.

6. רשימת מקורות

[1] X. Wu, X.Ying, K. Liu and L. Chen. "A Survey of Algorithms for Privacy-Preservation of Graphs and Social Networks". Invited book chapter. Managing and Mining Graph Data. Editors Charu C. Aggarwal and Haixun Wang. Kluwer Academic Publishers. August 2009

[2] B. Zhou, J. Pei, and W.-S. Luk. A brief survey on anonymization techniques for privacy preserving publishing of social network data. SIGKDD Explorations, 10(2), 2009.

[3] J. Casas-Roma, J. Herrera-Joancomartí , V. Torra: Comparing Random-Based and k-Anonymity-Based Algorithms for Graph Anonymization. MDAI 2012: 197-209

[4] L. Backstrom, C. Dwork, and J. Kleinberg. Wherefore art through anonymized social networks, hidden patterns, and structural steganography. In proceedings of the 16th international conference on World Wide Web, pages 181{190, New York, NY, USA, 2007. ACM Press.

[5] K. Liu and E. Terzi. Towards identity anonymization on graphs. In Proceedings of the ACM SIGMOD Conference, Vancouver, Canada, 2008. ACM Press.

[6] B. Zhou and J. Pei. Preserving privacy in social networks against neighborhood attacks. In proceedings of the IEEE 24th International Conference on Data Engineering, pages 506{515, 2008.

[7] L. Zou, L. Chen, and M. T. Ä Ozsu. K-automorphism: A general framework for privacy preserving network publication. In proceedings of 35th International Conference on Very Large Data Base, 2009.

[8] M. Hay, G. Miklau, D. Jensen, D. Towsely, and P. Weis. Resisting structural re-identification in anonymized social networks. In VLDB, 2008.

[9] Cheng, J., Fu, A.W.-C., Liu, J.: K-isomorphism: privacy-preserving network publication against structural attacks. In: SIGMOD (2010)

[10] Y. Song, P. Karras, Q. Xiao and S. Bressan, A. Ailamaki and S. Bowers, "Sensitive label privacy protection on socialnetwork data" in Scientific and Statistical DatabaseManagement, vol. 7338, pp. 562-571, 2012, Springer

[11] Wu, W., Xiao, Y., Wang, W., He, Z., Wang, Z.: K-symmetry model for identity anonymization in social networks. In proceedings of EDBT (2010)

[12] E. Zheleva, L. Getoor, "Privacy in Social Networks: A Survey", In proceedings of Social Network Data Analytics, Springer US, pp 277-306, 2011.

[13] F. Bonchi, A. Gionis, and T. Tassa. Identity obfuscation in graphs through the information theoretic lens. In ICDE, pages 924–935, 2011.

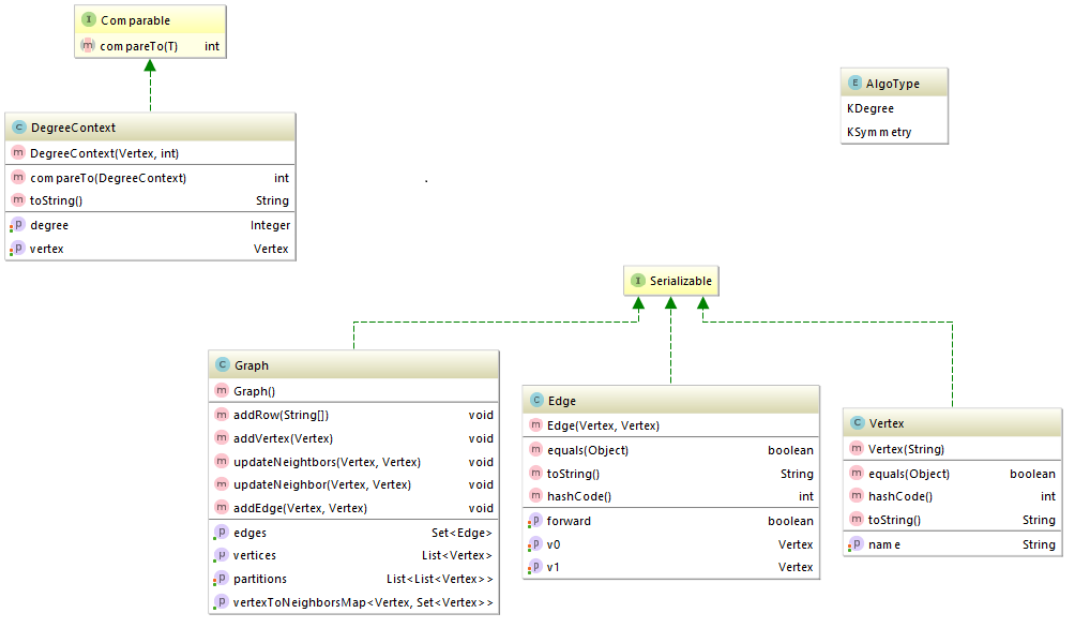
[14] L. Sweeney: k-Anonymity: A Model for Protecting Privacy. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 10(5): 557-570 (2002)

[15] Diaz, C. 2006. Anonymity metrics revisited. In proceedings of Anonymous Communication and its Applications, number 05411

7. נספחים

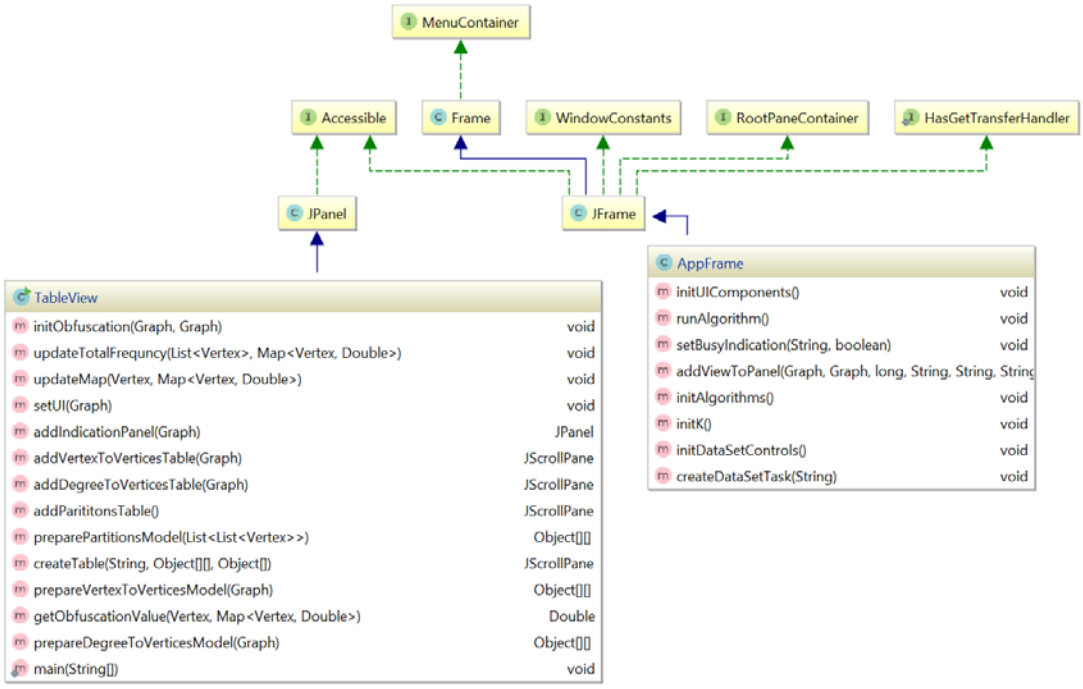
דיאגרמת מחלקות

ה- Model



- Degree Context- מחלקה שאחראית לשמור על מבנה נתונים של צומת ומספר הדרגות של הצומת. כמו כן, יודעת למיין את הצמתים לפי כמות הדרגות.
- Graph – מחלקה שאחראית לשמור על מבנה נתונים של רשת חברתית.
  - addRow – מתודה שאחראית להוסיף שורה מתוך קובץ Dataset אשר מכיל מידע על 2 צמתים והקשר ביניהם לתוך הרשת חברתית.
  - addVertex – מתודה שאחראית להוסיף צומת לתוך הרשת חברתית.
  - updateNeighbor/s – מתודות שמעדכנות את מספר הדרגות כאשר מתקבלת צומת חדשה לרשת.
  - addEdge – מתודה שאחראית להוסיף קשר חדש בין שני צמתים לרשת.
  - vertexToNeighbors – שדה שמאחסן לכל צומת ברשת את הצמתים ששכנים אלו מרמה ראשונה. (כלומר יש קשר ישיר).
- Edge – מחלקה שאחראית לשמור על מבנה נתונים של קשר ברשת החברתית.
  - כל קשר שומר בין אלו שני צמתים יש קשר.
- Vertex – מחלקה שאחראית לשמור על מבנה נתונים של צומת ברשת חברתית.
  - ההנחה היא שכל צומת מזוהה ייחודית על ידי שם כלשהו.
- AlgoType – רשימת שמות סגורה של האלגוריתמים שזמינים במערכת.

ה- View

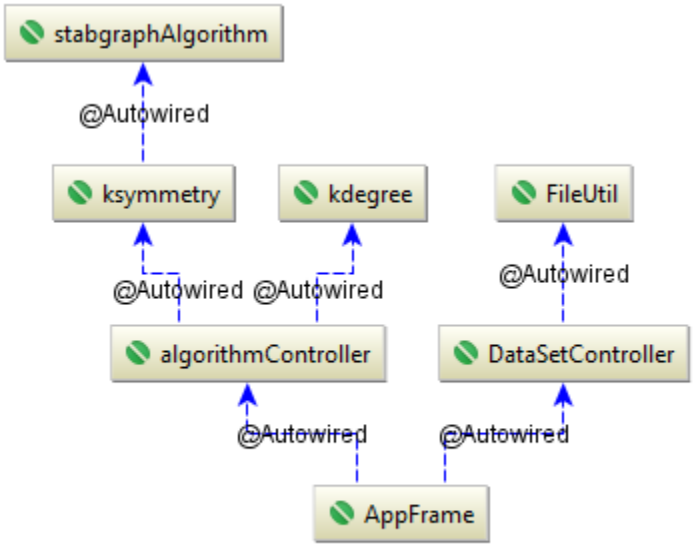


- AppFrame – מחלקה שיורשת מ- JFrame ואחראית על ה- frame הראשי על המערכת, היא המחלקה שעוטפת את כל ה-views ומאתחלת את כל רכיבי ה- UI.
  - initUIComponents – מתודה שאחראית לאתחול כל הרכיבים הגרפים ב- Frame
  - runAlgorithm – מתודה שאחראית ליצור תרד' חדש ולהריץ בו את האלגוריתם לפי הקונפיגורציה שנבחרה.
  - setBusyIndication – מתודה שאחראית להפעיל ולהפסיק את החיווי ב- Frame שישנו תהליך שעובד ברקע.
  - addViewToPanel – מתודה שאחראית להוסיף View חדש כלשהו לתוך Panel כלשהו בכדי לראות מספר תוצאות ריצה במקביל.
  - initAlgorithms – מתודה שאחראית לקבל מה- AlgorithmController את המידע אילו אלגוריתמים זמינים במערכת ולהציג רכיבים גרפיים לבחירה בהתאם.
  - initK – מתודה שאחראית ליצור רכיבים גרפיים בכדי לבחור את ערכי K.
  - initDataSetControls – מתודה שאחראית ליצור רכיבים גרפים לפי המאגרי מידע הקיימים במערכת. מידע זה מתקבל מ- DataSetController.
  - createDataSetTask – מתודה שאחראית ליצור תרד' חדש בכדי לטעון Dataset, מכיוון שפעולה זו יכולה לקחת זמן בהתאם לגודל המאגר מידע, יש צורך בלטעון ברקע ולא לפגוע בשימושיות של המשתמש.
- TableView- מחלקה שאחראית להצגת נתונים לאחר הרצת האלגוריתם. המחלקה נקראת TableView מכיוון שהיא מאתחלת כמה רכיבים גרפים מסוג JTable.
  - initObfuscation – מתודה שאחראית להציג את הסטטיסטיקה של שימושיות המידע לפני לעומת אחרי הרצת האלגוריתם.
  - updateTotalFrequency, updateMap – מתודות שאחראיות על עדכונים של המבני נתונים בכדי לחשב את שימושיות המידע לאחר ביצוע האלגוריתם.
  - setUI – מתודה שמאתחלת את כל הרכיבים הגרפים.

- addIndicationPanel – מתודה שאחראית להוסיף Panel שמראה את כל הסטטיסטיקות לתוך ה- View הראשי.
- addVertexToVerticesTable – מתודה שאחראית להוסיף Panel שמראה את כל השכנים של כל צומת ברשת.
- addDegreeToVerticesTable – מתודה שאחראית להוסיף Panel שמראה לכל כמות דרגות את הצמתים שיש להם את אותו מספר דרגות.
- addPartitionsTable – מתודה שאחראית להוסיף את טבלת החלוקות (שימושי באלגוריתם K-Symmetry).
- preparePartitionsModel – מתודה שאוספת את כל המידע שצריך בשביל טבלת החלוקות.
- PrepareVertexToVerticesModel – מתודה שאוספת את כל המידע שצריך בשביל טבלת השכנויות.
- PrepareDegreeToVerticesModel – מתודה שאוספת את כל המידע שצריך בשביל טבלת הדרגות לצמתים.
- createTable – מתודה ליצירת רכיב טבלה בעזרת נתונים כלשהם.

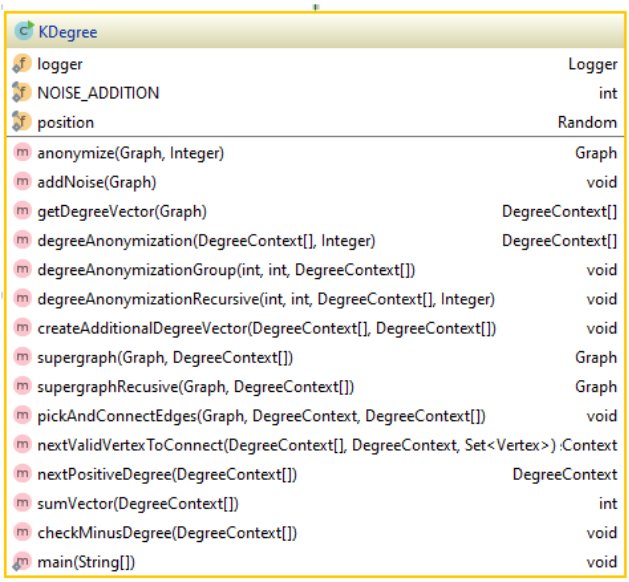
ה- Controller

מודול Algorithms ו- DataSetController

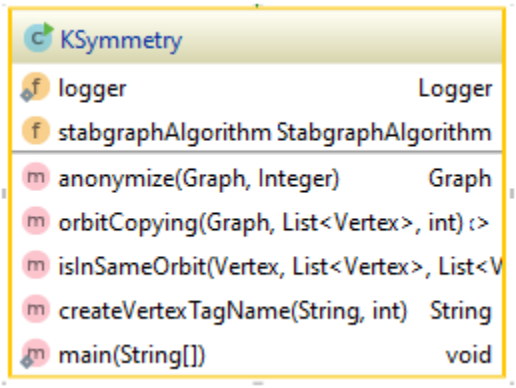


- AlgorithmController – מחלקה ראשית לביצוע אלגוריתמים במערכת. למחלקה פונקציה אחת anoniymize אשר מקבלת קונפיגורציה כלשהי של מספר k, סוג האלגוריתם להרצה ואת המאגר מידע שמעוניינים להריץ ומחזירה רשת חברתית אנונימית בהתאם.

- kDegree – מחלקה אשר מממשת את אלגוריתם K-Degree שמפורט בהרחבה בעבודה המסכמת. בהמשך ארחיב על אופן ביצוע האלגוריתם מבחינת תרשים זרימה.



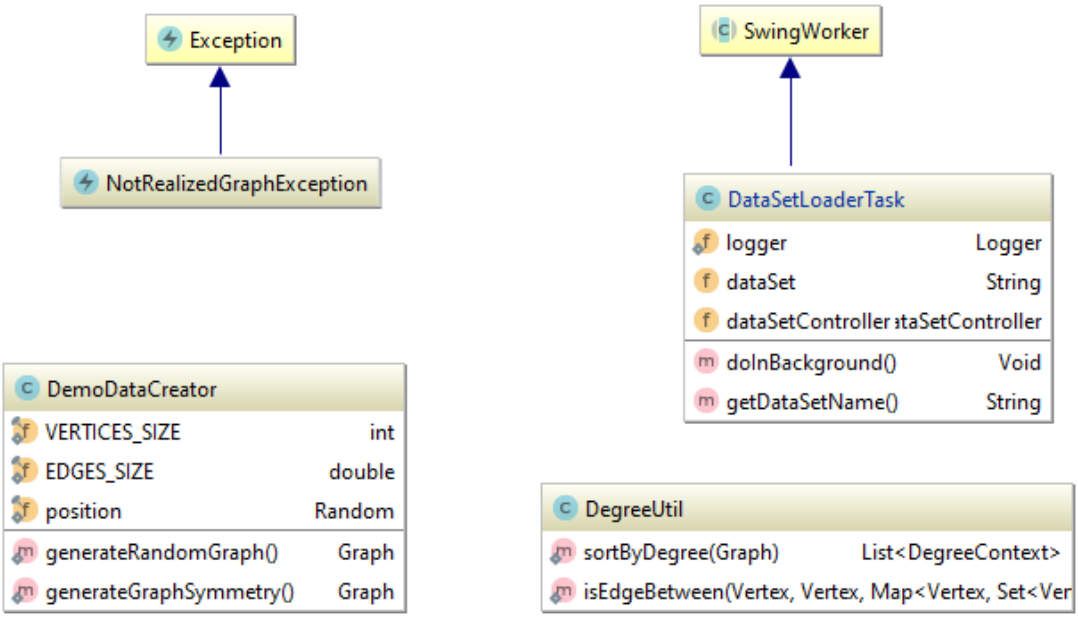
- kSymmetry – מחלקה אשר מממשת את אלגוריתם K-Symmetry שמפורט בהרחבה בעבודה המסכמת. בהמשך ארחיב על אופן ביצוע האלגוריתם מבחינת תרשים זרימה.



- StabgraphAlgorithm – מחלקה אשר מממשת את אלגוריתם Stabgraph שאלגוריתם K-Symmetry תלוי בו כחלק מבניית המודל.
- DataSetController – מחלקה ראשית של שומרת מידע לגבי אילו מאגרי מידע זמינים במערכת וטוענת אותם לזיכרון.
- FileUtil – מחלקת עזר לקריאת נתונים מקבצים לצורך טעינת מאגרי המידע.

<u>מודול UI Controller</u>

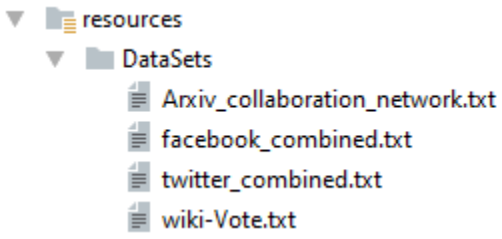
- מודול זה אחראי על חלקים שונים שהרכיבים הגרפים צריכים, לדוגמא לזרוק שגיאת מערכת, יצירת תרד׳ ברקע לביצוע עבודה מסוימת.



- NotRealizedGrpahException – מחלקה שיורשת ממחלקת Exception ונזרקת כאשר באלגוריתם K-Degree מנסים ליצר גרף בעזרת וקטור הדרגות האנונימיות אך לא מתאפשר ליצור גרף מתאים.
- DataSetLoaderTask – מחלקה שאחראית ליצור תרד׳ חדש ברקע כאשר טוענים מאגר מידע לזיכרון.
- DegreeUtil – מחלקת עזר שמחשבת מספר דברים על דרגות:
  - sortByDegree – מתודה שממיינת דרגות ברשת חברתית.
  - isEdgeBetween – מתודה שמחשבת האם קשר כלשהו נמצא בין שני צמתים שונים ברשת.
- DemoDataCreator – מחלקת עזר שנועדה ליצור מידע לצורך דמו וטסטים.
  - generateRandomGraph – מתודה שיוצרת מידע רנדומלי של רשת כלשהי.
  - generateGraphSymmetry – מתודה שיוצרת רשת חברתית לאלגוריתם K-Symmetry לפי הדוגמא מהמאמר.

<u>מודול Resources & Spring</u>

מודול ה-Resources היא תיקייה שמכילה מידע סטטי אשר ניתן לגישה מכל מחלקה במערכת. באזור זה שמורים המאגרי מידע כקבצי טקסט.



הפורמט של המאגרי מידע הוא זהה, בכל שורה בקובץ ישנם שני מספרים עם רווח ביניהם.

כל מספר מציין את השם של הצומת. השילוב של שני המספרים מתאר קשר בין שני הצמתים.

אין חשיבות לסדר המספרים מכיוון שהקשר שייווצר אינו בעל כיוון כלשהו.

מודול ה-Spring משולב במערכת על מנת ליצור שליטה במעגל החיים של האובייקטים במערכת ומתן גישה לכל אובייקט בצורה פשוטה. הקובץ שמגדיר אלו מחלקות משתתפות במעגל זה הוא beans.xml.

```
<context:annotation-config/>

<bean id="FileUtil" class="App.Common.Utils.FileUtil"/>
<bean id="DataSetController" class="App.Datasets.DataSetController"/>
<bean id="AppFrame" class="App.View.AppFrame"/>
<bean id="kdegree" class="App.Algorithm.KDegree"/>
<bean id="ksymmetry" class="App.Algorithm.KSymmetry"/>
<bean id="algorithmController" class="App.Algorithm.AlgorithmController"/>
<bean id="jNauty" class="App.lib.jNauty.McKayGraphLabelingAlgorithm"/>
<bean id="stabgraphAlgorithm" class="App.lib.jNauty.StabgraphAlgorithm"/>
```

דוגמא פשוטה ליכולת גישה בין מחלקות:

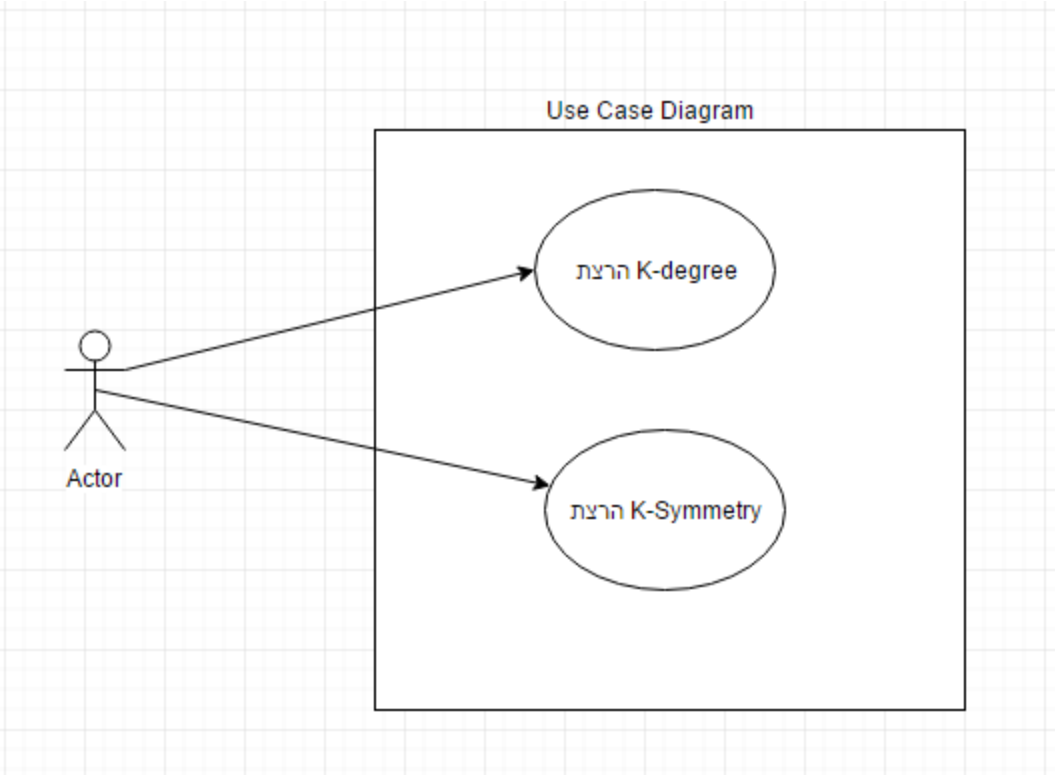
המחלקה AlgorithmController ניגשת למחלקות kDegree, kSymmetry על ידי שימוש בשדות שהוגדרו מבלי צורך לאתחל אותם, תשתית Spring דואגת שהם יאותחלו עם התלויות הרלבנטיות.

```
*/
public class AlgorithmController {

    @Autowired
    private KDegree kDegree;

    @Autowired
    private KSymmetry kSymmetry;
```

תיאור תהליכים מרכזיים

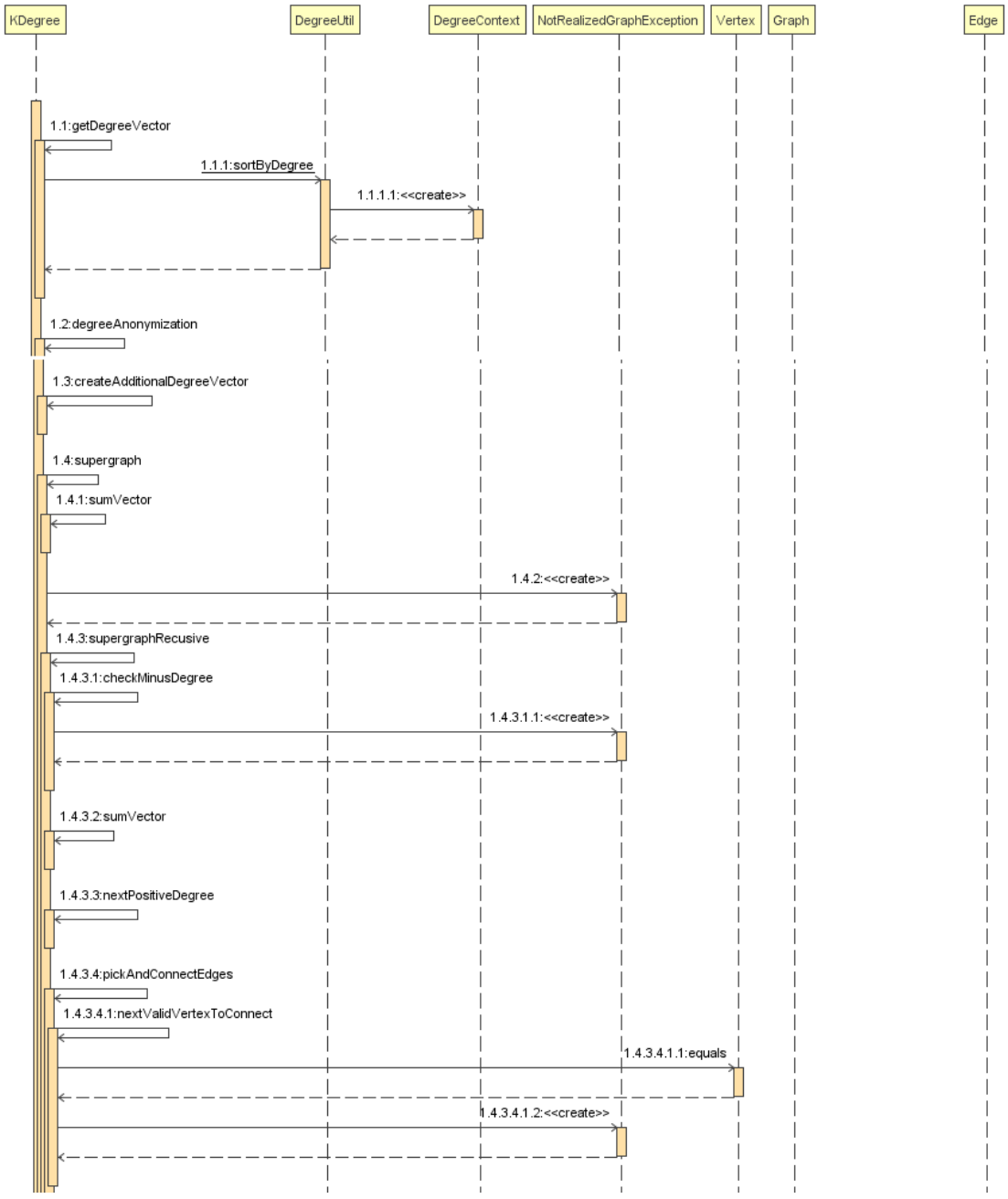
במערכת קיימים שני תהליכים מרכזיים שהמשתמש יכול לבצע:

- תהליך הרצת אלגוריתם k-Degree
- תהליך הרצת k-Symmetry


Use Case Diagram
הרצת K-degree
Actor
הרצת K-Symmetry

תהליך הרצת k-Degree

להלן תרשים זרימה של תהליך הרצת אלגוריתם k-Degree.



תיאור קוד של הפונקציה הראשית:

```java
/**
 * Main function
 *
 * @param originalGraph - original graph to anonymize
 * @param k - the K input parameter from the user
 * @return anonymized graph
 */
@Override
public Graph anonymize(Graph originalGraph, Integer k) {
    // 1. get vector of degrees descending
    DegreeContext[] originalDegrees = getDegreeVector(originalGraph);
    // 2. anonymize the degrees
    DegreeContext[] anonymizeDegreeVector = degreeAnonymization(originalDegrees, k);
    // 3. add additional edges according to the anonymize vector
    createAdditionalDegreeVector(originalDegrees, anonymizeDegreeVector);

    Graph anoymizeGraph = null;
    try {
        // 4. create sub-graph from the degrees vector
        anoymizeGraph = supergraph(originalGraph, anonymizeDegreeVector);
        // 5. return the anonymize graph
        return anoymizeGraph;
    } catch (NotRealizedGraphException e) {
        //logger.debug(e.getMessage());

        // not realized -> repeat with noise.
        // add noise to original graph and trying again
        addNoise(originalGraph);
        return anonymize(originalGraph, k);
    }
}
```

הסבר תרשים זרימה של k-degree:

שלב	חתימת הפונקציה	הסבר
1.1	`DegreeContext[] getDegreeVector(Graph originalGraph)`	הפונקציה מקבלת רשת מקורית ומחזירה רשימה של דרגות אשר ממוינות לפי סדר יורד.
1.2	`DegreeContext[] degreeAnonymization(DegreeContext[] originalDegrees, Integer k)`	הפונקציה מקבלת את וקטור הדרגות המקורי ממוין בסדר יורד ומחזירה וקטור דרגות אנונימי בהתבסס על פרמטר k. ישנו תיאור מדויק איך תהליך זה מתבצע בעבודה המסכמת.
1.3	`void createAdditionalDegreeVector(DegreeContext[] originalDegrees, DegreeContext[] anonymizeDegreeVector)`	הפונקציה מקבלת את וקטור הדרגות המקורי ואת וקטור הדרגות האנונימי ומחזירה וקטור דרגות חדש שהוא ההפרש ביניהם, שמתאר כמה קשרים יש להוסיף לרשת המקורית.
1.4	`Graph supergraph(Graph originalGraph, DegreeContext[] additionalDegreeVector) throws NotRealizedGraphException`	הפונקציה מקבלת את הגרף המקורי ואת הוקטור הדרגות האנונימי ומחזירה גרף אנונימי בהתאם או שגיאה שאי אפשר לבנות רשת כזאת בהתאם.
1.4.1	`int sumVector(DegreeContext[] additionalDegreeVector)`	בזמן יצירת רשת אנונימית בודקים האם הוקטור האנונימי בעל סכום זוגי, אחרת לא ניתן לבנות רשת מוקטור זה.
1.4.3	`Graph supergraphRecusive(Graph originalGraph, DegreeContext[] additionalDegreeVector) throws NotRealizedGraphException`	הפונקציה מקבלת את הרשת המקורית ואת וקטור הדרגות האנונימי ומחזירה רשת אנונימית בהתאם או שגיאה שלא ניתן לבנות רשת כזאת. פונקציה זו היא פונקציה רקורסיבית שבכל איטרציה יוצרים קשרים חדשים ברשת על ידי קבלת דרגה כלשהי שמהוקטור האנונימי.
1.4.3.1	`void checkMinusDegree(DegreeContext[] additionalDegreeVector) throws NotRealizedGraphException`	הפונקציה מקבלת את וקטור הדרגות האנונימי ומחזירה שגיאה במידה ויש דרגה כלשהי בעלת סכום שלילי, אחרת רק מסתיימת.
1.4.3.3	`DegreeContext nextPositiveDegree(DegreeContext[] additionalDegreeVector)`	הפונקציה מקבלת את הוקטור הדרגות האנונימי ומחזירה את הצומת בעלת דרגה עם ערך חיובי כלשהו. כאמור הוקטור ממוין באופן יורד.
1.4.3.4	`void pickAndConnectEdges(Graph originalGraph, DegreeContext degreeContext, DegreeContext[] additionalDegreeVector) throws NotRealizedGraphException`	הפונקציה מקבלת את הרשת המקורית, צומת כלשהי שיש להוסיף לה מספר מסוים של קשרים ואת הוקטור האנונימי ומחזירה שגיאה במידה ולא ניתן לבנות רשת בהתאם.

		הפונקציה בוחרת מספר צמתים לפי הדרגה שקיבלנו בכדי לחבר לצומת. מעדכנים את הוקטור האנונימי בהתאם להוספת הקשרים.
1.4.3.4.1	`DegreeContext nextValidVertexToConnect(DegreeContext[] additionalDegreeVector, DegreeContext degreeContext, Set<Vertex> vertexNeighbors) throws NotRealizedGraphException`	הפונקציה מקבלת את הוקטור האנונימי, הצומת שיש לחבר אליה קשרים ואת השכנים של אותה צומת ומחזירה צומת כלשהי שמצאנו מתאימה לחיבור או שגיאה במידה ולא ניתן. הצומת שבחרנו לחיבור נבחרת מכיוון שיש לה קשרים שצריך להוסיף לה על פי הוקטור האנונימי, והיא לא מופיעה כשכנה של הצומת שצריך להוסיף לה.

במידה ונזרקת שגיאה בשלב 1.4, אזי תופסים את השגיאה – NotRealizedGraphException מוסיפים רעש כלשהו (מוסבר בהמשך) ומנסים שוב את התהליך מההתחלה, כלומר קוראים שוב ל-anonymize.

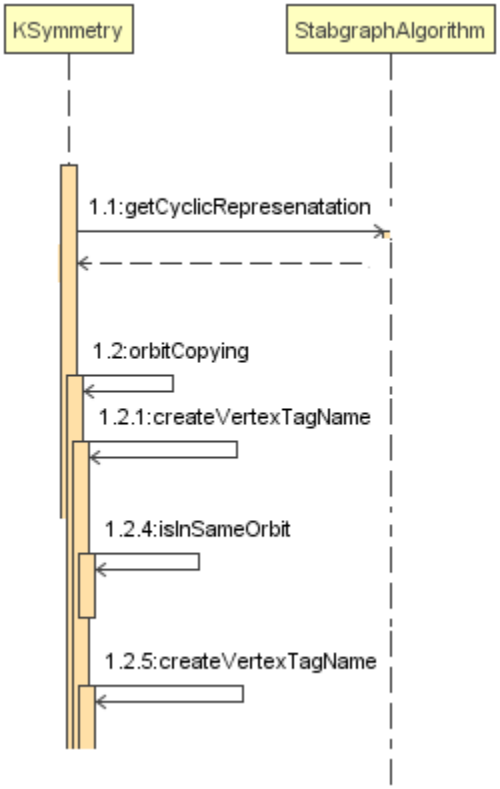
הוספת רעש לרשת המקורית: **void** addNoise(Graph originalGraph)

הפונקציה מקבלת את הרשת המקורית ומבצעת שינויים בה. הרעש במוגדר במאמר הוא על ידי בחירת מספר שלם רנדומלי קטן כלשהו, אני בחרתי מספר כלשהו עד 10, מבצעים לולאה לפי המספר שקיבלנו ובכל שלב מוספים קשר בין שני צמתים קיימים ברשת. במאמר מסבירים שמכיוון שלרוב הרשתות בנויות ממספר רב של צמתים בעלי דרגה נמוכה ורק מספר קטן יחסית של צמתים בעלי דרגה גבוהה אזי הוספת רעש כמעט ולא תתבצע או שתתבצע פעם אחת.

דוגמת הרצת K-Degree במעקב אחר צומת 2630 לחישוב Entropy

תהליך הרצת k-Symmetry

להלן תרשים זרימה של תהליך הרצת אלגוריתם k-Symmetry.



הסבר תרשים זרימה של k-Symmetry:

שלב	חתימת הפונקציה	הסבר
1.1	`List<List<Vertex>> getCyclicRepresenatation(Graph graph)` `הפונקציה נמצאת במחלקה StabgraphAlgorithm`	הפונקציה מקבלת רשת מקורית ומחזירה רשימה של רשימות של צמתים. רשימה זו היא רשימת החלוקות של הרשת.
1.2	`List<Vertex> orbitCopying(Graph graph, List<Vertex> orbit, int copyCounter)`	פונקציה שמקבלת את הרשת ורשימת האורביטים ומספר שמציין כמה העתקות כבר בוצעו. הפונקציה מבצעת שכפול האורביט בתוך הרשת.
1.2.1	`String createVertexTagName(String name, int copyCounter)`	הפונקציה מקבלת שם של צומת ואת מספר ההעתקות שבוצעו ומחזירה שם חדש. בכדי לייחד בין צומת מקורית לצומת משוכפלת אני מוסיף את מספר השכפול לשם והוספת תו"-" , כלומר vertex-5
1.2.4	`boolean isInSameOrbit(Vertex neighbor, List<Vertex> vertices, List<Vertex> orbit)`	הפונקציה מקבלת שכן כלשהו של צומת שברצוננו לבדוק האם הוא נמצא באורביט, את רשימת כל הצמתים ברשת ורשימת הצמתים באורביט ומחזירה תשובה בוליאנית האם השכן נמצא או לא באורביט.

הסבר נוסף בצורה מופשטת של האלגוריתם:

בשלב ראשון מקבלים את רשימת החלוקות של הרשת לפי אלגוריתם stabgraphAlgorithm שמומלץ לפי המאמר בעקבות ביצועים טובים יותר מאשר אלגוריתם jNauty.

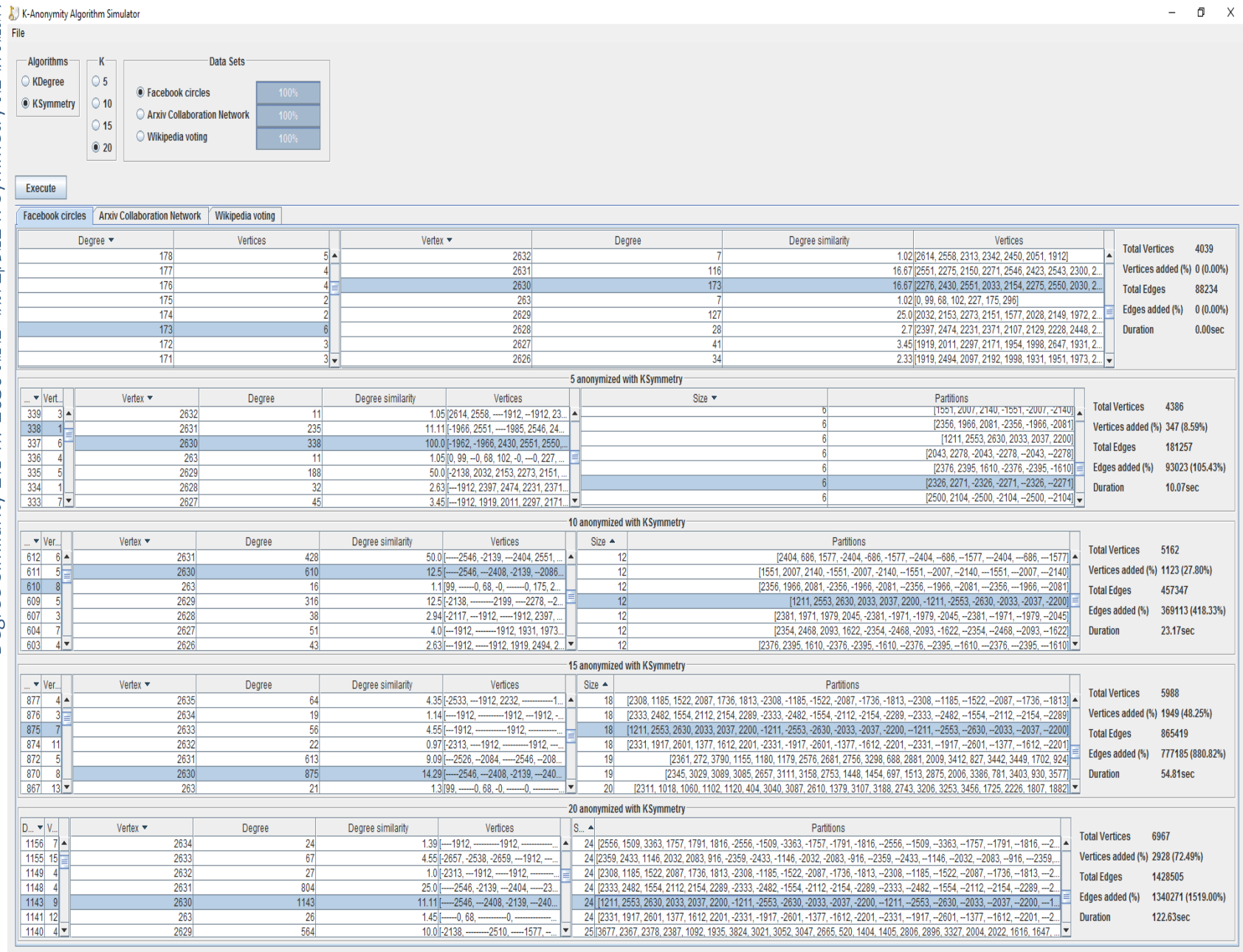
האלגוריתם מבצע מעבר על רשימת החלוקות (האורביטים), אם בחלוקה יש יותר מ-k צמתים אזי אין צורך לבצע בו שינויים ומוסיפים אותו לרשימת החלוקות האנונימיות.

במידה ויש פחות מ- k צמתים בחלוקה אזי מבצעים שכפול אורביט שמוסבר בעבודה המסכמת ומוסיפים את החלוקה לרשימת החלוקות האנונימיות.

לאחר הלולאה מחזירים את הרשת כולל הוספת ביצועי שכפולים של חלוקות.

תיאור קוד של הפונקציה הראשית:

```java
@Override
public Graph anonymize(Graph graph, Integer k) {
    // 1. fetch orbits from the graph by stabgraphAlgorithm algorithm (McKay).
    logger.debug("Start to findAutomorphisms");
    List<List<Vertex>> orbits = stabgraphAlgorithm.getCyclicRepresenatation(graph);
    if (orbits == null) {
        logger.debug("No orbits found");
        return graph;
    }
    List<List<Vertex>> anonymizedOrbits = new ArrayList<>();
    logger.debug(String.format("found %s orbits", orbits.size()));

    // 2. for each orbit -> call ocp until size at least k.
    for (int i = 0; i < orbits.size(); i++) {
        logger.debug(String.format("Iteration for orbit %s", i));

        List<Vertex> orbit = orbits.get(i);
        if (orbit.size() >= k) {
            /*for (Vertex v: orbit){
                logger.debug("Vertex not copied: " + v);
            }*/
            logger.debug(String.format("orbit %s size above k", i));
            anonymizedOrbits.add(orbit);
            continue;
        }
        // orbit size is below k, calling ocp procedure.
        int copyCounter = 1;
        while (orbit.size() < k) {
            logger.debug(String.format("Start orbitCopying for orbit %s", i));
            orbit = orbitCopying(graph, orbit, copyCounter);
            copyCounter++;
            logger.debug(String.format("Done orbitCopying for orbit %s", i));
        }
        anonymizedOrbits.add(orbit);
    }

    // 3. return the anonymized graph
    logger.debug("return the anonymized graph");
    graph.setPartitions(anonymizedOrbits);
    return graph;
}
```


K-Anonymity Algorithm Simulator
File
Algorithms
KDegree
KSymmetry
K
5
10
15
20
Data Sets
Facebook circles
Arxiv Collaboration Network
Wikipedia voting
100%
100%
100%
Execute
Facebook circles
Arxiv Collaboration Network
Wikipedia voting
Degree
Vertices
178 5
177 4
176 4
175 2
174 2
173 6
172 3
171 3
Vertex
Degree
Degree similarity
Vertices
2632 7 1.02 [2614, 2558, 2313, 2342, 2450, 2051, 1912]
2631 116 16.67 [2551, 2275, 2150, 2271, 2546, 2423, 2543, 2300, 2...
2630 173 16.67 [2276, 2430, 2551, 2033, 2154, 2275, 2550, 2030, 2...
263 7 1.02 [0, 99, 68, 102, 227, 175, 296]
2629 127 25.0 [2032, 2153, 2273, 2151, 1577, 2028, 2149, 1972, 2...
2628 28 2.7 [2397, 2474, 2231, 2371, 2107, 2129, 2228, 2448, 2...
2627 41 3.45 [1919, 2011, 2297, 2171, 1954, 1998, 2647, 1931, 2...
2626 34 2.33 [1919, 2494, 2097, 2192, 1998, 1931, 1951, 1973, 2...
Total Vertices 4039
Vertices added (%) 0 (0.00%)
Total Edges 88234
Edges added (%) 0 (0.00%)
Duration 0.00sec
5 anonymized with KSymmetry
339 3 2632 11 1.05 [2614, 2558, ----1912, --1912, 23...
338 1 2631 235 11.11 [-1966, 2551, ----1985, 2546, 24...
337 6 2630 338 100.0 [-1962, -1966, 2430, 2551, 2550,...
336 4 263 11 1.05 [0, 99, --0, 68, 102, -0, ---0, 227, ...
335 5 2629 188 50.0 [-2138, 2032, 2153, 2273, 2151, ...
334 1 2628 32 2.63 [---1912, 2397, 2474, 2231, 2371...
333 7 2627 45 3.45 [---1912, 1919, 2011, 2297, 2171...
6 [1551, 2007, 2140, -1551, -2007, -2140]
6 [2356, 1966, 2081, -2356, -1966, -2081]
6 [1211, 2553, 2630, 2033, 2037, 2200]
6 [2043, 2278, -2043, -2278, --2043, --2278]
6 [2376, 2395, 1610, -2376, -2395, -1610]
6 [2326, 2271, -2326, -2271, --2326, --2271]
6 [2500, 2104, -2500, -2104, --2500, --2104]
Total Vertices 4386
Vertices added (%) 347 (8.59%)
Total Edges 181257
Edges added (%) 93023 (105.43%)
Duration 10.07sec
10 anonymized with KSymmetry
612 6 2631 428 50.0 [----2546, -2139, ---2404, 2551, ...
611 5 2630 610 12.5 [----2546, ---2408, -2139, --2086...
610 8 263 16 1.1 [99, ------0, 68, -0, -------0, 175, 2...
609 5 2629 316 12.5 [-2138, ---------2199, ----2278, --2...
607 3 2628 38 2.94 [-2117, ---1912, -----1912, 2397, ...
604 7 2627 51 4.0 [---1912, --------1912, 1931, 1973...
603 4 2626 43 2.63 [---1912, -----1912, 1919, 2494, 2...
12 [2404, 686, 1577, -2404, -686, -1577, --2404, --686, --1577, ---1577]
12 [1551, 2007, 2140, -1551, -2007, -2140, --1551, --2007, --2140, ---2140]
12 [2356, 1966, 2081, -2356, -1966, -2081, --2356, --1966, --2081, ---2081]
12 [1211, 2553, 2630, 2033, 2037, 2200, -1211, -2553, -2630, -2033, -2037, -2200]
12 [2381, 1971, 1979, 2045, -2381, -1971, -1979, -2045, --2381, --1971, --1979, --2045]
12 [2354, 2468, 2093, 1622, -2354, -2468, -2093, -1622, --2354, --2468, --2093, --1622]
12 [2376, 2395, 1610, -2376, -2395, -1610, --2376, --2395, --1610, ---2376, ---2395, ---1610]
Total Vertices 5162
Vertices added (%) 1123 (27.80%)
Total Edges 457347
Edges added (%) 369113 (418.33%)
Duration 23.17sec
15 anonymized with KSymmetry
877 4 2635 64 4.35 [-2533, ---1912, 2232, ----------1...
876 3 2634 19 1.14 [----1912, ----------1912, ---1912, -...
875 7 2633 56 4.55 [---1912, ----------1912, ----------...
874 11 2632 22 0.97 [-2313, ----1912, ----------1912, ---...
872 5 2631 613 9.09 [---2526, --2084, -----2546, --208...
870 8 2630 875 14.29 [-----2546, ---2408, -2139, ---240...
867 13 263 21 1.3 [99, ------0, 68, -0, -------0, ----------...
18 [2308, 1185, 1522, 2087, 1736, 1813, -2308, -1185, -1522, -2087, -1736, -1813, --2308, --1185, --1522, --2087, --1736, --1813]
18 [2333, 2482, 1554, 2112, 2154, 2289, -2333, -2482, -1554, -2112, -2154, -2289, --2333, --2482, --1554, --2112, --2154, --2289]
18 [1211, 2553, 2630, 2033, 2037, 2200, -1211, -2553, -2630, -2033, -2037, -2200, --1211, --2553, --2630, --2033, --2037, --2200]
18 [2331, 1917, 2601, 1377, 1612, 2201, -2331, -1917, -2601, -1377, -1612, -2201, --2331, --1917, --2601, --1377, --1612, --2201]
19 [2361, 272, 3790, 1155, 1180, 1179, 2576, 2681, 2756, 3298, 688, 2881, 2009, 3412, 827, 3442, 3449, 1702, 924]
19 [2345, 3029, 3089, 3085, 2657, 3111, 3158, 2753, 1448, 1454, 697, 1513, 2875, 2006, 3386, 781, 3403, 930, 3577]
20 [2311, 1018, 1060, 1102, 1120, 404, 3040, 3087, 2610, 1379, 3107, 3188, 2743, 3206, 3253, 3456, 1725, 2226, 1807, 1882]
Total Vertices 5988
Vertices added (%) 1949 (48.25%)
Total Edges 865419
Edges added (%) 777185 (880.82%)
Duration 54.81sec
20 anonymized with KSymmetry
1156 7 2634 24 1.39 [----1912, ----------1912, ----------...
1155 15 2633 67 4.55 [-2657, -2538, -2659, ---1912, ---...
1149 4 2632 27 1.0 [-2313, ---1912, -----1912, ----------...
1148 4 2631 804 25.0 [-----2546, -2139, ---2404, -----23...
1143 9 2630 1143 11.11 [-----2546, ---2408, -2139, ---240...
1141 12 263 26 1.45 [------0, 68, ----------0, ----------...
1140 4 2629 564 10.0 [-2138, ---------2510, -----1577, --...
24 [2556, 1509, 3363, 1757, 1791, 1816, -2556, -1509, -3363, -1757, -1791, -1816, --2556, --1509, --3363, --1757, --1791, --1816, ---2...
24 [2359, 2433, 1146, 2032, 2083, 916, -2359, -2433, -1146, -2032, -2083, -916, --2359, --2433, --1146, --2032, --2083, --916, ---2359,...
24 [2308, 1185, 1522, 2087, 1736, 1813, -2308, -1185, -1522, -2087, -1736, -1813, --2308, --1185, --1522, --2087, --1736, --1813, ---2...
24 [2333, 2482, 1554, 2112, 2154, 2289, -2333, -2482, -1554, -2112, -2154, -2289, --2333, --2482, --1554, --2112, --2154, --2289, ---2...
24 [1211, 2553, 2630, 2033, 2037, 2200, -1211, -2553, -2630, -2033, -2037, -2200, --1211, --2553, --2630, --2033, --2037, --2200, ---1...
24 [2331, 1917, 2601, 1377, 1612, 2201, -2331, -1917, -2601, -1377, -1612, -2201, --2331, --1917, --2601, --1377, --1612, --2201, ---2...
25 [3677, 2367, 2378, 2387, 1092, 1935, 3824, 3021, 3052, 3047, 2665, 520, 1404, 1405, 2806, 2896, 3327, 2004, 2022, 1616, 1647, ...
Total Vertices 6967
Vertices added (%) 2928 (72.49%)
Total Edges 1428505
Edges added (%) 1340271 (1519.00%)
Duration 122.63sec

דוגמת הרצת K-Symmetry במעקב אחר צומת 2630 לחישוב Entropy