**האוניברסיטה הפתוחה**

**המחלקה למתמטיקה ולמדעי המחשב**

**זיהוי רשתות בוטנט בעזרת אלגוריתמי אשכול**

עבודה מסכמת זו הוגשה כחלק מהדרישות לקבלת תואר

מוסמך למדעים M.Sc. במדעי המחשב

באוניברסיטה הפתוחה

החטיבה למדעי המחשב

על-ידי

**תמיר אמינוב**

העבודה הוכנה בהדרכתה של **ד"ר מיה הרמן**

דצמבר 2017

תוכן העניינים

[1 תקציר 1](#_Toc481847201)

[2 מבנה העבודה 1](#_Toc481847202)

[3 בוטנט 1](#_Toc481847203)

[3.1 מהו בוטנט 1](#_Toc481847204)

[3.2 הפצת הבוטנט 2](#_Toc481847205)

[3.3 סוגי הבוטנט 2](#_Toc481847206)

[3.4 ארכיטקטורת הבוטנט 4](#_Toc481847207)

[3.5 זיהוי רשת בוטנט 4](#_Toc481847208)

[4 אלגוריתמי אישכול 5](#_Toc481847209)

[4.1 מבוא 5](#_Toc481847210)

[4.2 אלגוריתם K-means 8](#_Toc481847211)

[4.2.1 מבוא 8](#_Toc481847212)

[4.2.2 מטרה 8](#_Toc481847213)

[4.2.3 תיאור האלגוריתם 8](#_Toc481847214)

[4.2.4 יתרונות האלגוריתם 9](#_Toc481847215)

[4.2.5 חסרונות האלגוריתם 9](#_Toc481847216)

[4.2.6 ניתוח האלגוריתם 9](#_Toc481847217)

[4.3 אלגוריתם X-means 10](#_Toc481847218)

[4.3.1 מבוא 10](#_Toc481847219)

[4.3.2 מטרה 10](#_Toc481847220)

[4.3.3 תיאור האלגוריתם 10](#_Toc481847221)

[4.3.4 האצה של K-means 13](#_Toc481847222)

[4.3.5 אלגוריתם מסנן (Blacklisting Algorithm) 16](#_Toc481847223)

[4.3.6 האצה של שיפור המבנה 16](#_Toc481847224)

[4.3.7 האצות נוספות 16](#_Toc481847225)

[4.3.8 יתרונות האלגוריתם 17](#_Toc481847226)

[4.3.9 חסרונות האלגוריתם 17](#_Toc481847227)

[4.3.10 ניתוח האלגוריתם 17](#_Toc481847228)

[4.4 אלגוריתם אישכול היררכי 18](#_Toc481847229)

[4.4.1 מבוא 18](#_Toc481847230)

[4.4.2 מטרה 19](#_Toc481847231)

[4.4.3 תיאור האלגוריתמים 19](#_Toc481847232)

[4.4.4 יתרונות האלגוריתם 24](#_Toc481847233)

[4.4.5 חסרונות האלגוריתם 24](#_Toc481847234)

[4.4.6 ניתוח האלגוריתם 24](#_Toc481847235)

[5 ניתוח השוואתי בין שלושת משפחות האלגוריתמים 24](#_Toc481847236)

[5.1 מבוא 24](#_Toc481847237)

[5.2 הגדרת מדדים להשוואה 25](#_Toc481847238)

[5.2.1 מדדים של איכות האישכול 25](#_Toc481847239)

[5.2.2 מדדים של האלגוריתם 27](#_Toc481847240)

[5.3 תוצאות ניסוייות 28](#_Toc481847241)

[5.3.1 השוואה בין K-means לבין X-means 28](#_Toc481847242)

[5.3.2 השוואה בין K-means לבין אישכול היררכי אגלומרטיבי 32](#_Toc481847243)

[5.4 סיכום 34](#_Toc481847244)

[6 שימוש באלגוריתמי אישכול לזיהוי רשתות בוטנט 36](#_Toc481847245)

[6.1 מבוא 36](#_Toc481847246)

[6.2 אישכול התנהגותי של בוטנט מבוסס HTTP ויצירת חתימות לזיהוי בוטנט 37](#_Toc481847247)

[6.2.1 הקדמה 37](#_Toc481847248)

[6.2.2 מטרת המערכת 37](#_Toc481847249)

[6.2.3 תיאור המערכת 37](#_Toc481847250)

[6.2.4 ארכיטקטורת המערכת 37](#_Toc481847251)

[6.2.5 תוצאות ניסוייות 44](#_Toc481847252)

[6.3 זיהוי בוטנט בעזרת ניתוח לוגים ברשת ארגונית 49](#_Toc481847253)

[6.3.1 הקדמה 49](#_Toc481847254)

[6.3.2 מטרות המערכת 50](#_Toc481847255)

[6.3.3 ארכיטקטורת המערכת 50](#_Toc481847256)

[6.3.4 המידע הגולמי 50](#_Toc481847257)

[6.3.5 נורמליזציה 50](#_Toc481847258)

[6.3.6 מאפיינים 51](#_Toc481847259)

[6.3.7 אישכול 53](#_Toc481847260)

[6.3.8 תוצאות 54](#_Toc481847261)

[6.3.9 מסקנות 55](#_Toc481847262)

[6.4 מערכת BotMiner 55](#_Toc481847263)

[6.4.1 הקדמה 55](#_Toc481847264)

[6.4.2 גבולות המערכת 55](#_Toc481847265)

[6.4.3 מטרות המערכת 55](#_Toc481847266)

[6.4.4 ארכיטקטורת המערכת 56](#_Toc481847267)

[6.4.5 אישכול במישור C 56](#_Toc481847268)

[6.4.6 אישכול במישור A 59](#_Toc481847269)

[6.4.7 אישכול היררכי בין המישורים - Cross-plane Correlation 60](#_Toc481847270)

[6.4.8 תוצאות ניסוייות 61](#_Toc481847271)

[6.4.9 מסקנות 63](#_Toc481847272)

[7 סיכום 63](#_Toc481847273)

[8 המשך מחקר 64](#_Toc481847274)

[9 מקורות 65](#_Toc481847275)

# תקציר

עבודה זו עוסקת באיום גובר באינטרנט – בוטנט וזיהויו באמצעות אלגוריתמי אישכול. בוטנט היא רשת מחשבים שנשלטת על ידי אחד או כמה שרתי שליטה. המחשבים הופכים להיות חלק מבוטנט כאשר תוכנה זדונית משתלטת עליהם. בדרך כלל לא בצורה גלויה כך שהמשתמש אינו מודע למתרחש. כך, המחשבים מתחילים לקבל הוראות משרתי השליטה לביצוע משימות כמו שליחת ספאם או ביצוע מתקפת DDOS.

קיימות שיטות שונות לניטור רשת. בעבודה זו יוצגו 3 שיטות שונות שבעזרתן אפשר לזהות רשתות בוטנט:

* שיטה המזהה בוטנטים על ידי חתימות.
* שיטת ה Beehive שמנטרת רשת ארגונית.
* שיטת ה BotMiner שמזהה בוטנטים ללא תלות בארכיטקטורה או פרוטוקול תקשורת.

בכל השיטות הזיהוי מבוצע בעזרת אלגוריתמי אישכול. הקלט לאלגוריתמי האישכול הוא נתונים שמגיעים משרתי DNS של ספקית אינטרנט או נתונים שנאספו מרשת ארגונית גדולה. הנתונים חלקיים/רועשים ולכן יש לבצע עיבוד מוקדם לנתונים אלו. בעבודה בוצעה סקירה השוואתית של שלושה אלגוריתמי אישכול. האלגוריתמים הם:

K-Means, X-Means ואישכול היררכי. נשווה את האלגוריתמים לפי סיבוכיות, סקלביליות, איכות האשכול ועוד.

מסקנות העבודה הם:

1. אלגוריתמי אישכול שנסקרו בעבודה נמצאו כמאוד שימושיים בזיהוי בוטנט. כל אלגוריתם מתאים לבעיה מסוימת. לכן שלושת השיטות שהוצגו בעבודה משתמשות באלגוריתם אישכול שונה – בהתאם לדרישות ומבנה המערכת (גודל בסיס הנתונים, אופי בסיס הנתונים).
2. הבוטנטים מתפתחים במהירות רבה, לכן תמיד יש לשפר את השיטות לזיהויים או ליצור שיטות חדשות.

# מבנה העבודה

העבודה עוסקת בשני נושאים גדולים – בוטנט ואלגוריתמי אישכול. בפרק ‏3 מוצגת בעיית הבוטנט, דרכי הפצה, סוגים של בוטנטים והארכיטקטורה שלהם. כמו כן, מוסבר כיצד מזהים בוטנט. פרק ‏4 עוסק בהצגת שלושה אלגוריתמי אישכול. פרק זה מתאר ומנתח את האלגוריתמים. פרק ‏4‏5 עוסק בניתוח השוואתי בין האלגוריתמים. פרק ‏6 מציג סקירה של 3 שיטות שונות לזיהוי בוטנט בעזרת אלגוריתמי אישכול ומסתמך על השוואות ב פרק ‏4‏5. פרקים 7 ו ‏8 דנים בנושא המשך מחקר ומסכמים את העבודה.

# בוטנט

## מהו בוטנט

בוטנט (Botnet) היא רשת של רובוטים - אוסף של תוכנות המחוברות בינן לבין עצמן ברשת האינטרנט. בוטנט מתוכנת לבצע משימה אחת או מספר רב של משימות. מחשבים שהם חלק מבוטנט, הודבקו בתוכנה זדונית. המחשב מתקשר עם שרת מרוחק או מחשבים אחרים בבוטנט שקרובים אליו ומצפה מהם להוראות אשר מגיעות משרת שליטה. ארכיטקטורה זו מאפשרת לתוקף לשלוט במחשבים רבים למטרות זדוניות שלו או של אנשים אחרים שמשכירים את שירותי התוקף. התוקפים יכולים להחליט מה לעשות עם הבוטנט בכל זמן נתון. הם יכולים להשכיר את שירותיו של הבוטנט לאנשים אחרים שמעוניינים לבצע פעילות זדונית בעצמם. במקרה זה התוקפים אחראים רק להפצת הבוטנט בין כמות גדולה של מחשבים. בוטים יכולים לקבל פקודה להוריד תוכנות זדוניות נוספות כמו keylogger - תוכנה האחראית לשמירה של היסטוריית הקלדות של המשתמש – למעשה לקבל גישה לסיסמאות שלו. באיור 1 מוצג מחזור החיים של בוטנט בשלביו השונים. שלב 1: התוקף מצליח להחדיר למחשב הנתקף וירוס המשייך את המחשב לבוטנט - בתמונה מוצג כסוס טרויאני. שלב 2: המחשב שהותקף יוצר קשר עם שרת שליטה של התוקף. שלב 3: ישות הרוצה להשתמש בשירותי הבוטנט יוצרת קשר עם בעל הבוטנט למטרת שליחת ספאם. שלב 4: שרת שליטה של הבוטנט שולח למחשבי הבוטנט פקודה של שליחת ספאם והם מבצעים אותה.



איור 1. מחזור החיים של בוטנט [1].

## הפצת הבוטנט

מחשבים הופכים להיות חלק מבוטנט כאשר הם מפעילים תוכנה זדונית שאחראית לשיוך לבוטנט. הגעת התוכנה למחשב והפעלתה מתרחשת בגלל מגוון רחב של סיבות:

* הורדה והפעלה של תוכנה בטעות על ידי המשתמש. לדוגמא, מתן הרשאות להפעלה של רכיב ActiveX בדפדפן Explorer.
* פרצות אבטחה בדפדפן שמאפשרות הרצת קוד זדוני שמוריד ומפעיל קובץ הבוטנט על המחשב.
* פרצות אבטחה בתוכנות ומערכות הפעלה היכולות לאפשר להריץ קוד זדוני שהופך את המחשב לבוט.
* הנדסת אנוש (Social Engineering) – התוקף יכול לגרום במרמה למשתמש לבצע פעולה שהוא רוצה. לדוגמא השארת דיסק USB בגן ילדים. הילד של המשתמש יראה את ה USB וירצה לבדוק את תכולתו במחשב של הוריו. ההתקן של ה USB מתחבר למחשב אך מזדהה כמקלדת שמקלידה במהירות גדולה פקודות שמורידות את התוכנה הזדונית למחשב מהאינטרנט או מ USB עצמו.

## סוגי הבוטנט

נחלק את הבוטנט לשתי קטגוריות: חוקי ובלתי חוקי. בהמשך בעבודה נתייחס לבוטנט כאל בוטנט זדוני לא חוקי.

בוטנט חוקי משמש לצורך עשיית פעולות אוטומטיות או חישוביות:

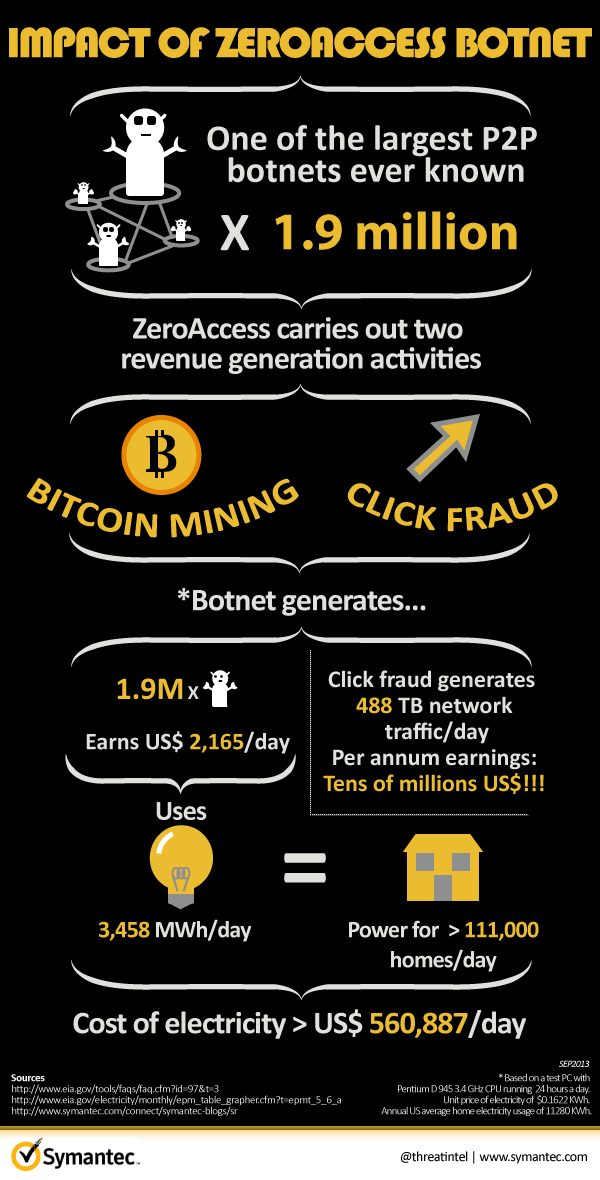
* בניהול רשת IRC - פרסום בצ'אט, זריקת (Kick) משתמשים מחדרי הצ'אט.
* ביצוע מחקר רפואי – תוכנה שמבצעת חישובים בנושאים כמו חקר הסרטן או הגנום האנושי. שרת שליטה מפצל משימות חישוביות לאלפי מחשבים כדי שאלה יבצעו חישובים מורכבים כשהמשתמש לא משתמש במחשב.
* חברות שמוכרות כוח חישובי – ארגונים שונים יכולים להשכיר כוח חישובי תמורת תשלום.

בוטנט לא חוקי משמש להשתלטות על מחשב זר וביצוע פעולות זדוניות בשם מחשב זה. דוגמא לפעולות:

* DDOS – Distributed Denial of Service attack – אלפי בוטים יוצרים בקשות אל יעד אחד וגורמים להפרעות רבות בפעילות של היעד.
* פרסומים Adware - פרסום מודעות ללא רשות או ידיעה של המשתמש. המודעות יכולות להופיע במקומות שונים – בדרך כלל יחליפו מודעות אינטרנטיות אחרות באתרים אליהם המשתמש גולש.
* רוגלות (Spyware) – תוכנות אלו שולחות נתונים רגישים של המשתמשים לתוקפים. נתונים כמו סיסמאות, מספרי כרטיסי אשראי, מידע פנימי של ארגונים.
* ספאם (Spam) – שליחת ספאם מחשבונות של אנשים למטרת פרסום או הפצת בוטנט.
* לחיצות פרסום (Click Fraud) –כניסה לאתרים כדי להעלות רייטינג או לחיצות על פרסומות בשביל כסף.
* תולעים (Worms) – דרך להפיץ את הבוטנט למחשבים אחרים דרך פרצות אבטחה במחשבים.
* Fast Flux – טכניקה שמאפשרת להחביא אתרים או שרתים, כאשר המחשב הופך להיות שרת Proxy.
* Brute Force – ביצוע מתקפות brute force נגד אתרים או סיסמאות אחרות.
* Scareware – תוכנות שמטרתן היא להפחיד את המשתמש ולקנות מוצר.
* פעולות חישוביות – כריית ביטקוין או כל פעולה חישובית לא רצונית.



איור 2. Scareware שמצפין דיסק קשיח ומבקש כסף מהמשתמש על מנת להחזיר את הקבצים [1].



איור 3. דוגמא לבוטנט שעוסק בכריית ביטקוין [1].

## ארכיטקטורת הבוטנט

בוטנט מורכב מקבוצה של בוטים – מחשבים שהשתלטו עליהם ושרת(י) שליטה. המפעיל של הבוטנט קובע פעולה זדונית, זמן התרחשות וגודל הפעולה. הפקודה עוברת מהמפעיל לשרתי שליטה ולאחר מכן לבוטים. בבוטנט יכול להיות שרת שליטה בודד אך זה פוגע בשרידותו לכן הבוטים נוטים להיות עם כמה שרתי שליטה מתחלפים. חלק מהבוטנטים יוצרים רשת p2p בין הבוטים שלהם כך שקשה לקבוע מי הוא השרת שליטה, כיוון שהפקודות יכולות להגיע מכל מקום וכל בוט למעשה הופך להיות שרת שליטה קטן או מעביר הודעות של שרת שליטה.

## זיהוי רשת בוטנט

זיהוי בוטנט מתבצע בהרבה שיטות. לכל אחת מהשיטות יש יתרונות וחסרונות מול שיטה אחרת. להלן שיטות לזיהוי רשת בוטנט:

1. זיהוי התקשרות של מחשב אל שרת בוטנט הידוע כשרת שליטה .
2. זיהוי התנהגות זדונית של המחשב כלפי מחשבים אחרים (שליחת ספאם או מתקפות DDOS).
3. ניתוח ולימוד תבניות התקשרות של המחשבים.
   1. זיהוי בוטנט מסוג DGA[[1]](#footnote-2) בעזרת התקשרויות NXDomain [2] [3].
   2. זיהוי בוטנט בעזרת ניתוח ההתקשרות [4].
4. מציאת תוכנה זדונית על המחשב בעזרת אנטי וירוס [5].
5. מציאת התקשרות זדונית או התקשרות לשרת שליטה בעזרת חתימות [5] [6].

בעבודה זו יוצגו מספר שיטות לזיהוי בוטנטים המבוססות על אלגוריתמי אישכול.

# אלגוריתמי אישכול

## מבוא

כריית מידע הוא תהליך של זיהוי\גילוי תבניות ויחסים שימושיים וענייניים בנפחי מידע גדולים במטרה להוציא ידע נחוץ מתוך כמויות אדירות של נתונים גולמיים. כריית מידע שימושית בתחום העסקי (ביטוח, בנקאות, קמעונאות), מדעי (אסטרונומיה ורפואה) ובטחוני (זיהוי עבריינים וטרוריסטים).כריית מידע מהווה שלב חשוב ביותר בתהליך גדול יותר – גילוי ידע בבסיסי נתונים (Knowledge Discovery from Data). גילוי ידע מורכב ממספר שלבים כמודגם באיור 4:

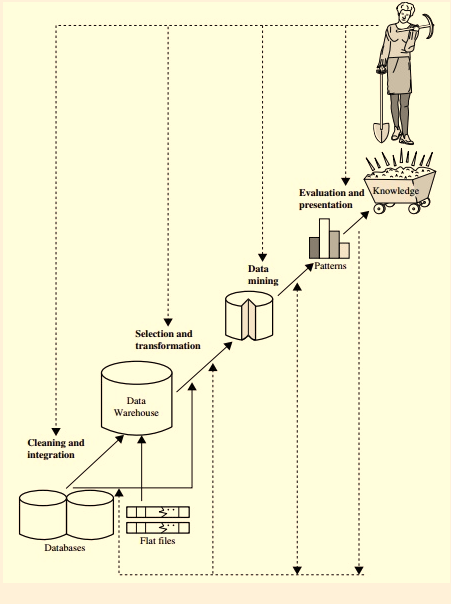
1. טיוב נתונים:

1. ניקוי נתונים – טיפול בנתונים רועשים.
2. אינטגרציה של מידע - צירוף מקורות מידע שונים לכלל DW[[2]](#footnote-3).
3. מיון מידע רלוונטי לכריית המידע.
4. המרה או איחוד מידע למבנה נתונים ההולך לכרייה.

2. כריית מידע. שימוש באלגוריתמי כריית מידע לצורך זיהוי וחיזוי, אשכול וכו'.

3. ניתוח תוצאות כריית המידע.

4. הצגת תוצאות כריית המידע והצגת המסקנות.



איור 4. שלבי תהליך גילוי ידע [7].

**אשכוֹל** (cluster)היא קבוצת רשומות כך שאיברי האשכולות דומים זה לזה, בעוד האשכולות עצמם שונים זה מזה.

סוגי נתונים בניתוח אשכולות

אלגוריתמי אישכול נפוצים משתמשים בד"כ באחד משני המבנים הבאים לייצוג הנתונים:

* **מטריצת נתונים**(Data Matrix) שמייצגת n אובייקטים עם p תכונות (משתנים). לדוגמא אובייקטים מייצגים אנשים ותכונות מייצגות גיל, גובה, משקל וכו.
* **מטריצת אי-דמיון** (Dissimilarity Matrix), אשר בה הערכים מייצגים את המרחק (מידת אי-דמיון) בין זוג אובייקטים.

ניתן לעבור מייצוג נתונים בעזרת מטריצת הנתונים לייצוג בעזרת מטריצת אי-דמיון לפני הרצת אלגוריתם אשכול.

חישוב מידת אי-דמיון בין אובייקטים.

החישוב תלוי בסוג המשתנים (תכונות). ניתן לחלק את המשתנים לסוגים הבאים:

* רציפים. לדוגמא רמת הכנסה, לחץ דם.
* בינאריים. לדוגמא הריון/לא הריון, עבר/לא עבר.
* נומינליים. זוהי הכללה של משתנים בינאריים. משתנים מסוג זה עשויים לכלול יותר משני מצבים כדוגמת אדום, צהוב, כחול, ירוק.
* אורדינליים. הסדר היחסי בין ערכי המשתנה האורדינלי יותר חשוב מגודל של אותם הערכים.

לדוגמא בספורט יותר חשוב הדירוג זהב-כסף-ארד מאשר הערך הממשי של דרגה מסוימת.

* משתנים של יחס. לדוגמא קצב למידה.

ברוב מאגרי מידע האובייקטים מתוארים ע"י משתנים מסוגים שונים.

נתאר שלבי חישוב מידת אי דמיון עבור משתנים רציפים:

1. סטנדרטיזציה - מנסה לתת לכל משתנה משקל שווה. שלב זה לא הכרחי, כי לפעמים דווקא נרצה לתת למשתנה מסוים משקל יותר גבוה.

נניח יש n מדידות של משתנה f עבור אובייקט i: :

נחשב mean absolute deviation שמבטא מרחק אבסולוטי ממוצע בין כל מדידה לממוצע המדידות. יסומן כ .

|  |  |  |
| --- | --- | --- |
|  | כאשר | (4.1) |

1. חישוב מרחק בין כל שני אובייקטים n-ממדיים ו בעזרת פונקציית מרחק. משתמשים בד"כ באחת משלושת פונקציות המרחק הבאות:

* מרחק אוקלידי

|  |  |  |
| --- | --- | --- |
|  |  | (4.2) |

* מרחק מנהטן

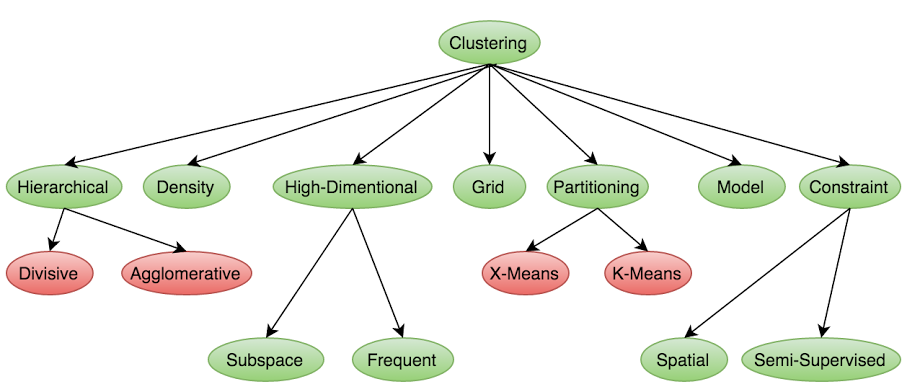
|  |  |  |
| --- | --- | --- |
|  |  | (4.3) |

* מרחק מינקובסקי

|  |  |  |
| --- | --- | --- |
|  |  | (4.4) |

כאשר p הוא מספר חיובי.

באופן כללי רוב שיטות האישכול מתחלקות לקטגוריות המפורטות באיור 5. בהמשך העבודה נתמקד בשתי קבוצות עיקריות של אלגוריתמי אישכול - אלגוריתמי חלוקה ואלגוריתמים היררכיים:



איור 5. סיווג אלגוריתמי אישכול [7].

## אלגוריתם K-means

### מבוא

בעיית החלוקה של n אובייקטים ל-k אשכולות, כך שכל אובייקט שייך לאשכול בעל מרכז כובד הכי קרוב לאותו האובייקט, היא בעיה NP-קשה. עם זאת קיימים אלגוריתמים היוריסטיים שמתכנסים מהר לאופטימום לוקאלי. אחד האלגוריתמים הוא K-means שמייצג כל אשכול בתור ממוצע של אובייקטים באשכול.

### מטרה

נתון בסיס נתונים של N אובייקטים. מעוניינים לבצע אשכול בהתאם לעקרונות הבאים:

1. כל אשכול חייב להכיל לפחות אובייקט אחד.
2. כל אובייקט חייב להשתייך לאשכול אחד בדיוק.
3. קיימת מידת דמיון גבוהה בין האובייקטים בתוך אותו אשכול (קריטריון הקומפקטיות).
4. מידת הדמיון בין האובייקטים באשכולות שונים היא נמוכה (קריטריון ההפרדה).

המטרה היא למצוא חלקוה של N אובייקטים ל *K* אשכולות, *כך ש K האשכולות יהיו כמה שיותר קומפקטיים וכמה שיותר מופרדים אחד מהשני*.

### תיאור האלגוריתם

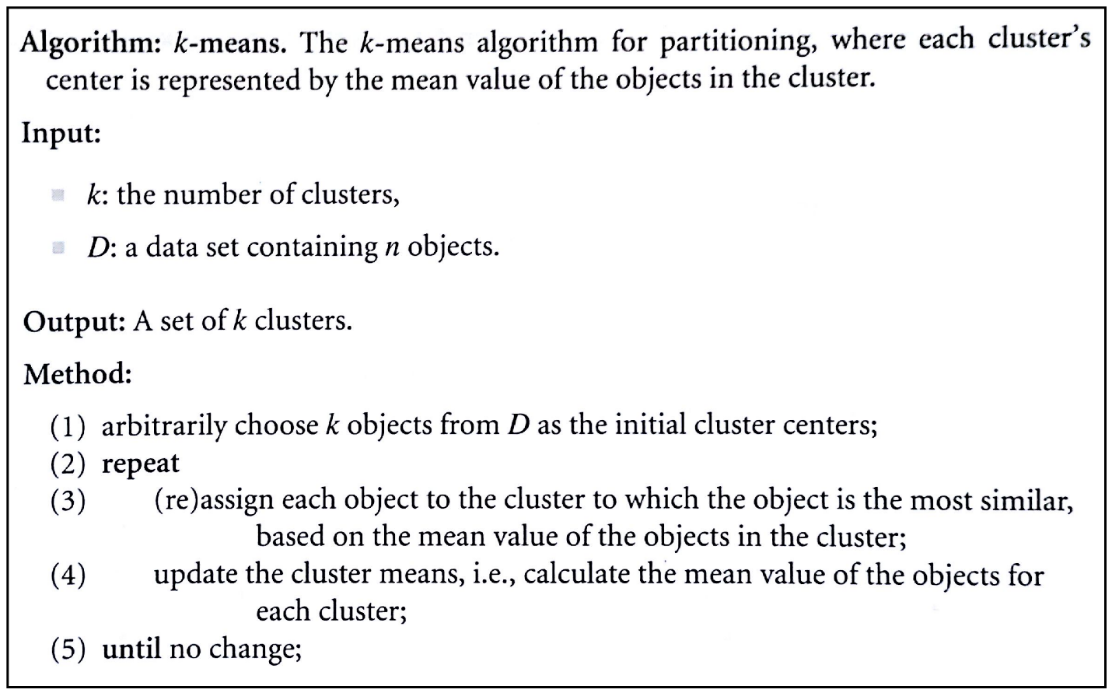
בהתחלה יש לאתחל את האשכולות ההתחלתיים.

שיטות אתחול

נציין 2 שיטות אתחול נפוצות:

* פורגי (Forgy) – בוחרים רנדומלית K אובייקטים מהמאגר נתונים ומשתמשים בהם בתור מרכזי כובד התחלתיים.
* חלוקה אקראית (Random Partition) – מקצים רנדומלית אשכול כלשהו לכל אובייקט ואז עוברים לשלב העדכון**.** מרכז הכובד הראשון של האשכול יהיה ממוצע אריתמטי של אובייקטים שנבחרו אקראית לאותו אשכול.

פסאודוקוד של אלגוריתם K-means:



אלגוריתם 1. אלגוריתם K-Means [7].

נסביר את האלגוריתם:

שורה 1: בוחרים רנדומלית K אובייקטים ומשתמשים בהם בתור מרכזי כובד התחלתיים (אתחול לפי פורגי).

שורה 3: כל אובייקט מתוך האובייקטים שנשארו משויך למרכז כובד שהכי קרוב אליו, בהתבסס על מרחק בין מרכז כובד לאותו האובייקט. אובייקט p משויך לאשכול , מכיוון שריבוע מרחק בינו לבין (מרכז כובד של ) הוא מינימלי:

|  |  |  |
| --- | --- | --- |
|  |  | (4.5) |

שורה 4: מעדכנים מרכז כובד חדש עבור כל אשכול:

|  |  |  |
| --- | --- | --- |
|  |  | (4.6) |

*שורה 5:* תהליך זה חוזר על עצמו עד שאין שינוי בהרכב האשכולות או עד שפונקציית הקריטריון מתכנסת לאופטימום לוקאלי. בד"כ משתמשים בקריטריון של שגיאה ריבועית [7] .

### יתרונות האלגוריתם

1. אלגוריתם פשוט.

2. האלגוריתם מתכנס מהר אם האשכולות הסופיים הם בעלי צורה ספרית – קומפקטיים ונפרדים אחד מהשני.

3. האלגוריתם מתאים לבסיסי נתונים גדולים.

### חסרונות האלגוריתם

1. מתכנס לאופטימום לוקאלי בד"כ, ולא לאופטימום גלובלי.
2. חייבים לספק את מספר האשכולות *K*.
3. כשמוגבלים לערך מקובע של *K*, החיפוש של אשכולות יכול להתכנס לאופטימום לוקאלי פחות טוב מזה שיכולנו לקבל בבחירה דינאמית של *K*.

### ניתוח האלגוריתם

נגדיר:

*N* – מספר כולל של אובייקטים.

*K* – מספר אשכולות.

*t* – מספר איטרציות.

*M* – מספר ממדים.

בדרך כלל ו

**סיבוכיות זמן** [7]

בכל איטרציה עבור כל מרכז כובד יש לחשב מרחק ממנו לכל האובייקטים.

**סיבוכיות מקום**

בכל איטרציה יש לשמור את כל האובייקטים ומרכזי הכובד.

בדרך כלל האלגוריתם מאוד מהיר, נהוג להריץ אותו כמה פעמים עם תנאים התחלתיים שונים, כדי לקבל תוצאה אופטימלית.

## אלגוריתם X-means

### מבוא

X-means מחשב את K בצורה דינאמית תוך כדי שימוש ב K-means. בנוסף X-means משתמש ב cached statistics כדי לחסוך בחישובים שיש ב K-means. ולכן X-means יעיל יותר עבור מאגרי מידע עם מספר גדול של רשומות.

### מטרה

המטרה היא לפתור את החסרונות הבולטים שיש ב K-means (פורטו בסעיף "חסרונות" בפרק הקודם) ולשפר זמני ריצה וסקלביליות של האלגוריתם.

### תיאור האלגוריתם

קלט:

טווח ערכי K אפשריים

*D* – קובץ המכיל *N* אובייקטים.

האלגוריתם מורכב משלבים הבאים:

1. שיפור הפרמטרים.
2. שיפור המבנה.
3. אם עצור, אחרת חזור לצעד 1.

בשלב **שיפור הפרמטרים** מתבצעת הרצת K-means **מואץ** (נסביר בהמשך הפרק) על רשימת מרכזי כובד נוכחית עד להתכנסות.

שלב **שיפור המבנה**  קובע האם יש להגדיר מרכז כובד חדש והיכן הוא ימצא.

X-means מתחיל מK (מספר אשכולות) המהווה גבול תחתון וממשיך בהוספת מרכזי כובד עד שמגיע לגבול עליון . תוך כדי התהליך, קבוצת מרכזי הכובד שקיבלה ציון מרבי נשמרת ומהווה פלט של האלגוריתם.

נתאר שתי גישות ל**שיפור המבנה**:

**גישה 1:**

עבור כל מרכז כובד במבנה הנוכחי:

1. ניצור מרכז כובד חדש לידו.
2. נריץ K-means עד להתכנסות ונבדוק אם שיפרנו את המודל.
3. אם כן שיפרנו את המודל, נוסיף את מרכז הכובד לקבוצת מרכזי הכובד הקיימים. אם לא שיפרנו, נחזור למבנה הקודם.

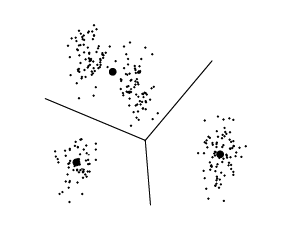
כפי שניתן לראות גישה זו דורשת צעדים של שיפור מבנה עד ש X-means יתכנס. מכיוון שכל טסט מצריך הרצת K-means, הוספת מרכז כובד אחד הופכת ליקרה מאוד.

**גישה 2:**

נבחר חצי ממרכזי הכובד בהתאם לקריטריון היוריסטי שקובע את היכולת הפוטנציאלית של אותם מרכזי הכובד להתפצל. נפצל את אותם מרכזי הכובד, נריץ את K-means ונבדוק אם שיפרנו את המודל. אם כן, נקבל את הפיצול. זוהי שיטה הרבה יותר אגרסיבית שדורשת צעדי **שיפור המבנה** עד ש X-means מסתיים.

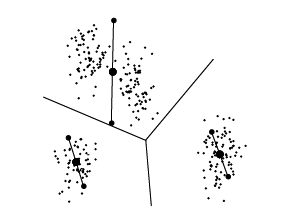
X-means מנצל יתרונות של שני הרעיונות ונמנע מהחסרונות שלהם.

נדגים את שלב שיפור המבנה של X-means:



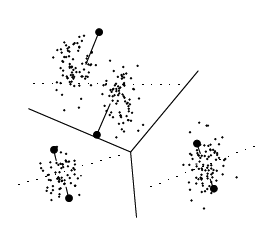
איור 6. תוצאת ריצת K-means עם 3 מרכזי כובד [8].

איור 6 מראה פתרון K-means יציב עם 3 מרכזי כובד. ניתן לראות גם גבולות האזורים אליהם שייכים מרכזי הכובד.



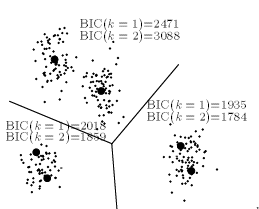
איור 7. כל מרכז כובד מקורי מתפצל ל-2 ילדים [8].

שיפור המבנה מתחיל בפיצול של כל מרכז כובד לשני ילדים (איור 7). שני הילדים נמצאים בכיוונים מנוגדים לאורך וקטור שנבחר רנדומלית ומרחק ביניהם פרופורציונלי לגודל האזור בו הם נמצאים. בכל אזור מריצים K-means לוקאלי (עם K=2) עבור כל זוג ילדים.



איור 8. צעד ראשון של ריצה מקבילית של 2-means. הקו שיוצא מכל מרכז כובד מראה לאן אותו מרכז הכובד זז [8].

איור 8 מראה צעד ראשון של שלושת ריצות לוקאליות של 2-means.



איור 9. תוצאת ריצת 2-means לוקאליים [8].

איור 9 מראה מיקום של הילדים אחרי שריצות 2-means לוקאליות הסתיימו. בנקודה זו יש להחליט עבור כל אזור לוקאלי האם שיפרנו את המודל שלו. משתמשים בקריטריון BIC כדי למדוד טיב המודל.

קריטריון BIC

נניח שנתון לנו קובץ מידע D ומשפחה של מודלים אפשריים , כאשר במקרה שלנו מודלים שונים מתייחסים לפתרונות עם ערכי K (מספר אשכולות) שונים. איך נבחר במודל הכי טוב? נשתמש בקריטריון BIC(Bayesian Information Criterion) או Schwarz Criterion – קריטריון שוורץ [8]:

|  |  |  |
| --- | --- | --- |
|  |  | (4.7) |

זהו קריטריון לבחירת מודל מתוך קבוצה סופית של מודלים. הקריטריון מבוסס על פונקציית נראות.

*R* – מספר תצפיות ב-D, שקול לגודל המדגם.

– לוג-נראות[[3]](#footnote-4) של התצפיות במודל ה-j בנקודה שממקסמת את פונקציית הנראות (אומד הנראות המרבית).

במקרה שלנו כל המודלים הם מסוג שמונח ע"י K-means (גאוסיאן כדורי[[4]](#footnote-5)). אומד הנראות המרבית (MLE) תחת הנחה של גאוסיאן כדורי [8]:

|  |  |  |
| --- | --- | --- |
|  |  | (4.8) |

כאן הוא ממוצע של האשכול אליו שייך .

לוג-נראות מקסימלית של התצפיות [8]:

|  |  |  |
| --- | --- | --- |
|  |  | (4.9) |

כאן הוא גודל האשכול ה n. .

– מספר פרמטרים חופשיים ב - . הפרמטרים החופשיים:

* הסתברויות של שייכות לאשכול.
* (M - מספר ממדים) קואורדינטות של מרכזי הכובד.
* אומד הנראות המרבית.

נשתמש בנוסחת BIC גלובאלית, כאשר X-means בוחר במודל הכי טוב שהוא מצא וגם לוקאלית בכל הטסטים של פיצול מרכזי כובד.



איור 10. מודל שקיבל ציון BIC מקסימלי [8].

איור 10 מראה את המודל הסופי שהתקבל אחרי השוואת ערכי BIC.

### האצה של K-means

נתרכז באיטרציית K-means יחידה. המטרה שלנו לקבוע עבור כל אובייקט לאיזה מרכז כובד הוא שייך. שלב **שיפור הפרמטרים** מאפשר לשייך בבת אחת קבוצה של אובייקטים למרכז כובד כלשהו ללא בדיקת שייכות של כל אובייקט בנפרד.

נתאר את פרוצדורת **שיפור הפרמטרים**:**(Update)**

נשמור אובייקטים (בצורת נקודות במרחב) בגרסה מיוחדת של kd-tree[[5]](#footnote-6) שנקראת mrkd-tree (multi-resolution kd-tree). תכונות של mrkd-tree [9]:

* עץ בינארי.
* כל צומת מכילה מידע על נקודות המצויות בה. h הוא היפר-מלבן מינימלי שתוחם את נקודות הצומת. h מוצג ע"י שני וקטורים M - ממדיים ו-. בנוסף בכל צומת שומרים את מספר הנקודות המצויות בתוך ההיפר-מלבן וסכום ווקטורי של הנקודות. כל הילדים של הצומת הם היפר-מלבנים שמוכלים בתוך h.
* לכל צומת שהוא לא עלה יש ממד פיצול d וערך פיצול v. הילדים של הצומת מוצגים ע"י היפר-מלבנים () - הנקודות של שניהם נמצאות בתוך h כך שלכל נקודה ב הקואורדינטה ה-d שלה קטנה יותר מ-v.
* צומת שורש מכיל היפר-מלבן המכיל את כל הנקודות.
* כל עלה בעץ מכיל בדיוק נקודה אחת.

נגדיר כמה מושגים:

עבור נקודה כלשהי והיפר-מלבן h נגדיר כנקודה ב-h הקרובה ביותר ל-.

**חישוב עורך זמן:** כאשר מצוי ב h הוא ה-closest. כאשר ה-מצוי על הגבולות של h הרי שניתן למצוא אותו ע"י שינוי קואורדינטות של , כך שיהיו בתוך h [10].

נגדיר מרחק להיות .

בהינתן קבוצת מרכזי כובד C והיפר-מלבן h, נגדיר להיות מרכז כובד כך שכל נקודה ב-h קרובה ל-c יותר מכל מרכז כובד אחר ב-C. לכן אם מרכז כובד כזה קיים, הוא יהיה ה-owner היחיד של h.

משפט 1

*בעת חיפוש , יש להתחשב רק במרכזי כובד בעלי מרחק מינימלי.*

נניח C היא קבוצה של מרכזי כובד ו-h הוא היפר-מלבן. נניח הוא . אז:

|  |  |  |
| --- | --- | --- |
|  |  | (4.10) |

*נניח בשלילה ש . אז קיימת נקודה ב-h שיותר קרובה ל- מאשר ל-c. הגענו לסתירה עם ההגדרה של c בתור .*

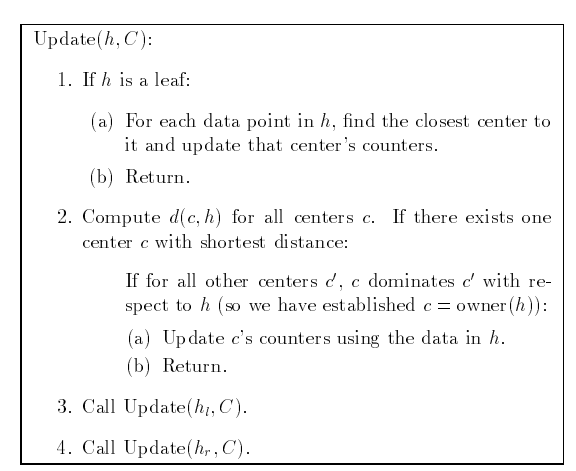
Domination

בהינתן היפר-מלבן h ושני מרכזי כובד ו- כך ש נגדיר ש דומיננטי על ביחס ל-h אם כל נקודה ב-h קרובה ל- יותר מ-.

**סיבוכיות זמן** של בעיית הכרעה "האם דומיננטי על ביחס לh-?" היא [10].

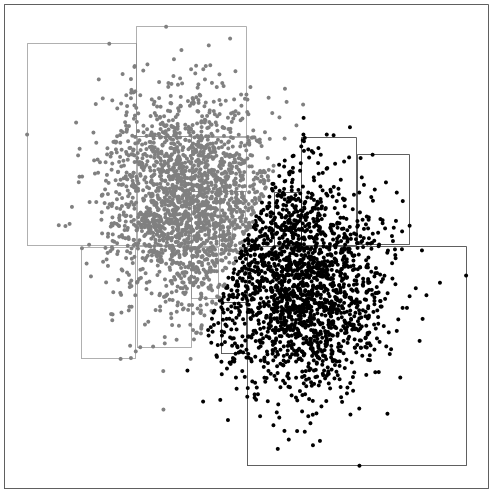
נתאר את האלגוריתם הפשוט:

נגדיר קבוצה של מונים – מונה אחד לכל מרכז כובד. כל מונה מחזיק מספר כולל של נקודות ששייכות לאותו מרכז הכובד. בנוסף כל מונה מחזיק סכום ווקטורי של הנקודות. הקלט ההתחלתי לפרוצדורה הרקורסיבית הוא היפר-מלבן שמכיל את כל הנקודות. תנאי עצירה מתקיים כאשר כל הנקודות בהיפר-מלבן שייכות לאותו מרכז כובד.



איור 11. פרוצדורת Update שמאיצה K-means בעזרת מעבר על [10] mrkd-tree.

אם הפרוצדורה מגיעה לעלים, היא תיקח אפילו יותר זמן מ K-means, לכן ההנחה כאן שהיפר-מלבנים מספיק גדולים יהיו שייכים למרכז כובד יחיד. איור 12 מדגים את הפרוצדורה.



איור 12. ויזואליזציה של היפר-מלבנים ששייכים למרכזי כובד [10].

באיור 12 קובץ מידע דו-ממדי מוצג כנקודות במישור. כל הנקודות ששייכות למרכז כובד מסוים צבועות באותו צבע (כאן K=2). ניתן לראות שלבים של יצירת צמתים בעץ kd. כל עוד לא נמצא owner להיפר-מלבן, הפרוצדורה חילקה את ההיפר-מלבן הנתון ל-2 היפר-מלבנים מינימליים חדשים. האיור מציג היפר-מלבנים שקיימו תנאי עצירה ונמצא owner (מרכז כובד) עבורם.

נקודות מחוץ להיפר-מלבנים עברו מיון בשיטה הרגילה (ע"י חישוב מרחק מכל נקודה לכל מרכז כובד) [10].

בכל צומת נשמר סכום ווקטורי של נקודות שנמצאות בהיפר-מלבן של הצומת ומספר של אותן הנקודות.

### אלגוריתם מסנן (Blacklisting Algorithm)

הרעיון לזהות את אותם מרכזי הכובד שבטוח לא יכולים להיות ה-owner-ים של היפר-מלבן h.

נניח הוא מרכז הכובד בעל מרחק מינימלי מ-h, ונניח הוא מרכז כובד כלשהו כך ש

. אם דומיננטי על ביחס ל-h, ישנן שתי אפשרויות. אחת ש . זוהי אפשרות טובה כי במקרה כזה לא דרוש שום חישוב נוסף. אפשרות אחרת שאין עבור הצומת. האלגוריתם הפשוט היה מתחיל חישוב מחדש עבור הילדים של h. הגרסה המסננת מזהה ש דומיננטי על ביחס ל- עבור כל שמוכל ב-h (הדבר נכון לפי הגדרה). מכיוון שכל הצאצאים של h ב mrkd-tree מוכלים ב-h, אנחנו יכולים להוציא את מרשימה של מרכזי כובד אפשריים עבור כל הצאצאים של h.

### האצה של שיפור המבנה

נשתמש ב K-means מואץ גם בשלב **שיפור המבנה**.

1. נריץ שלב **שיפור הפרמטרים** על רשימת מרכזי כובד נוכחית.
2. כשצומת מסוים בעץ ישויך למרכז כובד כלשהו:

2.1 נפצל את מרכז הכובד כפי שתיארנו למעלה בשלב **שיפור המבנה**.

2.2 נעביר את שני מרכזי הכובד שהתקבלו ביחד עם הצומת הנוכחי כקלט לפרוצדורת **שיפור הפרמטרים**.

### האצות נוספות

אחסון סטטיסטיקות מאיטרציות קודמות במטמון.

עבור כל צומת, נאחסן במטמון את התרומה של נקודות הצומת למונים של מרכזי הכובד. לכן, אם הרשימה של מרכזי הכובד הפוטנציאליים של הצומת לא השתנתה מאיטרציה קודמת, אין צורך לרדת עמוק יותר מהצומת הנוכחית בעץ.

השוואה יעילה בין מיקומים נוכחיים של מרכזי הכובד לבין מיקומים אילו באיטרציה קודמת תבוצע באופן הבא: נצטרך מקום וטבלת גיבוב לחיפוש מזהים של מרכזי הכובד. אם המיקום של מרכז כובד השתנה, נוצר מזהה חדש ייחודי ומזהה "ישן" נמחק. בדרך זו נוכל להשוות בין מרכזי הכובד ב זמן. ההשוואה תבוצע בין מזהים בלבד.

### יתרונות האלגוריתם

1. X-means מנסה להימנע מהתכנסות לאופטימום לוקאלי [8].
2. סקלביליות.
3. זמן ריצה טוב יותר מזמן ריצה של K-means בד"כ.

### חסרונות האלגוריתם

אם X-means מגיע לעלים, זמן הריצה שלו גדול יותר מזמן ריצה של K-means.

### ניתוח האלגוריתם

נגדיר:

*N* – מספר כולל של אובייקטים.

*K* – מספר אשכולות.

*t* – מספר איטרציות של K-means.

*M* – מספר ממדים.

ננתח שלב שיפור הפרמטרים:

* חישוב לוקח זמן, לכן גם יקח זמן.
* לכן חיפוש הקצר ביותר יקח זמן.
* סיבוכיות זמן של בעיית הכרעה "האם דומיננטי על ביחס לh-?" היא .
* לכן הבדיקה האם תיקח זמן. נשים לב שאם לא מצאנו , נוכל לזרוק את כל מרכזי הכובד שבטוח לא יכולים להיות ה-owner-ים של ילדי h (לפי Blacklisting Algorithm).
* עדכון מונים של מרכזי כובד ייקח זמן.

לכן החישוב בכל צומת בעץ ייקח זמן [10].

במקרה הגרוע נבקר בכל הצמתים בעץ. נניח שמדובר בעץ מאוזן. דהיינו, עץ בו כמות הצמתים היא מאותו סדר גודל כמו כמות העלים - .

לפי [9], סיבוכיות זמן של בניית עץ mrkd היא .

לכן:

**סיבוכיות זמן** .

שלב שיפור המבנה הוא בעצם שלב שיפור הפרמטרים עם K=2, לכן סיבוכיות זמן נקבעת לפי סיבוכיות של שלב שיפור הפרמטרים.

נזכיר שהאלגוריתם מתבסס על הנחה שלא נגיע למצב בו הגענו לעלים בעץ.

נזכיר גם שבד"כ ו .

נבדוק כמה זיכרון נדרש:

* כל מונה מחזיק מספר כולל של נקודות ששייכות למרכז הכובד וסכום וקטורי של הנקודות. ז"א נדרש מקום עבור מונים.
* כל צומת בעץ מחזיק שני וקטורים בגודל M (עבור היפר-מלבנים), סכום וקטורי של הנקודות בצומת ומרכז הכובד שלהן. יש כ צמתים בעץ. לכן נדרש מקום עבור העץ.
* כל צומת יחזיק את התרומה של נקודות הצומת למונים של מרכזי הכובד. הדבר דורש מקום.
* עבור השוואה יעילה של מיקומי מרכזי הכובד צריך מקום. [10]

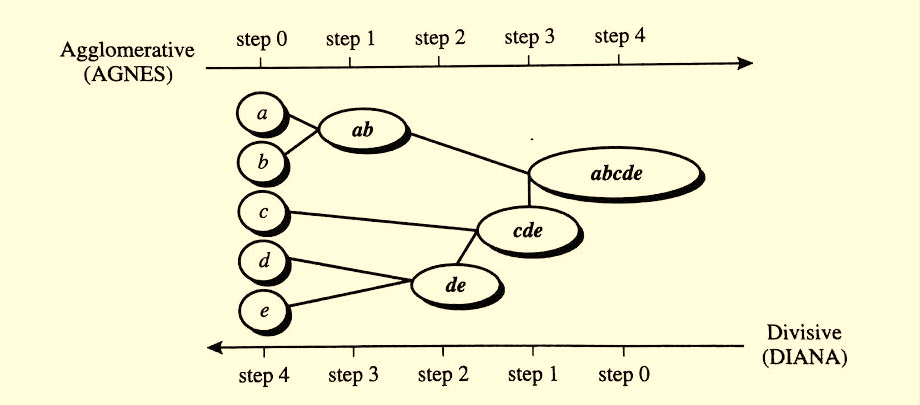
לכן:

**סיבוכיות מקום** .

## אלגוריתם אישכול היררכי

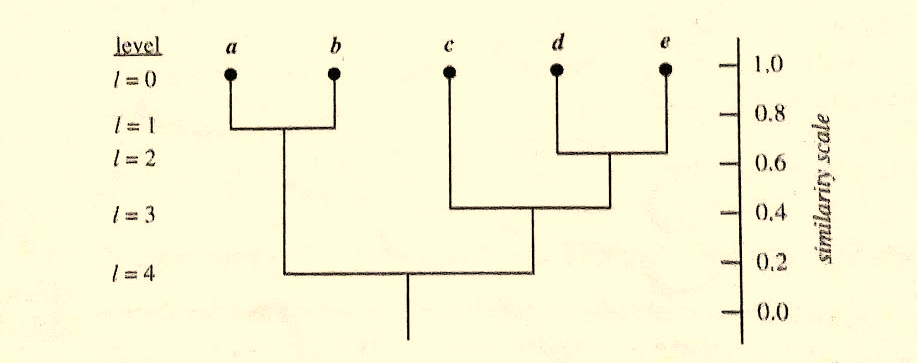
### מבוא

אישכול היררכי יוצר פירוק היררכי של קבוצת אובייקטי מידע. בסיום ריצת האלגוריתם נוצרים אשכולות מקוננים המסודרים בעץ היררכי. מבחינים במשפחת אלגוריתמים היררכיים בין אלגוריתמים צוברים (Agglomerative) ומפלגים (Divisive), כמתואר ב איור 13:



איור 13. אשכול צובר ומפלג על אובייקטי מידע {a, b, c, d, e} [7].

בדרך כלל משתמשים בעץ שנקרא dendrogram לייצוג תהליך של אשכול היררכי. ניתן לראות ב dendrogram איך האובייקטים מקובצים (או אשכולות מפולגים) צעד אחרי צעד.



איור 14. dendrogram של אובייקטים שהוצגו באיור 13.

מראה חמשת האובייקטים כאשכולות. ב אובייקטים a ו-b מקובצים ביחד כדי ליצור אשכול ראשון והם נשארים להיות ביחד בכל השלבים הבאים. נוכל להשתמש בציר אנכי כדי להראות סקלת דמיון בין האשכולות. לדוגמא, כאשר דמיון בין שתי קבוצות אובייקטים {a, b} ו {c, d, e} הוא בערך 0.16, הם מתאחדים כדי ליצור אשכול יחיד [7].

בעת בניית הדנדרוגרמה, יש להגדיר קריטריון לעצירת בנייתה. דהיינו, יש להגדיר מהי הנקודה שבה יש לחתוך את עץ האשכולות ההיררכי.

### מטרה

נתון בסיס נתונים של N אובייקטים. מעוניינים לבצע אשכול בהתאם לעקרונות הבאים:

1. כל אשכול חייב להכיל לפחות אובייקט אחד.
2. כל אובייקט חייב להשתייך לאשכול אחד בדיוק.
3. קיימת מידת דמיון גבוהה בין האובייקטים בתוך אותו אשכול (קריטריון הקומפקטיות).
4. מידת הדמיון בין האובייקטים באשכולות שונים היא נמוכה (קריטריון ההפרדה).

נרצה גם לקבוע את רמת דיוק האשכול תוך כדי ריצת האלגוריתם ולא להגדיר את K (מספר האשכולות) מראש, כמו ב K-means.

בנוסף נרצה לקבל אותה תוצאה בכל הרצת האלגוריתם (הדבר לא אפשרי ב K-means).

### תיאור האלגוריתמים

#### אלגוריתמים צוברים (Agglomerative)

מייצרים אשכולות בגישת Bottom - Up. באלגוריתמים מסוג זה, כל אובייקט משויך בהתחלה לאשכול אטומי משלו, ולאחר מכן האשכולות האטומיים מתחילים להתמזג זה עם זה וליצור אשכולות גדולים שאף הם מתמזגים זה עם זה עד לתנאי עצירה נתון. כל מיזוג מביא ליצירת שלב בהיררכיה (מדרג). נציג את הפסאודוקוד של האלגוריתם הפרימיטיבי:



אלגוריתם 2. אשכול היררכי Bottom-Up [11]

שורות 1-4: כל אובייקט משויך בהתחלה לאשכול אטומי משלו, גודל של כל אשכול הוא 1.

שורה 5: מבצעים N-1 איטרציות.

שורה 6: נמצא זוג אשכולות הכי קרובים לפי מטריצת מרחקים d.

שורות 7-9: נעדכן קבוצת אשכולות ודנדרוגרמה.

שורה 10: FORMULA בשורה 10 היא פונקציית מרחק מאשכול לאשכול החדש . כאן מעדכנים את מטריצת המרחקים d - עבור כל אשכול מעדכנים את המרחק שלו לאשכול החדש .

שורה 11-12: נעדכן את גודל האשכול החדש ואת קבוצת האשכולות.

שורה 14: אחרי N-1 איטרציות נחזיר את הדנדרוגרמה שהתקבלה.

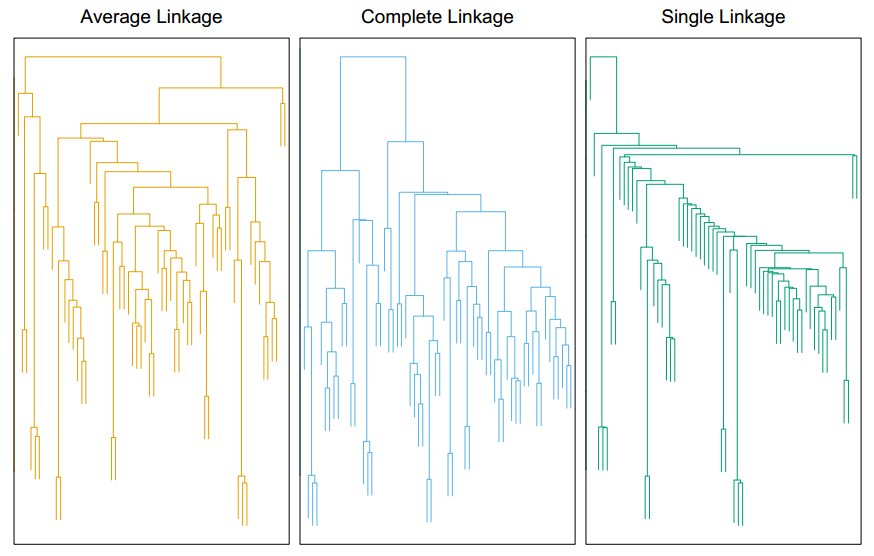
בהינתן זוג אשכולות A ו-B, נסביר אודות פונקציות מרחק שכיחות:

קישור יחיד - Single linkage

טכניקת הקישור היחיד מיוצגת על ידי המשוואה הבאה:

|  |  |  |
| --- | --- | --- |
|  |  | (4.11) |

זהו המרחק בין שני האובייקטים הקרובים ביותר, כשכל אובייקט שייך לאשכול אחר.

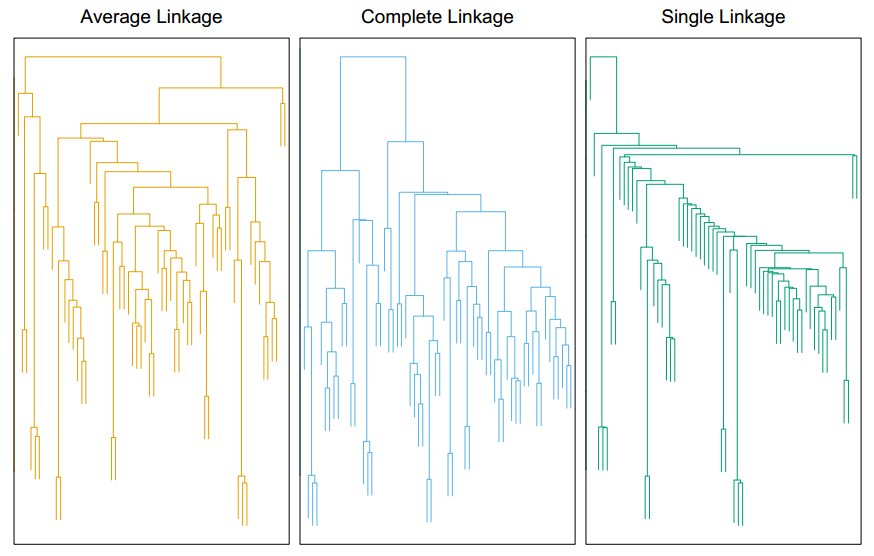


איור 15. Single linkage [12].

קישור מלא - Complete linkage

טכניקת קישור מלא מוצאת את זוג האובייקטים הרחוקים ביותר כאשר שני האובייקטים נמצאים באשכולות שונים.

|  |  |  |
| --- | --- | --- |
|  |  | (4.3) |

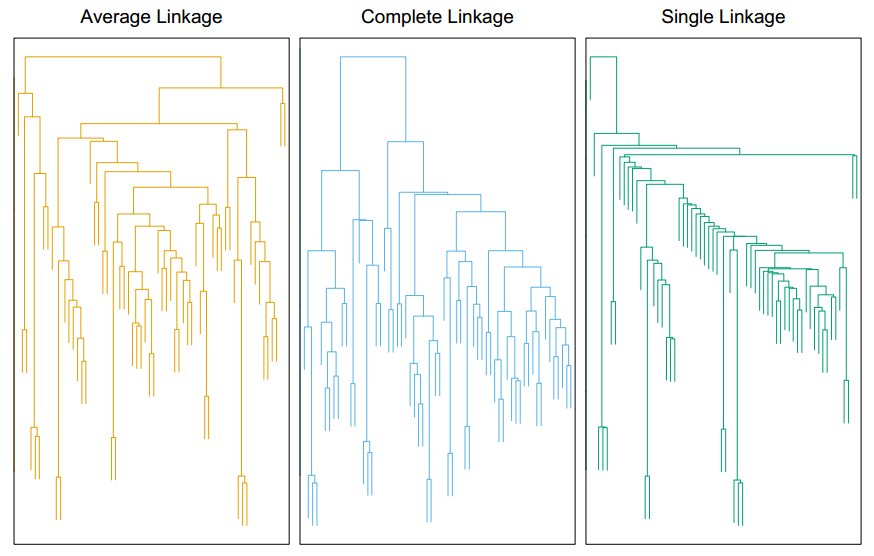


איור 16. Complete linkage [12].

ממוצע קבוצתי – Group average

טכניקה זו מאזנת בין שתי הטכניקות הקודמות. בשיטה זו מוצאים את המרחק הממוצע בין כל שני אובייקטים באשכולות.

|  |  |  |
| --- | --- | --- |
|  |  | (4.4) |



איור 17. Group/Average Linkage [12].

שלושת הטכניקות הפופולריות ביותר שהוצגו לעיל יתנו תוצאות דומות זה לזה ככל שהאשכולות מופרדים טוב יותר אחד מהשני ואיבריהם מקובצים קרוב אחד לשני. לפניכם טבלת השוואה בין השיטות:

|  |  |  |  |
| --- | --- | --- | --- |
| תוצאות | תכונות | הגדרת השיטה | שיטה |
| יכולים להיווצר אשכולות שלא מקיימים את קריטריון הקומפקטיות. | אשכולות נוצרים בצורת שרשראות ארוכות. נקודות מאוד רחוקות אחת מהשנייה יכולות למצוא את עצמן באותו אשכול. | שני איברים הכי קרובים באשכולות | Single Linkage |
| יכולים להיווצר אשכולות שלא מקיימים את קריטריון ההפרדה. | אשכולות נוצרים בצורת ספרות. אם האשכולות האמתיים הם קומפקטיים ושווים בגודלם השיטה תניב תוצאות טובות. | שני איברים הרחוקים ביותר באשכולות | Complete Linkage |
| אשכולות קומפקטיים הרחוקים אחד מהשני. | אלטרנטיבה מאזנת לשתי השיטות הקודמות. מושפעת פחות מאיברים חריגים. | ממוצע כל האיברים באשכולות | Average Linkage |

טבלה 4.1. טבלת השוואה בין השיטות השונות.

#### אלגוריתמים מפלגים (Divisive)

מייצרים אשכולות בגישת Top – Down. המצב ההתחלתי הוא שכל הנקודות מרוכזות באשכול אחד. באיטרציה הראשונה האשכול מתפצל לשני אשכולות. באיטרציות הבאות נבחר בכל פעם האשכול בעל מידת השוני הגבוהה ביותר לפיצול. קריטריון סיום הוא כשמידת השוני בין כל שני אשכולות קטנה ממידת השוני המינימלית שנקבעה מראש.

אשכול בשיטת ההפרדה אינו פופולרי בגלל הסיבוכיות שלו ומתאים יותר למקרים בהם רוצים פיצול למספר 2

אלגוריתם הכללי

1. במצב התחלתי אשכול אחד המכיל את כל האיברים.
2. כל עוד כל האשכולות לא פוצלו או קיים תנאי על שוני מינימלי, בצע:
   1. פצל כל אשכול לשני אשכולות.

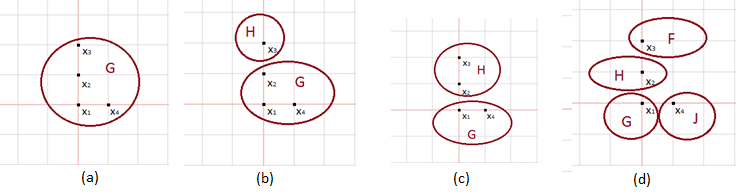
יישום השיטה על פי Macnaughton Smith [12]:

|  |
| --- |
| 1. Place all observations in a single cluster G. 2. Choose that observation whose average dissimilarity from all the other observations is largest. 3. This observation forms the first member of a second cluster H. 4. At each successive step that observation in G whose average distance from those in H, minus that for the remaining observations in G is largest, is transferred to H. 5. This continues until the corresponding difference in averages becomes negative. 6. That is, there are no longer any observations in G that are, on average, closer to those in H. 7. The result is a split of the original cluster into two daughter clusters, the observations transferred to H, and those remaining in G. These two clusters represent the second level of the hierarchy. 8. Each successive level is produced by applying this splitting procedure to one of the clusters at the previous level. |

אלגוריתם 3. אשכול היררכי Top-Down [12].

בחירת האשכול החדש לפיצול יכולה להיעשות בכמה אופנים:

1. אשכול בעל הקוטר הגדול ביותר
2. אשכול בעל השוני הגדול ביותר בין איבריו.



איור 18. דוגמא של פיצול לאשכולות על ידי אלגוריתם Divisive [7].

### יתרונות האלגוריתם

1. אשכול היררכי יכול לספק חלוקות שונות בהתאם לרמת הדיוק הדרושה. המשתמש מחליט על רמת האשכול ההולמת את צרכיו. אין צורך להגדיר את K (מספר האשכולות) מראש, כמו ב K-means.
2. ניתן להשתמש בכל פונקציית מרחק ולידית. לעומת זאת ב K-means משתמשים בפונקציית מרחק אוקלידי. לכן אשכול היררכי מתאים למציאת אשכולות בצורות מגוונות, לעומת K-means שעובד טוב עבור מציאת אשכולות בעלי צורה ספרית, אך עלול לגלות קושי בגילוי אשכולות בעלי צורה שרירותית.
3. מקבלים אותה תוצאה בכל הרצת האלגוריתם.

### חסרונות האלגוריתם

1. סיבוכיות זמן ריצה של אשכול היררכי היא פולינומית בגודל המאגר ולכן הוא בד"כ פחות סקלבילי מ X-means.

### ניתוח האלגוריתם

נגדיר N – מספר איברים שצריך לחלק לאשכולות.

#### Bottom-Up

**סיבוכיות זמן** של האלגוריתם הפרימיטיבי היא זמן [11]:

* ישנן N-1 איטרציות בסה"כ.
* *בכל האיטרציה* ה-i *מחשבים מרחקים בין האיברים ב* N-i*אשכולות.*

יש לשמור את המרחקים שחושבו בכל האיטרציה ה-i, לכן:

**סיבוכיות מקום** [11].

#### Top-down

ישנן אפשרויות לחלק N איברים לשני אשכולות. נסביר:

נבחר איבר כלשהו להיות באשכול A, עכשיו לכל אחד מ איברים שנשארו יש בחירה להצטרף ל-A או לא להצטרף אליו. סה"כ פרמוטציות. הפרמוטציה בה כל האיברים נמצאים באשכול A אינה מתאימה, לכן נחסיר אותה מהתוצאה. ונקבל פרמוטציות.

זה לא מעשי לעבור על כל חלוקה אפשרית, לכן Macnaughton Smith ניסה לבצע את החלוקה בצורה איטרטיבית. מכיוון שכאן אין נוסחת עדכון כמו באלגוריתם האגלומרטיבי, דיכוטומיה אחת גורמת לחישוב מחדש של קריטריון חלוקת איברים בין שני אשכולות שנוצרו. החישוב לוקח זמן. יצירת כל דיכוטומיה לוקחת זמן. מכיוון שבניית היררכיה בינארית שלמה לוקחת איטרציות, לכן

**סיבוכיות זמן** . סיבוכיות כזו דורשת כוח חישובי חזק, אבל הדבר עדיין אפשרי עבור מידע בגודל בינוני [12].

**סיבוכיות מקום** , כי בכל איטרציה צריך לשמור את תוצאות המרחקים הממוצעים של כל איבר [12].

# ניתוח השוואתי בין שלושת משפחות האלגוריתמים

## מבוא

בפרק זה נגדיר מדדי השוואה בין האלגוריתמים. נשווה בין שני נציגים של אלגוריתמי חלוקה – X-means ו K-means לבין אלגוריתמים היררכיים מהסוג average.

מטרת ההשוואה הינה לדעת מתי יש להשתמש בכל אחד מהאלגוריתמים.

## הגדרת מדדים להשוואה

### מדדים של איכות האישכול

שגיאה אבסולוטית ממוצעת

שגיאה אבסולוטית ממוצעת מודדת את ההבדל בין האומד לבין מה שנאמד.

|  |  |  |
| --- | --- | --- |
|  |  | (5.1) |

כאשר:

– אומד.

– ערך אמתי.

*n – מספר כולל של נקודות.*

קריטריון BIC

Bayesian Information Criterion או Schwarz Criterion – קריטריון שוורץ. זהו קריטריון לבחירת מודל מתוך קבוצה סופית של מודלים. הקריטריון מבוסס על פונקציית נראות.

|  |  |  |
| --- | --- | --- |
|  |  | (5.2) |

הקריטריון הוסבר בפרק ‏4.3.3.

Distortion

בהינתן אישכול , נגדיר להיות מרכז כובד המשויך לאובייקט ע"י האישכול. לכן בK-means הוא מרכז הכובד הכי קרוב ל-. נגדיר את ה-Distortion של האישכול:

|  |  |  |
| --- | --- | --- |
|  |  | (5.3) |

*– מרחק בין נקודה ל- .*

*n – מספר כולל של אובייקטים.*

*ידוע שK-means מתכנס למינימום לוקאלי של מידת עיוות.*

אנטרופיה (Entropy)

משתמשים באנטרופיה כדי למדוד את איכות האישכול. ככל שתצפית מסוימת קרובה לאשכול מסוים ורחוקה משאר האשכולות, ערך האנטרופיה נמוך יותר ואיכות האישכול גבוהה יותר. אבל צריך לזכור שבאישכול שבו כל אשכול מכיל בדיוק תצפית אחת, ערך האנטרופיה יהיה אופטימלי.

נניח הוא אישכול, נגדיר [13]:

|  |  |  |
| --- | --- | --- |
|  |  | (5.4) |

כאשר:

– גודל של אשכול j.

– מספר אשכולות.

- מספר כולל של תצפיות

היא אנטרופיה של אשכול j. בהינתן סיווג של תצפיות (שיוך כל תצפית למחלקה מסוימת), מסמן סיכוי שתצפית מתוך אשכול j שייכת למחלקה i.

|  |  |  |
| --- | --- | --- |
|  |  | (5.5) |

F - measure [13]

מדד שיעור ההחלטות הנכונות של האלגוריתם.

נשתמש במושגים של Recall ו-Precision.

Recall הוא שיעור התוצאות הרלוונטיות שהוחזרו ע"י האלגוריתם מתוך התוצאות הרלוונטיות הצפויות.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | (5.6) |

Precision הוא שיעור התוצאות הרלוונטיות מתוך כל התוצאות שהאלגוריתם החזיר.

|  |  |  |
| --- | --- | --- |
|  |  | (5.7) |

בהינתן סיווג תצפיות (שיוך כל תצפית למחלקה מסוימת), נתייחס לכל אשכול j כפלט של האלגוריתם ולכל מחלקה i כאל קבוצת תוצאות רלוונטיות צפויות. נחשב Recall ו-Precision של האשכול עבור כל מחלקה נתונה:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | (5.8) |  |
|  |  | (5.9) |  |

כאשר:

– מספר הנציגים של מחלקה i באשכול j.

– מספר התצפיות באשכול j.

– מספר התצפיות במחלקה i.

ערך של Precision ו-Recall נמצא בטווח .

Recall גבוה אומר שהאלגוריתם החזיר את רוב התוצאות הרלוונטיות. לכן שואפים שערך ה-Recall יהיה ככל הניתן גבוה.

Precision גבוה אומר שהאלגוריתם החזיר לרוב יותר תוצאות רלוונטיות מאשר לא רלוונטיות. לכן שואפים שערך ה-Precision יהיה ככל הניתן גבוה.

F - measure של אשכול j ומחלקה i הוא ממוצע הרמוני של Precision ו-Recall:

|  |  |  |
| --- | --- | --- |
|  |  | (5.10) |

F-measure שואף ל-1 כאשר Recall ו-Precision שואפים ל-1, לכן שואפים שערך F-measure יהיה ככל הניתן גבוה.

סך F - measure הוא שקלול של על פי גודל המחלקות:

|  |  |  |
| --- | --- | --- |
|  |  | (5.11) |

באישכול היררכי max מחושב מכל האשכולות בכל הרמות.

n – סה"כ התצפיות.

מכיוון ששואפים למקסם את , שואפים גם שסך F - measure יהיה ככל הניתן גבוה.

לכידות (Cohesiveness) [13]

מדד המודד את מידת הדמיון בין האובייקטים באשכולות.

אחת השיטות לחישוב הלכידות היא להשתמש בדמיון משוקלל של הדמיון הפנימי באשכול:

|  |  |  |
| --- | --- | --- |
|  |  | (5.12) |

כאשר

*d* – ייצוג וקטורי של תצפיות.

*S* – קבוצת תצפיות וייצוג וקטורי מתאים שלהן.

|  |  |  |
| --- | --- | --- |
|  |  | (5.13) |

נשתמש במדד cosine כמדד לדמיון בין שני וקטורים. ערך המדד נע בין -1 (שני הווקטורים מנוגדים) לבין 1 (שני הווקטורים מתלכדים). לכן שואפים שהערך יהיה ככל הניתן קרוב ל-1.

מרכז הכובד מוגדר בצורה הבאה:

|  |  |  |
| --- | --- | --- |
|  |  | (5.14) |

מכאן:

|  |  |  |
| --- | --- | --- |
|  |  | (5.15) |

אז הלכידות מחושבת כאורך בריבוע של מרכז הכובד של אשכול.

מכיוון שהלכידות היא סכום משוקלל של cosine-ים , שואפים שערך המדד יהיה ככל הניתן גבוה.

### מדדים של האלגוריתם

סיבוכיות זמן

#### נתון אורך קלט n.

#### סיבוכיות זמן של [אלגוריתם](https://he.wikipedia.org/wiki/%D7%90%D7%9C%D7%92%D7%95%D7%A8%D7%99%D7%AA%D7%9D) הוא הזמן הדרוש לאלגוריתם לרוץ כ[פונקציה](https://he.wikipedia.org/wiki/%D7%A4%D7%95%D7%A0%D7%A7%D7%A6%D7%99%D7%94) של אורך הקלט n.

סיבוכיות מקום

כמות זיכרון הדרושה בכל נקודה באלגוריתם.

## תוצאות ניסוייות

### השוואה בין K-means לבין X-means

ניתוח השוואתי בין האלגוריתם K-means לבין האלגוריתם X-means מבוסס על המאמרים של

Pelleg and Moore [8] [10].

#### השוואת X-means ו K-means לפי Distortion של האישכול [8]

הגדרת בסיס הנתונים

במאמר של Pelleg and Moore [8] יצרו 5 בסיס נתונים סינטטיים. עבור כל בסיס נתונים בחרו מרכזי כובד רנדומליים והציבו סביבם נקודות שמתפלגות גאוסיאנית עם תוחלת במרכז כובד ושונות (3 זהו מספר ממדים). כל בסיס הנתונים מכיל 250 אשכולות. כמות הנקודות בכל בסיס הנתונים הייתה 80.000, 90.000, 100.000, 110.000 ו -120.000 בהתאם.

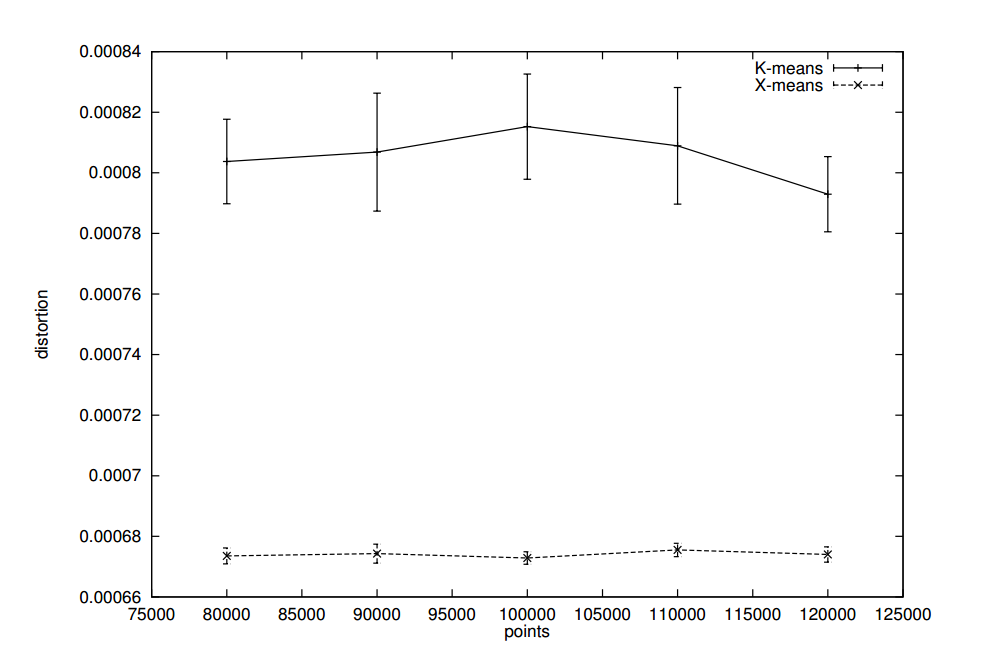
תיאור הניסוי

על כל בסיס נתונים ביצעו 30 ריצות של X-means ו K-means.

K-means קיבל מספר מרכזי כובד אמתי, לעומת זאת X-means היה צריך לחפש מספר מרכזי כובד מתאים בטווח.

תוצאות הניסוי

כל תוצאה בגרף היא ממוצע של 30 ריצות.



איור 19. Distortion של האישכול. התוצאות הן ממוצע של 30 ריצות על מידע 3-D שמכיל 250 אשכולות אמתיים [8].

נתבונן באיור 19. ניתן לראות שערכי Distortion של תוצאות X-means היו נמוכים יותר. זאת אומרת קיבלנו תוצאות איכותיות יותר עבור X-means.

מסקנות

X-means מוסיף מרכזי כובד חדשים בהדרגתיות ובמקומות שצריך אותם, בניגוד ל K-means שמשתמש במרכזי הכובד שקיבל כקלט בתחילת הריצה. לכן התוצאות של X-means יותר איכותיות מבחינת Distortion.

#### השוואת זמן ריצה של X-means ו K-means [8]

הגדרת בסיס הנתונים

יצרו 9 בסיסי נתונים 3-D כמו מקודם תוך שימוש בהתפלגות גאוסיאנית. כמות האשכולות בבסיסי הנתונים היא וכמות הנקודות בבסיסים נעה בין 4.000 ל-36.000 (סה"כ 9 אפשרויות לכמות הנקודות).

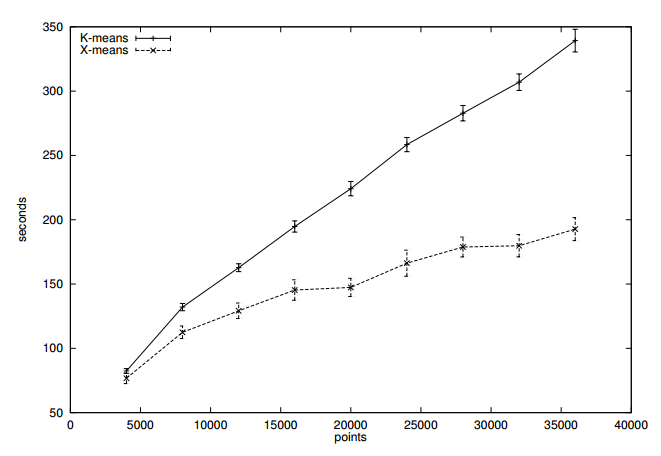
תיאור הניסוי

על כל בסיס נתונים ביצעו 30 ריצות של X-means ו K-means.

K-means קיבל מספר מרכזי כובד אמתי, לעומת זאת X-means היה צריך לחפש מספר מרכזי כובד מתאים בטווח.

תוצאות הניסוי

כל תוצאה בגרף היא ממוצע של 30 ריצות.



איור 20. זמני ריצה של X-means ו K-means. התוצאות הן ממוצע של 30 ריצות על מידע 3-D שמכיל 250 אשכולות אמתיים [8].

ניתן לראות שככל שגודל הקלט גדל, ההבדל בין זמני ריצה של X-means ו K-means גדל לטובת X-means.

מסקנות

* זמן ריצת X-means טוב יותר מזן הריצה של K-means.
* X-means סקלבילי יותר מ K-means - ככל שגודל הקלט גדל, ההבדל בין זמני ריצה של

X-means ו K-means גדל לטובת X-means.

#### השוואת X-means ו K-means לפי שגיאה אבסולוטית ממוצעת של מספר האשכולות [8]

שאלה מעניינת אחרת היא עד כמה טוב ה X-means בלגלות מספר אמתי של אשכולות.

הגדרת בסיס הנתונים

יצרו 27 בסיסי נתונים 2-D כמו מקודם תוך שימוש בהתפלגות גאוסיאנית. כמות האשכולות בבסיסי הנתונים נעה בין וכמות הנקודות בבסיסים נעה בין 4.000 ל-36.000 (סה"כ 9 אפשרויות לכמות הנקודות).

תיאור הניסוי

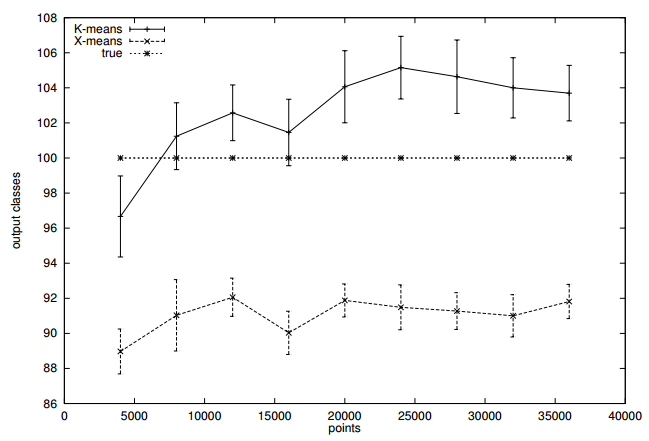
על כל בסיס נתונים ביצעו 30 ריצות של X-means. טווח מספר מרכזי כובד עבור X-means: .

על כל בסיס נתונים ביצעו 30 ריצות של K-means משודרג.

K-means משודרג מריץ K-means 20 פעמים: משתמשים ב-20 ערכים שנמצאים במרחקים שווים מ-2 עד עבור מספר מרכזי כובד כקלט. ובוחרים בריצה שקיבלה ציון BIC (מוסבר בפרק ‏4.3.3) מקסימלי.

תוצאות הניסוי

נתבונן בתוצאת ריצה על 9 בסיסי נתונים בעלי 100 אשכולות כל אחד. כל תוצאה בגרף היא ממוצע של 30 ריצות.

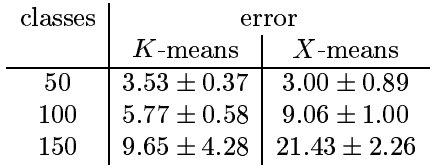


איור 21. מספר האשכולות בפלט כפונקציה של גודל הקלט .התוצאות הן ממוצע של 30 ריצות על מידע 2-D שמכיל 100 אשכולות אמתיים [8].

כפי שניתן לראות K-means נוטה להעריך יתר על המידה את מספר האשכולות. וככל שמספר הרשומות R גדל, כך גדל גם מספר האשכולות ב K-means. לעומת זאת X-means בד"כ נותן הערכה נמוכה מדי ל-K ובכללי הוא אינו רגיש ל-*R*.

בניסוי זה התוצאה של K-means יותר קרובה לערך אמתי מהתוצאה של X-means.

ניקח ממוצע של שגיאות אבסולוטיות של 9 הריצות עבור :



טבלה 5.1. ממוצע של שגיאות אבסולוטיות של מספר האשכולות. [8]

כפי שניתן לראות מהתוצאות, X-means מחזיר מבנה שבו מספר אשכולות בטווח 15% ממספר אמתי של אשכולות. ניתן גם להבחין בכך ש K-means מניב תוצאות יותר טובות מבחינה זו – סטייה של 6% בממוצע.

מסקנות

* X-means פחות רגיש למספר הרשומות R. לכן הוא יותר סקלבילי.
* X-means הניב הערכה פחות טובה לכמות האשכולות (אך עדיין מספיק איכותית) מ K-means המשודרג. נזכור גם ש K-means המשודרג הריץ K-means 20 פעמים – הדבר פחות יעיל (מבחינת זמן) מלהריץ X-means פעם אחת.

#### השוואת X-means ו K-means לפי ערך [8]BIC

הגדרת בסיס הנתונים

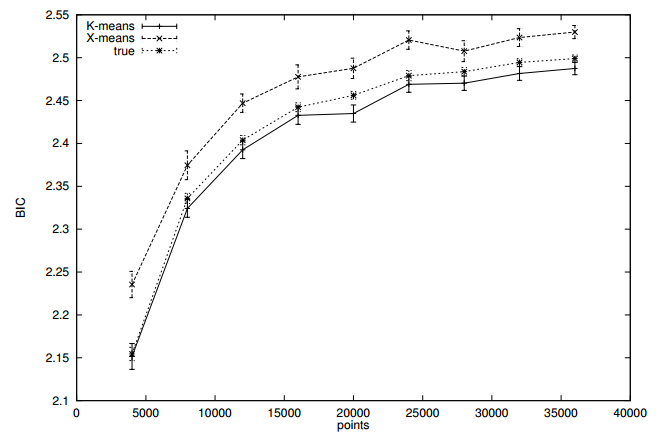
השתמשו באותם בסיסי נתונים שבפרק ‏5.3.1.3**.**

תיאור הניסוי

מהלך הניסוי זהה לזה שבפרק ‏5.3.1.3, רק במקום למדוד כמות האשכולות בפלט חישבו את ציון BIC (מוסבר בפרק ‏4.3.3) של האישכול שהתקבל.

תוצאות הניסוי

נתבונן בתוצאת ריצה על 9 בסיסי נתונים בעלי 100 אשכולות כל אחד. כל תוצאה בגרף היא ממוצע של 30 ריצות.



איור 22. ציון BIC של אישכול שהתקבל. התוצאות הן ממוצע של 30 ריצות על מידע 2-D שמכיל 100 אשכולות אמתיים [8].

ערך אמתי בגרף זהו ערך BIC של מרכזי הכובד ששימשו ליצירת המידע.

ניתן לראות שתוצאות של X-means טובות יותר מאלה של K-means.

יתר על כך, האשכולות ש X-means יצר אפילו קיבלו ציון BIC גבוה יותר מהציון שקיבלו מרכזי כובד שבעזרתם יצרו את בסיסי הנתונים. הדבר אפשרי אם ניתן היה לייצג את בסיס הנתונים ע"י מספר קטן יותר של מרכזי הכובד. לדוגמא, שני (או יותר) מרכזי כובד שנבחרו רנדומלית היו קרובים אחד לשני ויכלו לייצר אשכול יחיד במקום 2 אשכולות נפרדים.

מסקנות

* X-means מספק אישכול איכותי יותר מאישכול של K-means.

### השוואה בין K-means לבין אישכול היררכי אגלומרטיבי

ניתוח השוואתי בין K-means הנאיבי לבין אשכול היררכי אגלומרטיבי מסוג average מבוסס על המאמרים של Steinbach and Karypis [13] ו Kaur and Kaur [14].

הגדרת בסיס הנתונים

מאמר של Steinbach and Karypis [13] עוסק באישכול של מסמכים.

בנתוני הקלט בוצע טיוב שכלל השמטת מילות עצירה כגון “a”, “are”, “do”, and “for”. בנוסף כל המסמכים עברו stemming (מציאת שורש המילה) ע"י הרצת suffix-stripping Porter’s. לכן כל המילים שיש להן אותו שורש נחשבות לאותה מילה. לדוגמא מילים “compute”, “computing”, and “computed” שויכו למילה "comput".

לאחר מכן בוצע סיווג המסמכים (שיוך כל מסמך למחלקה).

במאמר של Kaur and Kaur [14] לקחו 400 רשומות של נתוני סטודנטים (ציונים במקצועות, ציון ממוצע, שם וכו). גם כאן בוצע סיווג של הנתונים.

תיאור הניסוי

ב [13] הריצו את שני האלגוריתמים על 8 בסיסי נתונים וחיפשו כמות אשכולות שונה באותם בסיסי נתונים

().

ב [14] הריצו את שני האלגוריתמים על אותו בסיס נתונים כאשר כמות האובייקטים בו נעה בין 50 ל-400.

K-means חילק את הנתונים לשני אשכולות ואישכול היררכי ביצע חלוקה ל-5 אשכולות.

**תוצאות ניסוייות**

השוואת K-means ואשכול היררכי לפי אנטרופיה

ב [13] K-means הניב תוצאות אנטרופיה יותר טובות מאשכול היררכי.

לעומת זאת ב [14] אשכול היררכי הניב תוצאות אנטרופיה יותר טובות מ K-means.

השוואת K-means ואשכול היררכי לפי F measure

ב [13] הריצו Bisecting K-means (K-means משופר) וקיבלו תוצאות F measure יותר טובות עבור K-means.

ב [14] לעומת זאת קיבלו תוצאות F measure יותר טובות עבור אשכול היררכי.

השוואת K-means ואשכול היררכי לפי לכידות (Cohesiveness)

ב [13] K-means הניב תוצאות לכידות יותר טובות מאשכול היררכי.

השוואת זמן ריצה של K-means ואשכול היררכי

ב [13] אישכול היררכי רץ יותר משעה על אותו בסיס נתונים ש X-means היה רץ פחות מדקה.

ב [14] זמן ריצת K-means יותר קצר. ככל שבסיס הנתונים גדל, גדל גם זמן ריצת האלגוריתם – הדבר נכון גם לאישכול היררכי וגם ל K-means. אבל זמן ריצה של אישכול היררכי מושפע יותר מגודל בסיס הנתונים.

**ניתוח תוצאות ומסקנות**

מבחינת איכות החלוקה אין קונסנסוס על השיטה העדיפה , כי יש מאגרי מידע שהולמים יותר ל K-means ויש מאגרי מידע שמתאימים יותר לאשכול היררכי.

למשל:

* מאגר המידע במאמר [13] מכיל הרבה איברים שהם שכנים קרובים, אך שייכים למחלקות שונות. במצבים בהם שיטת השכן הקרוב אינה אמינה, דרושה גישה המסתמכת על תכונות גלובאליות יותר. חישוב מרחק ממרכז כובד למסמך זהה לחישוב דמיון ממוצע בין המסמך לבין כל המסמכים שנמצאים באשכול, לכן K-means כבר משתמש בגישה גלובאלית. זה יכול להסביר למה K-means הניב חלוקה טובה יותר ב [13].
* מצד שני K-means עובד טוב עבור מציאת אשכולות בעלי צורה ספרית, אך עלול לגלות קושי בגילוי אשכולות בעלי צורה שרירותית [7]. אשכול היררכי לעומת זאת מתאים למציאת אשכולות בצורות מגוונות (בגלל שניתן להשתמש בכל פונקציית מרחק ולידית).

מבחינת זמן ריצה וסקלביליות K-means עדיף על אישכול היררכי.

## סיכום

נגדיר:

*N* – מספר כולל של אובייקטים.

*K* – מספר אשכולות.

*t* – מספר איטרציות גלובאליות.

*M* – מספר ממדים.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| מדדים | אשכול היררכי Top-down | אשכול היררכי Bottom-up | X-means | K-means |
| סיבוכיות זמן |  |  |  |  |
| סיבוכיות מקום |  |  |  |  |
| מעבר על פרמוטציות | ברגע שהצעד נעשה (איחוד או פיצול), אי אפשר לבטלו, לכן אין צורך לדאוג לפרמוטציות קומבינטוריות שונות – הדבר מקטין זמן חישוב. [7] | | כדי לא לעבור על כל הפרמוטציות האפשריות, משתמשים בשיטה יוריסטית שבה אשכול מוצג ע"י ערך ממוצע של אובייקטים באשכול. [7] | |
| K | אין להגדיר K מראש. | אין להגדיר K מראש. | אין להגדיר K מראש. | הגדרת K מראש. |
| גמישות בחלוקות | אשכול היררכי יכול לספק חלוקות שונות בהתאם לרמת הדיוק הדרושה. המשתמש מחליט על רמת האשכול ההולמת את צרכיו. | | אשכול מבוסס חלוקה. ניתן להסתפק בחלוקה אחת בלבד. | |
| פונקציית מרחק | ניתן להשתמש בכל פונקציית מרחק ולידית. | | משתמשים בפונקציית מרחק אוקלידי. | |
| זמן ריצה בניסויים | זמן ריצת K-means יותר קצר מזה של אשכול היררכי. [14] | | X-means מראה תוצאות הרבה יותר טובות מ K-means. [8] | |
| מדד BIC |  |  | האשכולות של X-means היו יותר טובים לפי מדד BIC מאלה של K-means. [8] | |
| Distortion |  |  | ערכי Distortion של אישכול  X-means היו נמוכים יותר (זאת אומרת קיבלנו תוצאות איכותיות יותר) מאלה של K-means. [8] | |
| שגיאה אבסולוטית ממוצעת |  |  | K-means הניב תוצאות יותר טובות מ X-means. [8] | |
| Entropy | אין הסכמה חד משמעית לגבי עדיפות של K-means על שיטות אשכול היררכי ולהפך. קיימים מאגרי מידע ההולמים יותר לאשכול היררכי ויש כאלה שמתאימים יותר ל .K-means [14] [13] | | | |
| F measure | אין הסכמה חד משמעית לגבי עדיפות של K-means על שיטות אשכול היררכי ולהפך. התוצאות תלויות במאגר נתונים שרצים עליו. יש מאגרי מידע שמתאימים יותר לאשכול היררכי ויש המתאימים יותר ל .K-means [14] [13] | | | |
| לכידות | K-means הניב תוצאות יותר טובות מאשכול היררכי. [13] | | | |
| סקלביליות | סיבוכיות האלגוריתם פולינומיאלית כתלות במספר האובייקטים N. לכן האלגוריתם עובד טוב על מאגרי מידע קטנים. | | ריצת X-means על קבוצת 800.000 אובייקטים ו-4.000 מרכזי כובד שנוצרו לקחה 4.5 שעות. [8] | זמן ריצת K-means  על 800.000 אובייקטים הגיע לסף עליון אחרי שהיה רץ כבר פי 2 יותר זמן מ X-means. [8]  K – means עובד טוב על מאגרי מידע בגודל קטן עד ממוצע. [7] |
| אשכול היררכי Top-down הוא בחירה פחות טובה למאגרי מידע גדולים מכיוון שסיבוכיות זמן ריצה שלו יותר גדולה. | | תוצאות ניסוייות מוכיחות שככל שמאגר נתונים גדל, X-means מראה ביצועים יותר טובים מ K-means. [8] [7] | |
| מידע רב ממדי | ככל שמספר הממדים גדל, המידע מתחיל להיות דליל יותר ויותר, כך שמדידת מרחק בין זוגות נקודות נהפכת לחסרת משמעות. [7] | | סיבוכיות האלגוריתם היא כתלות במספר ממדים M. לכן זמן ריצה על מידע רב ממדי יכול להיות גדול מאוד.  בנוסף, ככל שמספר הממדים גדל, המידע מתחיל להיות דליל יותר ויותר, כך שמדידת מרחק בין זוגות נקודות נהפכת לחסרת משמעות. [7] | |
| מבנה מאגר נתונים | אשכול היררכי מתאים למציאת אשכולות בצורות מגוונות (בגלל שניתן להשתמש בכל פונקציית מרחק ולידית). | | עובד טוב עבור מציאת אשכולות בעלי צורה ספרית, אך עלול לגלות קושי בגילוי אשכולות בעלי צורה שרירותית. [7] | |
| כשהריצו את שני האלגוריתמים על קובץ שהכיל נתונים על 330.000 גלקסיות, זמן ריצה של X-means היה 238 שניות ולעומת זאת זמן ריצה של K-means היה 7793 שניות. [8] | |

טבלה 5.2. טבלת השוואה בין שלושת האלגוריתמים.

# שימוש באלגוריתמי אישכול לזיהוי רשתות בוטנט

## מבוא

בפרק זה יוצגו שלוש שיטות שונות לזיהוי בוטנטים.

שיטה ראשונה מזהה בוטנט בעזרת חתימות. מטרת המערכת היא זיהוי תוכנות זדוניות שמתקשרות עם Web בצורה דומה ואיחוד של אותן התוכנות בקבוצות. לאחר מכן יוצרים חתימות שמייצגות כל קבוצה ועוזרות בזיהוי בקשות HTTP חשודות.

החלוקה לקבוצות מתבצעת על בסיס לוגים של תעבורת HTTP. אופי הקבוצות שאמורות להיווצר אינו ידוע מראש, לכן שימוש באלגוריתמי אישכול הוא הפתרון המתבקש.

Perdisci, Lee & Feamter [5] השתמשו באישכול היררכי אגלומרטיבי שמשתמש ב-single linkage כפונקציית מרחק בין האשכולות. המוטיבציה העיקרית לבחירת אישכול היררכי היא עובדה שאישכול היררכי מסוגל לאתר אשכולות בצורות מגוונות ואינו מוגבל למרחב אוקלידי.

כפי שהוסבר בפרקים 4 ו 5, זמן ריצה של אישכול היררכי בד"כ ארוך יותר מזמן ריצת X-means. אבל הניסויים הראו שחישוב אשכולות (בעזרת אישכול היררכי) עבור 25720 תוכנות זדוניות שנאספו במהלך חודש מסוים, לוקח שעות ספורות בלבד. לכן גם מבחינת זמן ריצה אישכול היררכי נמצא מתאים לצרכי המערכת.

כדי לקבל החלטה סופית, Perdisci, Lee & Feamter [5] אף הריצו X-means, אישכול היררכי מסוג single linkage ואישכול היררכי מסוג complete linkage על הנתונים שנוצרו ע"י המערכת. אישכול היררכי מסוג single linkage הראה ביצועים הכי טובים.

שיטה שנייה מתבססת על ניתוח כמות גדולה של לוגים במערכת ארגונית. מקור הלוגים הוא כל מחשבי הארגון (הוסטים). מכיוון שלא ידוע מראש איזה הוסטים הודבקו או מתנהגים אנומלית, זיהוי ההוסטים הזדוניים התבצע בעזרת למידה בלתי מונחית – אישכול. ההנחה היא שעובדי החברה מבצעים פעולות ספציפיות ברשת ארגונית ויש מספר עובדים בכל מחלקה בחברה. לכן אחרי האישכול נמצא אשכולות של הוסטים שמתנהגים בצורה דומה והוסטים נפרדים שמתנהגים שונה מאחרים – הוסטים חשודים.

המערכת בונה ווקטור מאפיינים רב ממדי עבור כל הוסט ומבצעת אישכול של הווקטורים. כמו בשיטה הראשונה, האישכול בניסוי התבצע על כ - 30000 אובייקטים. אבל כאן Yen,Oprea & Onarlioglu [15] החליטו להשתמש בוריאציה של K-means עבור האישכול. כפי שהוסבר בפרקים 4 ו 5, K-means סקלבילי יותר מאישכול היררכי. בנוסף, הניסויים הראו שבד"כ מספיקה איטרציה אחת של וריאציה של K-means כדי לבודד מחשבים בעלי התנהגות חשודה.

השיטה השלישית מכוונת לזיהוי כללי של רשתות בוטנט באינטרנט, ללא תלות בפרוטוקול תקשורת או ארכיטקטורת שרתי שליטה. השיטה פועלת בשני מישורים לזיהוי בוטנט – מישור המזהה פעילות זדונית (DDOS, ספאם) ברשת ומישור שמנתח פרמטרים שונים של תקשורת ברשת.

במישור התקשורתי המערכת מחפשת התקשרויות דומות. כדי למצוא התקשרויות דומות השתמשו באישכול X-means. Gu&Perdisci [4] בחרו ב X-means מכיוון שלעומת K-means אלגוריתם אישכול זה אינו דורש לספק K – מספר האשכולות (ואכן מספר האשכולות אינו ידוע מראש). בנוסף X-means, כפי שהוסבר בפרקים 4 ו 5, מראה ביצועים טובים עבור בסיסי נתונים גדולים (מאות אלפי התקשרויות) ומידע רב ממדי (52 ממדים). כאן האישכול התבצע על מאות אלפי אובייקטים לעומת עשרות אלפי אובייקטים בשיטה הראשונה והשנייה ולכן ברורה הבחירה ב X-means שהוא סקלבילי יותר מ K-means ואישכול היררכי.

במישור הפעילות הזדונית לא משתמשים באלגוריתם אישכול קלאסי כלשהו. פשוט מקבצים הוסטים לפי סוג הפעילות (סריקות, הורדות תוכנות זדוניות, ספאם ועוד...).

בשלב הסופי מבצעים איחוד של שני סוגי האישכולים בעזרת אישכול היררכי אגלומרטיבי. מחברי המאמר בחרו כאן באישכול היררכי, כי פונקציית מרחק בין שני ההוסטים שהגדירו אינה אוקלידית.

## אישכול התנהגותי של בוטנט מבוסס HTTP ויצירת חתימות לזיהוי בוטנט [5]

### הקדמה

כותבי תוכנות זדוניות דואגים ליצור כמות גדולה של גרסאות פולימורפיות של אותה תוכנה זדונית (לדוגמא בעזרת obfuscation). הדבר גורם להרבה false negatives של תוכנות אנטיוירוס. אבל יש גורם משותף בין הגרסאות הפולימורפיות של אותה תוכנה – אותן התוכנות עדיין יתנהגו דומה. לדוגמא, הן יתקשרו עם Web באותו אופן. ברגע שמקבצים ביחד אותן תוכנות זדוניות שמתנהגות בצורה דומה, ניתן ליצור חתימות התנהגותיות גנריות שתעזורנה לאתר תוכנות זדוניות חדשות תוך כדי שמירה על רמה נמוכה של false negatives ו false positives.

### מטרת המערכת

מטרת המערכת היא למצוא קבוצות של תוכנות זדוניות שמתקשרות עם Web בצורה דומה, ללמוד מהו מודל ההתנהגות של כל קבוצה, ולאחר מכן להשתמש במודלים אלה כדי לזהות נוכחות של מכונות שנחשפו לתוכנות זדוניות ברשת מנוטרת.

### תיאור המערכת

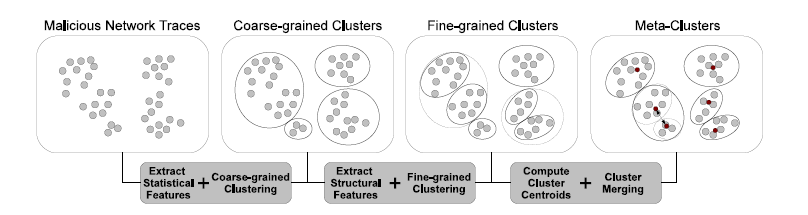
בהתחלה מבצעים אשכול התנהגותי של דגימות של תוכנות זדוניות ע"י מציאת דמיון מבני בין צירופים של בקשות HTTP שנוצרו כתוצאה מהידבקות. כלומר, בהינתן קובץ של דגימות תוכנות זדוניות , הריצו כל דגימה בסביבה מבוקרת למשך זמן T, ושמרו לוגים של תעבורת HTTP שלה . לאחר מכן חילקו לאשכולות בהתאם להגדרה של דמיון מבני בין לוגים של תעבורת HTTP .

### ארכיטקטורת המערכת

כדי להגיע לאשכולות באיכות גבוהה ולהקטין עלות חישובית של אישכול, ביצעו אישכול בשלושה שלבים:

* בהתחלה ביצעו אישכול גס בהתבסס על סטטיסטיקות פשוטות של תעבורת HTTP. לכן, חישוב המרחק בין זוגות של דגימות תוכנות זדוניות מצטמצם לחישוב המרחק בין וקטורים מספריים. ניתן לבצע חישוב כזה ביעילות.
* לאחר מכן ביצעו אישכול עדין שמתבסס על דמיון מבני בין רצפים של בקשות HTTP. אישכול עדין רץ על כל אחד מאשכולות גסים שנוצרו.
* בשלב זה מבצעים איחוד אשכולות דומים שנוצרו באישכול עדין.

ניתן לראות את השלבים באיור 23:



איור 23. מערכת לאשכול התנהגותי מבוסס HTTP [5].

בכל שלושת השלבים השתמשו באישכול היררכי מסוג single linkage.

הבחירה האינטואיטיבית באישכול היררכי הסתמכה על עובדה שאישכול היררכי מאתר אשכולות בצורות מגוונות ואינו מוגבל למרחב אוקלידי.

מחברי המאמר אף הריצו X-means, אישכול היררכי מסוג single linkage ואישכול היררכי מסוג complete linkage על הנתונים שנוצרו ע"י המערכת וגילו שאישכול היררכי מסוג single linkage הוא הכי מתאים לסוג הנתונים שמשתמשים במערכת.

כדי למצוא חיתוך דנדרוגרמה אופטימלי (חיתוך שמספק אשכולות הכי מופרדים והכי קומפקטיים), השתמשו ב DB index עבור בדיקת טיב האשכול [16].

#### אישכול גס (Coarse-grained Clustering):

נניח ש קבוצת דגימות תוכנות זדוניות ו לוג תעבורת HTTP שהתקבל לאחר הרצת לזמן נתון T. נתרגם כל לוג לווקטור המכיל את התכונות הסטטיסטיות הבאות שממדלות איך כל תוכנה זדונית משתמשת ב-Web:

1. כמות כוללת של בקשות HTTP.
2. כמות בקשות GET.
3. כמות בקשות POST.
4. אורך ממוצע של URLs.
5. כמות ממוצעת של פרמטרים בבקשה.
6. כמות ממוצעת של מידע שנשלח ע"י בקשות POST.
7. אורך ממוצע של תגובה.

בגלל השונות של התכונות בווקטורים, נורמל אוסף הנתונים כך שתוחלת התכונות היא 0 ושונות שווה ל-1.

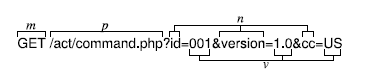
חילקו M לאשכולות גסים בעזרת אשכול היררכי מסוג single linkage והשתמשו במרחק אוקלידי כפונקציית מרחק בין הווקטורים.

#### אישכול עדין (Fine-grained Clustering):

לאחר חלוקת התוכנות הזדוניות לאשכולות יחסית גדולים (אשכול גס), חולק כל אשכול לקבוצות קטנות יותר. בשלב זה לוקחים בחשבון דמיון מבני בין רצפים של בקשות HTTP (בניגוד לדמיון סטטיסטי שהשתמשו באשכול גס). מדידת דמיון מבני בין זוגות לוגים של תעבורת HTTP הוא דבר יחסית יקר. לכן אישכול לפי דמיון מבני בוצע על כל אחד מאשכולות גסים במקום על קובץ דגימות שלם.

הגישה מבוססת על הבחנה ששתי דגימות של תוכנות זדוניות שמסתמכות על אותה אפליקציית Web ייגשו ל URL-ים בעלי מבנה דומה ובסדר דומה.

כדי לחשב את מידת הדמיון הגדירו מרחק בין שתי בקשות HTTP ו שנוצרו ע"י שתי דגימות תוכנות זדוניות. נתבונן באיור 24, כאשר m, p, n ו-v מציגות חלקים שונים של בקשת HTTP:



איור 24. מבנה של בקשת HTTP באשכול עדין. M=Method; p=Page; n=Parameter Names; v=Parameter Values [5].

* m - סוג הבקשה (GET, POST, HEADER וכו'). נגדיר פונקציית מרחק ששווה ל-0 אם שתי הבקשות ו- הן מאותו סוג, אחרת ערך הפונקציה יהיה 1.
* p – משמעותו page, כלומר החלק הראשון של URL הכולל כתובת ושם העמוד, אבל לא כולל פרמטרים. נגדיר להיות מרחק לוינשטיין מנורמל[[6]](#footnote-7) בין המחרוזות של כתובת ועמוד שמופיעות בשתי בקשות ו- .
* n – קבוצה של שמות פרמטרים (ז"א n={id, version, cc} בדוגמא באיור 24). נגדיר כמרחק Jaccard[[7]](#footnote-8) בין קבוצות פרמטרים בשתי הבקשות.
* v – קבוצה של ערכי פרמטרים. נגדיר להיות מרחק לוינשטיין מנורמל בין מחרוזות שנוצרו ע"י שרשור של ערכי פרמטרים (לדוגמא 0011.0US).

נגדיר מרחק כללי בין בקשות HTTP בצורה הבאה:

|  |  |  |
| --- | --- | --- |
|  |  | (6.1) |

כאשר הם משקלים שהוגדרו מראש. המשקלים נותנים יותר חשיבות למרחק בין סוגי הבקשות ועמודים ופחות חשיבות למרחק בין ערכי הפרמטרים.

עבור אשכול עדין, נגדיר מרחק בין שתי דגימות של תוכנות זדוניות בתור מרחק מינימלי ממוצע בין רצפים של בקשות HTTP משתי הדגימות. נריץ אלגוריתם אשכול היררכי מסוג single linkage.

#### איחוד אשכולות

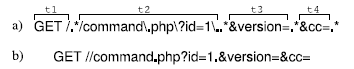
מחד, אשכול עדין נוטה לייצר אשכולות צמודים שמניבים חתימות ספציפיות עבור תוכנות זדוניות. מאידך, מטרת המערכת להנפיק חתימות של תוכנות זדוניות שיכולות לשמש לזיהוי כמה שיותר גרסאות תוכנות זדוניות עתידיות, תוך כדי שמירה על שיעור נמוך של תוצאות חיוביות שגויות. להשגת מטרה זו הוסיפו צעד שיפור נוסף שבו מאחדים ביחד אשכולות שנוצרו באשכול עדין ומתנהגים בצורה דומה מבחינת תעבורת HTTP שהם מייצרים. עבור כל אשכול שנוצר באשכול עדין מחשבים מרכז כובד שסוכם בקשות HTTP שבוצעו ע"י דגימות של תוכנות זדוניות באשכול. לאחר מכן ניתן להגדיר מידת מרחק בין מרכזי הכובד (ולכן גם בין אשכולות). שלב איחוד אשכולות הוא צעד של מטא – אשכול שבו מוצאים קבוצות אשכולות של תוכנות זדוניות שמאוד קרובות אחת לשנייה ומאחדים אותן כדי לייצר אשכולות יותר גדולים.

מרכזי כובד של אשכולות

נגדיר להיות אשכול של דגימות תוכנות זדוניות ו קבוצת לוגים של תעבורת HTTP שהתקבלו ע"י הרצה של כל דגימה ב . נגדיר מרכז כובד של להיות קבוצה של חתימות. כל חתימה הופקה ממאגר בקשות HTTP שנבחרו מתוך לוג של התעבורה ב-. בהתחלה נתאר אלגוריתם שמייצר פּוּלים של בקשות HTTP ואחרי זה נתאר איך מפיקים את החתימות מהפּוּלים שהתקבלו.

כדי לייצר קבוצת פּוּלים של בקשות , נבחר אחת הדגימות של תוכנות זדוניות באשכול להיות מרכז הכובד התחלתי. נניח שבחרנו ב- למטרה זו. ניקח בחשבון קבוצת בקשות HTTP בלוג תעבורת HTTP . נאתחל קבוצת פּוּלים ע"י כך שנשים כל בקשה בפּוּל נפרד (עד עכשיו ריק) . עכשיו נשתמש בהגדרת מרחק בין בקשות HTTP בנוסחה 6.1: לכל בקשה נמצא בקשה הכי קרובה מדגימת תוכנה זדונית אחרת . נוסיף לפול . נחזור על הפעולה עבור כל תוכנה זדונית . אחרי שהתהליך מסתיים, ופול מלא בבקשות HTTP, נחזור על התהליך כדי ליצור . נתחיל מבקשה עד שכל הפולים יתמלאו.

אחרי שהפולים התמלאו בבקשות HTTP, ניצור חתימה מכל פול ע"י הפעלת אלגוריתם Token-Subsequences [17]. חתימת Token-Subsequences היא רשימה מסודרת של תת מחרוזות קבועות שמשותפות לכל הבקשות בפול . לכן ניתן להציג חתימה בתור ביטוי רגולרי כאשר t-ים הם תת מחרוזות קבועות שמשותפות לכל הבקשות בפול . ניקח בחשבון רק חלק ראשון של כל בקשת HTTP בשביל יצירת חתימה, כלומר סוג הבקשה ו-URL כפי שמוצג באיור 25(a):



איור 25. דוגמת חתימה ((a, וגרסה טקסטואלית פשוטה שלה (b) [5].

אשכול

GET /test1.html HTTP/1.1

POST /index1.html HTTP/1.1

GET /test2.html HTTP/1.1

POST /index2.html HTTP/1.1

בקשות HTTP

בקשות HTTP

פולים

POST /index.html HTTP/1.1

Token-Subsequences

מרכז כובד S – אוסף חתימות

GET /test.html HTTP/1.1

Token-Subsequences

איור 26. תהליך יצירת חתימות ומרכזי כובד.

מטא – אישכול

אחרי שחישבנו מרכז כובד עבור כל אשכול עדין, נוכל לחשב מרחק בין זוגות מרכזי הכובד . בהתחלה נגדיר מרחק בין זוגות של חתימות ואחר-כך נרחיב את ההגדרה שתכלול קבוצות של חתימות. נניח ש- היא החתימה ו- היא חיבור טקסטואלי שטוח של תת מחרוזות קבועות בחתימה . לדוגמא היא גרסה טקסטואלית פשוטה של חתימה . ראה דוגמא באיור 25. נגדיר מרחק בין שתי החתימות בצורה הבאה:

|  |  |  |
| --- | --- | --- |
|  |  | (6.2) |

כאשר:

- פונקציה המבצעת התאמה מקורבת של ביטוי רגולרי [18]של חתימה למחרוזת , ומחזירה את מספר שגיאות ההתאמה. בפועל, שווה לאפס כאשר מכסה בצורה מושלמת(ז"א יותר גנרי מ-) , ושואף ל-1 ככל שחתימות ו- יותר שונות.

- אורך מחרוזת .

בהינתן הגדרה של מרחק בין שתי חתימות, נגדיר מרחק בין שני מרכזי כובד (זהו גם מרחק בין שני אשכולות) בתור מרחק ממוצע מינימלי בין שתי קבוצות חתימות. בצורה פורמלית:

|  |  |  |
| --- | --- | --- |
|  |  | (6.3) |

כשמחשבים מרחק בין שני מרכזי כובד, מתייחסים לחתימות שעונות לתנאי הבא: .

- גרסת טקסטואלית שטוחה של .

– אורך מחרוזת .

– סף האורך שהוגדר מראש.

משתמשים ב כדי לא להפעיל פונקציית על חתימות קצרות. לכן הן כנראה גנריות מדי שיתאימו בסבירות גבוהה לרוב בקשות HTTP (לדוגמא ) ויסיטו ערך המרחק לאפס בצורה מלאכותית.

לאחר מכן נפעיל שוב אלגוריתם אשכול היררכי מסוג single linkage כדי למצוא קבוצות של אשכולות שקרובים אחד לשני ויש לאחדם.

#### שימוש בחתימות

תהליך איחוד האשכולות שתואר בפרק ‏6.2.4.3 מהווה שלב אחרון של אישכול התנהגותי. השלב מייצג חלוקה סופית של דגימות תוכנות זדוניות לקבוצות תוכנות שמתקשרות עם Web בצורה דומה.

עכשיו עבור כל אשכול שקיבלנו בפרק 6.2.4.3 נוכל לחשב קבוצת חתימות מעודכנת בעזרת האלגוריתם שתיארנו בפרק זה. ניתן להכניס קבוצת החתימות למערכת לזיהוי פריצות בקצה של הרשת כדי לזהות בקשות HTTP זדוניות.

חשוב לציין שתוכנות זדוניות מסוימות יכולות להתחבר לאתרים ידועים (כמו פייסבוק או טוויטר) לצרכים זדוניים. לכן לא כדאי לסנן כל הבקשות HTTP שנשלחו לאתרים ידועים (במטרה להקטין מספר false positives), כי יכול להיות שדווקא בקשות אלה יעזרו ליצור חתימות מדויקות יותר שיניבו שיעור גילוי גבוה עם מספר קטן של false positives. במקום סינון בקשות HTTP לאתרים ידועים, משתמשים בתהליך signature pruning – בודקים קבוצת חתימות מול מאגר מידע גדול של תעבורה חוקית ונפטרים מחתימות שמייצרות false positives.

#### בדיקת טיב האשכול

שיטת למדידת טיב האשכול מבוססת על מדידת הלכידות (cohesion) של כל אשכול והפרדה (separation) בין אשכולות שונים. מחברי המאמר מדדו לכידות והפרדה במושגים של התאמה בין תגיות שהוקצו לתוכנות זדוניות באשכול ע"י תוכנות האנטי-וירוס.

נשים לב שבפרק ‏5.2 השתמשנו במדד cosine כמדד ללכידות של האיברים בתוך האשכול, כי כל אשכול היה מיוצג ע"י וקטור. במאמר זה כל אשכול מיוצג ע"י קבוצת חתימות. לכן מחברי המאמר נאלצו להגדיר מדד לכידות משלהם.

בשלב הראשון נדגים מהו גרף תגיות של תוכנות האנטי-וירוס (AV Label Graphs).

באופן פורמלי, גרף תגיות של תוכנות האנטי-וירוס הוא גרף ממושקל לא מכוון. בהינתן אשכול נגדיר:

|  |  |  |
| --- | --- | --- |
|  |  | (6.4) |

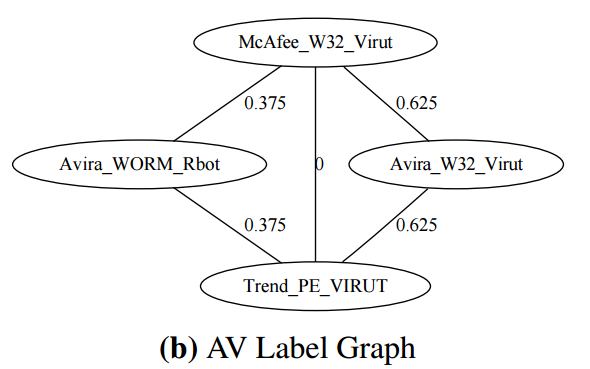
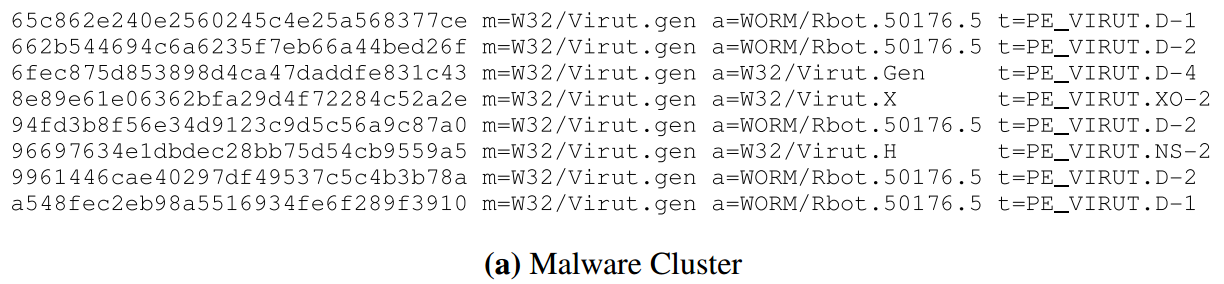
כאשר וקטור תגיות הוא קבוצה של תגיות שהוקצו לתוכנה זדונית ע"י תוכנות אנטיוירוס שונות. גרף תגיות של תוכנות האנטי-וירוס נבנה ע"י הוספת צומת עבור כל תגית . שני צמתים , מחוברים ע"י קשת ממושקלת רק אם התגיות ו נמצאות בוקטור כלשהו. המשקל של כל קשת מחושב באופן הבא:

|  |  |  |
| --- | --- | --- |
|  |  | (6.5) |

m - מספר וקטורים שמכילים גם וגם.

– מספר תוכנות זדוניות ב .

נתבונן בדוגמא לגרף תגיות באיור 27 [5]*:*



איור 27. דוגמא לווקטור תגיות שמייצג אשכול של תוכנות זדוניות(a). גרף תגיות של תוכנות האנטיוירוס (b) [5].

כל דגימה של תוכנה זדונית מתויגת ע"י 3 תוכנות האנטיוירוס שונות: McAfee(m), Avira(a) ו-Trend Micro(t).

כפי שניתן לראות צמתים McAfee\_W32\_Virut ו Trend\_PE\_Virut מחוברים ע"י קשת בעלת משקל אפס, מכיוון ש McAfee ו Trend Micro מסווגים כל דגימה באשכול כ W32/Virut ו PE\_VIRUT בהתאמה, ז"א m=n.

מצד שני לקשת בין McAfee\_W32\_Virut לבין Avira\_W32\_Virut יש משקל 0.625, כי במקרה זה m=3 ו n=8.

מדדים

בהינתן אשכול , נניח הוא גרף תגיות AV (גרף תגיות של תוכנות האנטיוירוס) שלו. ו המסלול הכי קצר בין הצמתים ו ב . אם לא קיים מסלול בין שני הצמתים ו , אז שווה לקבוע . הוא משקל של קשת גנרית .

**מדד לכידות** של אשכול מוגדר בצורה הבאה [5]:

|  |  |  |
| --- | --- | --- |
|  |  | (6.6) |

כאשר:

n - מספר דגימות של תוכנות זדוניות.

v – מספר תוכנות אנטי-וירוס שונות.

לפי הגדרה של גרף תגיות AV, וכן הוגדר . בפועל מדד לכידות יהיה שווה ל-1 כאשר כל תוכנות האנטיוירוס מקצות אותן תגיות עבור כל תוכנה זדונית באשכול . מצד שני המדד ישאף לאפס כאשר כל תוכנות האנטיוירוס יקצו תגיות שונות עבור כל תוכנה זדונית באשכול. לדוגמא, באיור 27 b)) מדד לכידות שווה ל-0.999, מכיוון ש-McAfee ו-Trend Micro מקצים אותה תגית לכל הדגימות באשכול.

בהינתן שני אשכולות ו ושני גרפים תגיות שלהם ו , נניח הוא אשכול שהתקבל ע"י איחוד של ו ו גרף התגיות שלו. לפי ההגדרה יכלול כל הצמתים ו .

**מדד הפרדה** מוגדר בצורה הבאה [5]:

|  |  |  |
| --- | --- | --- |
|  |  | (6.7) |

– המסלול הקצר ביותר בגרף בין הצמתים ו . הוגדר קודם. בפועל מדד ההפרדה נמצא בטווח . יהיה שווה ל-0 אם דגימות באשכולות ו יתויגו לאותה משפחה ע"י כל תוכנות האנטיוירוס.

### תוצאות ניסוייות

#### אשכול התנהגותי מבוסס HTTP

###### **נתונים**

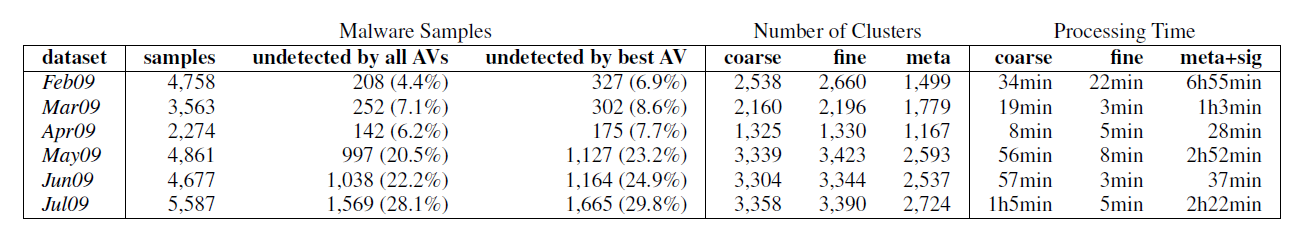
המאגר מורכב מ 25720 דגימות של תוכנות זדוניות שונות שכל אחת מהן מייצרת לפחות בקשת HTTP אחת על מכונת קורבן. דגימות התוכנות הזדוניות נאספו במהלך 6 חודשים מפברואר עד יולי 2009 ממקורות שונים כגון MWCollect, Malfease וממקורות מסחריים [5]. כל דגימה הורצה במשך שבמהלך זמן זה הקליטו תעבורת HTTP לצרכי אשכול התנהגותי.

**פרמטרים**

הציבו בנוסחה 6.1 את הערכים הבאים: . הגדירו . השתמשו ב-10 בקשות HTTP ראשונות שנוצרו ע"י כל תוכנה זדונית כדי לבצע אשכול עדין ומטא-אשכול. ביצעו התאמה של ביטויים רגולריים תוך כדי שימוש בספריית TRE. כל הניסויים בוצעו על מכונת 4-core 2.67GHz Intel Core-i7 עם 12 GB RAM.

**תוצאות האישכול**

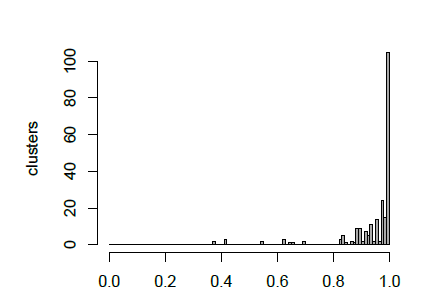
כל דגימה של התוכנה הזדונית נסרקה עם שלוש תוכנות אנטי-וירוס: McAfee, Avira, Trend Micro. בחינת טיב האישכול התבסס על תגיות AV. בטבלה 6.1 מוצגות תוצאות האשכול:



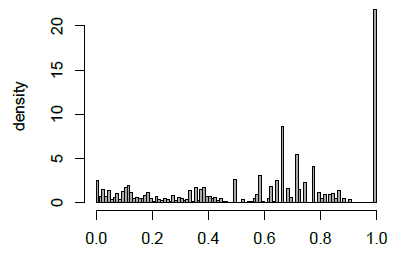
טבלה 6.1. סיכום תוצאות אישכול [5].

עמודה ראשונה ושנייה בטבלה מראות מספר דגימות תוכנות זדוניות שנאספו בכל חודש. כפי שניתן לראות מעמודה שלישית ורביעית, כל מאגר של תוכנות זדוניות הכיל דגימות שלא זוהו ע"י אף תוכנת האנטיוירוס. Best AV – תוכנת האנטיוירוס שזיהתה מספר הכי גדול של דגימות בסה"כ. Meta+sig – זמן שלוקח לבצע מטא-אשכול וליצור חתימות [5].

כדי למדוד את טיב האשכול השתמשו במדדי לכידות והפרדה.



איור 28. היסטוגרמה של ערכי הלכידות שחושבו עבור כל האשכולות שנוצרו ממאגר של Feb09. היסטוגרמה זו לוקחת בחשבון רק אשכולות שמכילים 2 או יותר דגימות של תוכנות זדוניות. לאשכולות שמכילם רק דגימה אחת ערך הלכידות שווה ל-1 בהגדרה [5].



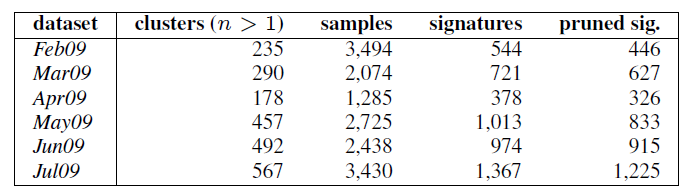
איור 29. התפלגות של ערכי ההפרדה שחושבו בקרב האשכולות שנוצרו ממאגר של Feb09 [5].

אידאלית, רצוי שערך הלכידות עבור כל אשכול יהיה קרוב ל-1. איור 28 מאשר אפקטיביות של שיטת האשכול. רוב האשכולות שנוצרו ע"י שיטת אשכול שתיארנו היו קומפקטיים במובן של גרף תגיות AV. עובדה זו מוכיחה התאמה טובה בין תוצאות האשכול שקיבלנו לבין התגיות שניתנו לתוכנות זדוניות ע"י תוכנות האנטיוירוס. היינו רוצים גם שכל זוגות האשכולות יהיו מופרדים עם ערך הפרדה של 1. איור 29 מראה שרוב הזוגות מופרדים טוב. באופן כללי התפלגויות של ערכי לכידות והפרדה מראות שרוב האשכולות שקיבלנו הם קומפקטיים ומופרדים די טוב במושגים של תגיות של תוכנות האנטי-וירוס.

לסיכום, אישכול היררכי אגלומרטיבי מסוג single linkage הוא בחירה טובה עבור המערכת. חישוב אשכולות עבור תוכנות זדוניות מבוססות HTTP לוקח שעות ספורות. והאשכולות שנוצרו היו קומפקטיים ומופרדים.

#### חתימות

הרעיון הוא להפיק חתימות מדגימות תוכנות זדוניות חדשות ולמדוד אפקטיביות של החתימות בזיהוי תעבורת HTTP זדונית שנוצרה ע"י תוכנות זדוניות עכשוויות ועתידיות.



טבלה 6.2. יצירת חתימות אוטומטית וסינון תוצאות. זמן הדרוש ליצירת חתימות מפורט בטבלה 6.1 (עמודה meta+sig) [5].

טבלה מסכמת תוצאות של תהליך אוטומטי ליצירת חתימות. לקחו בחשבון רק אשכולות שמכילים לפחות 2 דגימות. חתימות של אשכולות בעלי דגימה אחת יותר מדי ספציפיות ולא מייצגות משפחה של תוכנות זדוניות.

נתאר את תהליך הסינון (signature pruning) בקצרה:

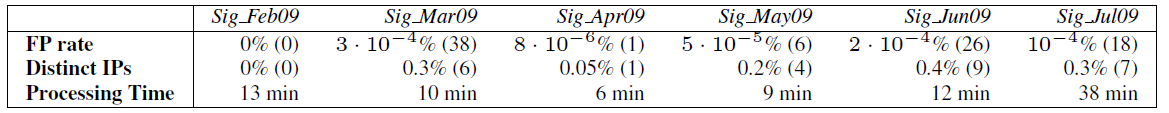
1. אספו מאגר של תעבורת HTTP חוקית במשך יומיים. ( בקשות HTTP מ-2010 קליינטים לאלפי אתרי Web שונים).
2. השתמשו בטכניקות אוטומטיות קיימות לזיהוי תעבורת HTTP זדונית והשתמשו באנליזה ידנית כדי לוודא שהתעבורה שאספו נקיה ככל האפשר.
3. תעבורת HTTP שאספו ביום הראשון שימשה לסינון. ותעבורה של היום השני שימשה לשערוך של כמות ה false positives שנוצרו ע"י החתימות שנשארו אחרי הסינון.

חתימות Feb09 שנשארו אחרי סינון Sig\_ Feb09

תוצאות חיוביות שגויות (False Positives)

Sig\_ Feb09 – חתימות Feb09 שנשארו אחרי סינון.

Sig\_Mar09 – חתימות Mar09 שנשארו אחרי סינון. וכך הלאה.



טבלה 6.3. תוצאות חיוביות שגויות שנמדדו על תעבורה חוקית של יום אחד (בערך 12M של בקשות HTTP מתוך 2010 IP-ים שונים) [5].

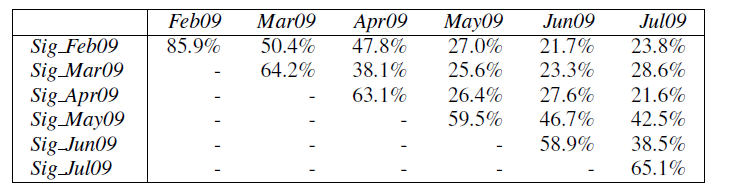
שורה ראשונה בטבלה כוללת שיעור תוצאות חיוביות שגויות שנמדד כמספר התראות שנוצרו ע"י קבוצת חתימות נתונה מחולק במספר בקשות HTTP במאגר חוקי. המספרים בתוך סוגריים מייצגים מספר אבסולוטי של התראות שנוצרו. שורה שנייה כוללת שיעור של כתובות IP שתעבורת HTTP שלהן התאימה לחתימות שלנו. המספרים בתוך סוגריים מייצגים מספר אבסולוטי של כתובות IP שעבורן הופעלה התראה.

כפי שניתן לראות , החתימות שהמערכת יצרה גורמות לשיעור נמוך של תוצאות חיוביות שגויות.

אבחנה חשובה נוספת היא שזמן עיבוד של 12M בקשות חוקיות, שנאספו במהלך יום אחד, הוא דקות ספורות בלבד. לכן המערכת תחזיק מעמד בזמן אמת.

שיעור זיהוי

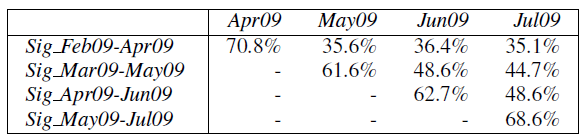
כאן מדדו יכולת של החתימות לזהות דגימות עכשוויות ועתידיות. לדוגמא, בהינתן חתימות מסוננות של Feb09, השוו אותן לתעבורת HTTP שנוצרה ע"י תוכנות זדוניות ב Feb09, Mar09, Apr09 וכו'. בשביל זיהוי הדגימה מספיק שתעבורת HTTP שלה תגרום להתראה אחת לפחות. ניתן לראות תוצאות הגילוי בטבלה:



טבלה 6.4. שיעור זיהוי תוכנות זדוניות עכשוויות ועתידיות בעזרת חתימות [5].

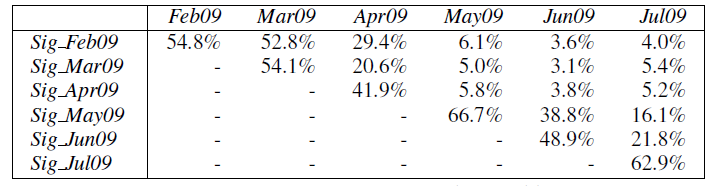
כפי שניתן לראות, ניתן להכליל קבוצות החתימות לתוכנות זדוניות חדשות שלא נאספו קודם. מאחר והחתימות מנסות לתאר התנהגות של משפחות תוכנות זדוניות ולא התנהגות של דגימות אינדיבידואליות. למרות שגרסאות שונות של תוכנות זדוניות מאותה משפחה יכולות להיווצר בקצב גבוה (לדוגמא תוך כדי שימוש ב packing tools [5]), כשהתוכנות רצות, הן מתנהגות בצורה דומה. לכן, ניתן לזהותן ע"י החתימות ההתנהגותיות. המערכת אינה מזהה דגימות ייחודיות שמתנהגות שונה מכל קבוצות תוכנות זדוניות שזוהו ע"י אלגוריתם אשכול התנהגותי. אף-על-פי-כן ניתן לראות מטבלה 6.4 שאם דואגים לעדכן חתימות מעת לעת, אפשר להשיג שיעור זיהוי יחסית גבוה על דגימות עכשוויות ועתידיות.

טבלה מדגימה שאיחוד חתימות של מספר חודשים יחדיו, משפר את שיעור הזיהוי.



טבלה 6.5. שיעור זיהוי תוכנות זדוניות עכשוויות ועתידיות בעזרת חתימות. איחוד של 3 חודשים [5].

טבלה מוכיחה שהחתימות שנוצרו ע"י המערכת הן השלמה טובה והכרחית לתוכנות האנטיוירוס (כי תוכנות האנטיוירוס נוטות לשיעור גבוה של false negatives):



*טבלה 6.6. שיעור זיהוי של תוכנות זדוניות שלא זוהו ע"י שלושת תוכנות האנטיוירוס (McAfee, Trend, Avira)* [5]*.*

#### השוואה עם שיטות אחרות

מטרות

* **לבדוק האם ההחלטה להשתמש באישכול בעל 3 שלבים הייתה נכונה.**
* **להשוות את המערכת מול מערכות לזיהוי תוכנות זדוניות קיימות.**

השיטות שהשתתפו בהשוואה

* **מערכת לזיהוי תוכנות זדוניות של** Bayer&Milani [19].
* Polygraph [20].
* **מערכת שמשתמשת באישכול עדין (**fine-grained**) בלבד.**
* **מערכת בעלת 3 שלבי אישכול המקורית.**

הגדרת בסיס הנתונים

**נערכו שלושה ניסויים. בכל ניסוי הוגדר בסיס נתונים משלו.**

ניסוי 1:

**השתמשו באותו בסיס הנתונים שהוגדר בסעיף "נתונים" בפרק ‏6.2.5.1.**

ניסוי 2:

מחברי המאמר שלחו קרוב ל-4000 דגימות ממאגרי תוכנות זדוניות Feb09 ו-May09 למחברי המאמר ה [19].

3 המערכות שהשתתפו בניסוי השתמשו באותו המאגר המצומצם ליצירת אשכולות.

בדיקת החתימות התבצעה על מאגרי תוכנות זדוניות מלאים (סעיף "נתונים" בפרק ‏6.2.5.1).

ניסוי 3:

תת קבוצה של 49 דגימות תוכנות זדוניות ממשפחת Virut.

מדדים להשוואה

* **מדידת שיעור זיהוי.**
* **מדידת זמן אישכול.**
* NMDR – next month detection rate**. שיעור זיהוי של תוכנות זדוניות עתידיות ע"י חתימות שנוצרו בהתבסס על תוכנות זדוניות עכשוויות.**

תוצאות ומסקנות

ניסוי 1:

**השוואת מערכת שמשתמשת באישכול עדין (**fine-grained) **בלבד מול מערכת בעלת 3 שלבי אישכול המקורית.**

טבלה מציגה NMDR עבור חתימות שנוצרו ע"י הרצת אשכול עדין בלבד.



טבלה 6.7. תוצאות שהתקבלו משימוש באשכול עדין (fine-grained). [5].

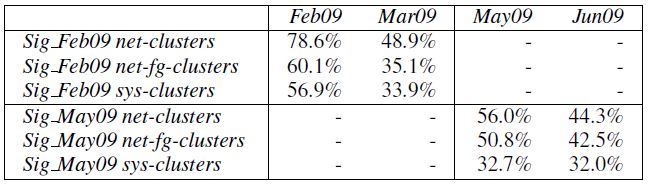
לדוגמא, הריצו אישכול עדין על מאגר Feb09. לאחר מכן יצרו קבוצות חתימות מכל אשכול ובדקו אותן על תעבורת HTTP שנוצרה ע"י דגימות תוכנות זדוניות ממאגר Mar09. נשים לב ש NMDR עבור Jul09 אינו מוגדר מכיוון שלא נאספו תוכנות זדוניות באוגוסט 2009.

אם נשווה טבלה וטבלה, נראה שחתימות שנוצרו ע"י אשכול המורכב מ-3 שלבים הניבו שיעור זיהוי יותר גבוה מאלה שנוצרו ע"י אשכול עדין בלבד.

לכן, ההחלטה להשתמש באישכול בעל 3 שלבים הייתה נכונה.

ניסוי 2:

**השוואת מערכת שמשתמשת באישכול עדין (**fine-grained**) בלבד, מערכת בעלת 3 שלבי אישכול המקורית ו מערכת לזיהוי תוכנות זדוניות של** Bayer&Milani [19].



טבלה 6.8. שיעור זיהוי תוכנות זדוניות בעזרת חתימות שנוצרו מאשכול המורכב מ-3 שלבים (net-clusters), אשכול עדין בלבד (net-fg-clusters) ואשכול שתואר ב [19] (sys-clusters) [5].

ניתן לראות בטבלה שחתימות שנוצרו ע"י אישכול המורכב מ-3 שלבים הניבו שיעורי זיהוי גבוהים יותר בכל המקרים.

בנוסף, המערכת שמבצעת אשכול עדין בלבד הניבה תוצאות יותר טובות מזו של Bayer&Milani [19].

אחת הסיבות לכך, היא עובדה שהמערכת היותר ישנה כבר ידועה בקרב מפתחי תוכנות זדוניות והם פיתחו כלים לעקוף אותה.

ניסוי 3:

בנוסף הריצו Polygraph [20] על תת קבוצה של 49 דגימות תוכנות זדוניות ממשפחת Virut. Polygraph רץ במשך יותר משבועיים ולא הסתיים.

לכן, גם כאן אישכול המורכב מ-3 שלבים הניב תוצאות טובות יותר.

#### מקרה בוחן

ניטרו במהלך 4 ימים תעבורה של חברה גדולה המורכבת מאלפי צמתים שמריצים תוכנות האנטיוירוס מסחריות. הציבו חתימות Sig\_Jun09 ו Sig\_Jul09 לניטור תעבורה אל web proxy של החברה. 2 הימים הראשונים שימשו לסינון החתימות (פרק ‏6.2.5.2) ו-2 ימים אחרונים שימשו למדידת תוצאות חיוביות שגויות של קבוצת חתימות מסוננת. במהלך יומיים ראשונים נמצאו 32 חתימות שגרמו לתוצאות חיוביות שגויות וביטלו אותם. תוך יומיים אחרונים החתימות שנותרו יצרו רק 12 תוצאות חיוביות שגויות. אותן החתימות עזרו לזהות קרוב ל-50 מכונות נגועות.

## זיהוי בוטנט בעזרת ניתוח לוגים ברשת ארגונית [15]

### הקדמה

מערכות ארגוניות גדולות מותקפות בעזרת תוכנות זדוניות, בפרט בוטנט, שמקורן יכול להיות אקראי (מפורסם לכולם) או ייעודי למערכת הארגונית. תוכנות אלו משתפרות עם הזמן ונכנסות למרוץ עם תוכנות האנטיוירוס שתפקידן לנטרל או להתריע. מצד שני משתמשי המערכת גם מהווים סכנה לארגון כאשר הם מבצעים פעילות הנוגדת את התקנון הארגוני. ישנם פתרונות האוכפים את החוקים האלו אך גם הם לא מושלמים.

קושי נוסף שעולה במערכת ארגונית גדולה היא כמות הלוגים הנשמרים במערכת. על מנת לנתח כמות אדירה של לוגים יש צורך בתכנון, זמן ומשאבי מערכת רבים. הלוגים באים ממקורות שונים לכן הם חייבים לעבור עיבוד לפני שנוכל להשתמש בהם.

### מטרות המערכת

מערכת Beehive מנסה לזהות שני סוגי תרחישי איומים על הרשת הארגונית:

1. מחשב המכיל תוכנה זדונית או מהווה חלק מבוטנט. מחשב זה יציג התנהגות טיפוסית למחשב נגוע בבוטנט: ניסיונות להתקשר לשרתי שליטה, ביצוע התקפות Denial of Service, התקשרויות עם מחשבים לא מזוהים ושליחת מידע אליהם.
2. משתמש של המחשב מבצע פעילות בניגוד לתקנון הארגוני. שיתוף קבצים, שימוש בשירותים להתכתבויות מידיות, גלישה באתרים המכילים תכנים למבוגרים בלבד, שימוש במשחקים ועוד...

### ארכיטקטורת המערכת

מערכת Beehive תנתח בצורה אוטומטית את הלוגים שנוצרו ברשת הארגונית ותזהה פגיעויות ופעילות הנוגדות את התקנון של הארגון בשלושה שלבים:

1. הכנת נתוני לוגים לעיבוד.
2. יצירת ווקטורים של מאפיינים מהלוגים.
3. ביצוע אשכול ומציאת פעילויות לא רצויות במערכת.

### המידע הגולמי

המערכת מנטרת נקודות רבות בארגון EMC. מקורם של הלוגים נמצא ב:

1. Web Proxies – נקודות שנמצאות בין הרשת הפנימית של הארגון לרשת החיצונית.
2. DHCP servers – לוגים שמכילים מידע על שיוך כתובות IP למחשבים.
3. שרתי VPN – שומרים לוגים על התקשרויות למערכת הארגונית.
4. Windows Domain Controllers – לוגים המכילים ניסיונות אימות זהויות במערכת.
5. Antivirus Software – לוגים שמגיעים מאנטי-וירוסים שפרוסים במערכת הארגונית.

המערכת מסתמכת בעיקר על הלוגים שמגיעים מ Web Proxies, לוגים שמכילים מידע על תקשורת HTTP. לוגים אלו מכילים את המידע הבא:

1. IP Header – כתובת IP ופורט של מקור והיעד.
2. Transport Header – דומיין, URL, סטאטוס של הבקשה, referrer, מוניטין הדומיין, קטגוריה של הדומיין, user-agent.
3. Connection Attribute – מספר בתים שנשלחו, מספר בתים שהתקבלו, network policy.

כמות הלוגים המתקבלת ביום במערכת הארגונית EMC היא 1.4 מיליארד או טרה-בייט בנפח אחסון. מידע זה הוא לא מעובד ולכן הוא יעבור שלבים של נורמליזציה לפני שנוכל להשתמש בו הלאה לכריית מאפיינים ואשכול.

### נורמליזציה

לפני שימוש בנתונים יש לבצע נורמליזציה לפי פרמטרים הבאים:

1. זמן – חותמות הזמן שנמצאות בלוגים של המערכת לא מסונכרנות, הפורמט שונה ואזורי זמן שונים.
2. קישור מחשב לכתובת IP משתנה – המערכת משתמשת בנתוני לוג של DHCP שמאפשרים לה לראות לאורך זמן, איזה כתובת הוקצתה לאיזה מחשב באיזה פרק זמן.
3. קישור מחשב לכתובת IP קבועה – המערכת משתמשים ברשימה של כל הכתובות IP הקיימות ומחסירה ממנה את כל כתובות IP הדינאמיות. הכתובות שנשארו עוברים ip resolving ונשמרים ככתובות סטטיים S. ביום למחרת חוזרים על הפרוצדורה – אם יש כתובת ב-S ששויכה למחשב אחר, מסירים את הכתובת מ-S.
4. מחשבים ייעודיים – אלו מחשבים שיש להם משתמש אחד. מערכת Beehive מתרכזת במחשבים מסוג זה. הנתונים על מחשבים ייעודיים נלקחים ממערכת הבקרה של הארגון. Microsoft Windows Domain Controllers מספקים מידע על כל אימותי משתמשים שנעשה. המחשב נקבע כייעודי אם 95% מהפעולות אימות משתמשים מבוצעות על ידי אותו משתמש.

### מאפיינים

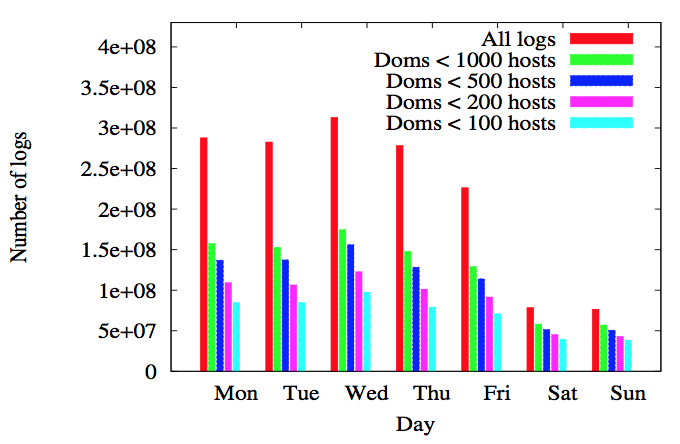
עבור כל מחשב ייעודי נגדיר 15 מאפיינים. ווקטור מאפיינים זה מחושב מדי יום מחדש. המאפיינים מבוססים על הגדרות התקנון של הארגון ודרישות המערכת Beehive. ניתן לחלק את המאפיינים לקטגוריות:

1. מבוססי יעד:
   1. (1) יעדים חדשים – מספר היעדים החדשים מחוץ לרשת הארגונית ביום. אחרי תקופה של חודש כל יעד חיצוני אשר לא היה בלוגים נחשב כיעד חדש. מאפיין זה התגלה כבעייתי משום שלקח 15 שעות על מנת לעבור על כמות לוגים יומית – 300 מיליון. הנחה שלא יהיו הרבה יעדים חדשים התגלתה כשגויה. בכל יום כ 30% מהתעבורה עוברת ביעדים חדשים. סיבה לכך היא שחלק גדול מהיעדים הוא CDNs (Content Delivery Networks) ושרותי Cloud (שמשתמשים במחרוזות רנדומאליות בתור subdomain-ים) או כתובות ip ששייכים לאתרים פופולריים (Google, Twitter, Facebook).

הוגדרו מאפיינים מסננים על מנת להוריד את מספר הלוגים. יעדים פופולריים ברשת הארגונית הוסרו אם רמת הפופולריות שלהם עלתה על ערך סף, ראה איור 30. פופולריות של היעד נמדדת לפי כמות המחשבים הפנימיים בחברה שניגשו לאותו היעד בפרק זמן מסוים (במקרה שלנו שבוע אחד). איור 30 מראה את כמות הלוגים שנוצרו במשך שבוע ע"י יעדים שפנו אליהם פחות מ 100 מחשבים פנימיים (Doms < 100), יעדים שפנו אליהם פחות מ 200 מחשבים פנימיים (Doms < 200) וכך הלאה. קביעת ערך סף ל 100 מחשבים פנימיים בלבד הורידה 74% מהלוגים.

דומיינים קוצרו עד רמה שניה, כלומר abc.google.com ו xyz.google.com מתייחסים לאותו דומיין google.com.

המערכת לא תבצע resolving של יעדים שמהווים כתובת IP, כי רוב הפניות לדומיינים חוקיים מתבצעות לפי שם הדומיין. אם הכתובת לא נמצאת ברשימות הכתובות המותרות אז היא תחשב כחדשה.



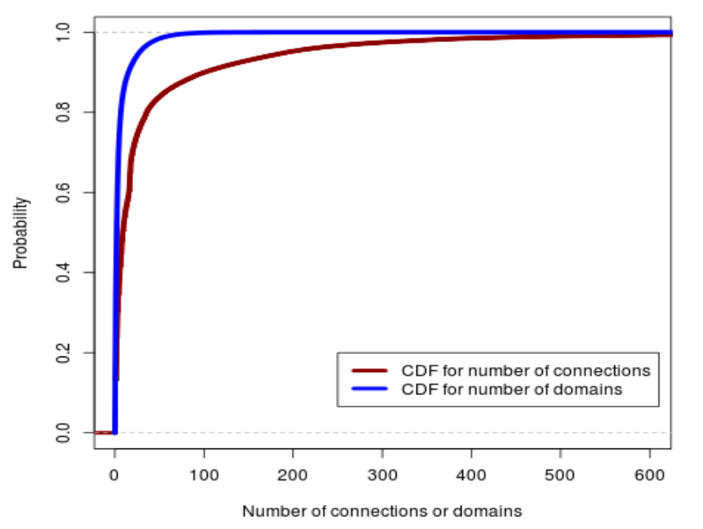
איור 30. כמות לוגים שנוצרו במשך שבוע ע"י יעדים שונים [15].

* 1. (2) יעדים חדשים ללא referrer – מספר יעדים ש referrer שלהם חסר – כלומר הכתובת של האתר או השירות המפנה לא קיימת. בד"כ משתמשים מופנים לאתרים חדשים ע"י מנועי חיפוש, אתרי חדשות או פרסומות. במקרה כזה נצפה שהאתר המפנה יהיה ברשימת כתובות מותרות. מחשבים שהופנו ע"י אתר או שירות לא ידועים או ללא אתר מפנה בכלל נחשבים חשודים.
  2. (3) כתובות IP לא פופולריות – מספר כתובות IP לא פופולריות מתוך היעדים שיש למחשב. בדרך כלל משתמשים ניגשים לאתרים בעזרת שם האתר. גישה לאתר בעזרת כתובת IP יכולה להיחשב חשודה.
  3. (4) אחוז כתובות IP לא פופולריות – אחוז כתובות IP לא פופולריות מתוך היעדים שיש למחשב.

1. מבוססי הגדרות מקור
   1. (5) מחרוזות user-agent חדשות – בהעדר נתונים על הגדרות המחשבים, ניקח את הנתון של user-agent שמתקבל מבקשות ברשת. מחרוזת זו מכילה מידע על התוכנה שממנה יצאה התקשרות. בונים היסטוריה של מחרוזות במשך חודש אחד. לאחר מכן מחרוזת תחשב כחדשה אם היא שונה משמעותית מהמחרוזות שנשמרו בהיסטוריה. מידת השוני בין שתי מחרוזות נקבעת לפי מרחק לוינשטיין. זהו מרחק שמבוסס על כמות הפעולות שיש לבצע במחרוזת כדי להשוות אותה למחרוזת השנייה. הדבר נועד למנוע מצבים בהם התוכנה שיוצרת את התקשורת מעלה גרסה ונחשבת לחדשה.
2. מבוססי תקנון הארגון – רשת ארגונית מאפשרת גישה לאתרים מסוימים שיש בהם אמון. אתרים אחרים נחסמים בשל הפרות תקנון הארגון או בגלל שאתרים אלה הוגדרו כיעד לא ידוע בתוך הרשת הארגונית. אם המשתמש מסכים על אחריותו להיכנס לאתר שנחסם אז התנהגותו חשודה. ששת המאפיינים הבאים מתארים את כל התרחישים האפשריים:
   1. (6) דומיינים שלא קיבלו הסכמה.
   2. (7) חיבורים שלא קיבלו הסכמה.
   3. (8) דומיינים שקיבלו הסכמה.
   4. (9) חיבורים שקיבלו הסכמה.
   5. (10) דומיינים שביקשו הסכמה.
   6. (11) חיבורים שביקשו הסכמה.
3. מבוססי תעבורת רשת – נגדיר spike כעלייה חדה בפעילות המערכת בחלון זמן של דקה. אם במשך חלון זמן זה המחשב יצר מספר חיבורים למספר אתרים גבוה מערך הסף אז קרה spike. Burst הוא האינטרוול הזמן שבו כל דקה היא spike. על פי המידע, ראה איור 31, שנמצא בלוגים, 90% מהמחשבים לא יוצרים יותר מ 101 התקשרויות ל 17 אתרים חדשים בדקה. אלו הם ערכי הסף עבור spike. עבור burst נחשיב ערכי סף שונים מ :spike ניקח אחוזון 75 של כל חלונות זמן של דקה עבור כל ההוסטים. לפי איור 31 זהו 26 התקשרויות ו 6 אתרים חדשים בדקה.

על פי הגדרות אלה נחשב את ארבעת המאפיינים עבור כל מחשב.

* 1. (12) מספר spike-ים בכמות חיבורים.
  2. (13) מספר spike-ים בכמות אתרים.
  3. (14) אורך burst הארוך ביותר במספר חיבורים.
  4. (15) ארוך burst הארוך ביותר במספר אתרים.



איור 31. פונקציית הצטברות מול כמות החיבורים ואתרים במשך אינטרוול של דקה אחת [15].

### אישכול

נשתמש באשכול על מנת למצוא מחשבים פגועים. ההנחה ברשת הארגונית היא שהרוב הגדול של העובדים יבצעו אותן פעולות. מחשבים אשר יראו שונים מאחרים יקובצו לתוך אשכול משלהם ויישלחו לבדיקה ידנית של אבטחת מידע.

בהינתן 15 מאפיינים עבור כל מחשב ייעודי נגדיר את הווקטור . נבצע המרת PCA [15] לווקטורים. מטרת ההמרה היא להוריד תלויות בין המאפיינים ואת מספר הממדים בווקטור. נניח שביצוע המרת PCA מוריד את מספר הממדים ל m אשר משמרים 95% מהשונות של המידע המקורי. נפעיל אלגוריתם אשכול שהוא וריאציה של K-means ללא צורך של בחירה של מספר אשכולות:

1. נסמן באופן אקראי ווקטור אחד כמרכז הכובד של האשכול ונשייך את כל שאר הווקטורים אל האשכול.
2. נסמן את הווקטור הכי מרוחק ממרכז הכובד וניצור אשכול חדש. כל הווקטורים ישתייכו לאשכול הקרוב ביותר אליהם
3. נחשב את המרחק הממוצע בין כל מרכזי הכובד של האשכולות ונחלקו ב 2, נסמן ב d. אם קיים ווקטור שהמרחק שלו ממרכז האשכול גדול יותר מ d נחזור על סעיף 2.

נשתמש בפונקציית מרחק L1 [15]:

|  |  |  |
| --- | --- | --- |
|  |  | (6.8) |

– אחד מהערכי המאפיינים

אשכול מבוצע כל יום על כ 30,000 מחשבים ייעודיים פעילים בימי עבודה ו 10,000 מחשבים ייעודיים פעילים במשך סופי שבוע. תוצאות האשכול לאחר האיטרציה הראשונה הם בדרך כלל אשכול אחד גדול שמכיל את כל מחשבים עם התנהגות נורמלית וכמה אשכולות עם מחשבים בודדים בעלי התנהגות חשודה. במקרים מיוחדים נוצרים רק שני אשכולות אחד שמכיל רק מחשב אחד ואשכול שני שמכיל את כל שאר האשכולות. במקרה כזה נפעיל PCA על הווקטורים של האשכול הגדול ולאחר מכן נבצע אשכול מחדש. התהליך יימשך עד שהמערכת תמצא 50 מחשבים חשודים באשכולות נפרדים.

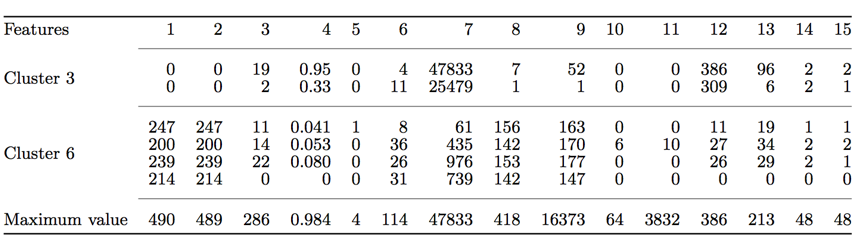
### תוצאות

מערכת Beehive עבדה שבועיים ברציפות ברשת הארגונית של EMC. המערכת זיהתה 784 תקריות שסומנו כחשודות כאשר מתוכן רק 8 אותרו על ידי מערכות אבטחת מידע שהיו מותקנות בארגון. בטבלה 6.1 נוכל לראות בפירוט את ערכי הווקטורים שנמצאים באשכולות 3 ו 6.

לדוגמא נוכל לראות שאשכול 3 מאופיין על ידי ערכים גבוהים מהרגיל של מאפיינים הבאים: מספר גבוה של חיבורים חסומים (7), spike בחיבורים (12) ודומיינים (13). מחשבים באשכול זה אובחנו כמפירים תקנון החברה יותר מדי פעמים.

אשכול 6 ייחודי בגישה לכתובות חדשות (1)(2) , spike בחיבורים (12) לדומיינים חסומים (6)(7)(8)(9) ודומיינים שביקשו גישה (10)(11). ניתוח ידני של אשכול 6 מראה שאשכול זה מכיל בוטנטים שמשתמשים ב DGA.

מאפיין שהתגלה יעיל מאוד במציאת בוטנטים מסוג DGA הוא הכתובות החדשות ששייכות למשפחת מאפיינים של היעדים.

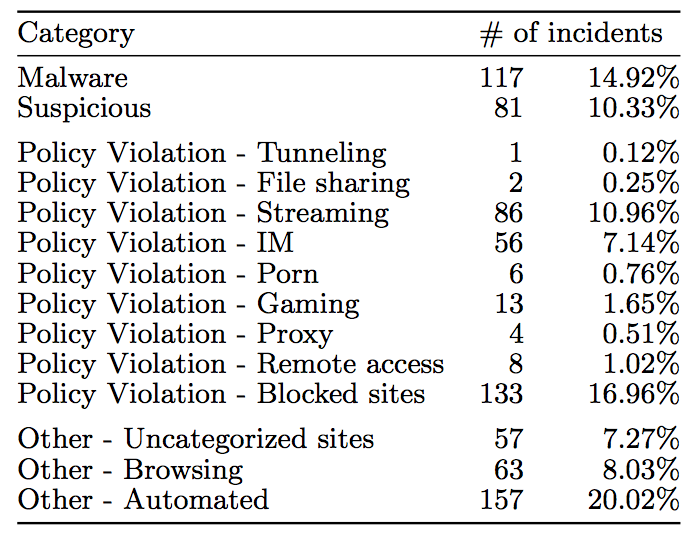


טבלה 6.1. דוגמאות של אשכולות 3 ו 6. כל שורה מכילה וקטור שנמצא באחד מהאשכולות [15].

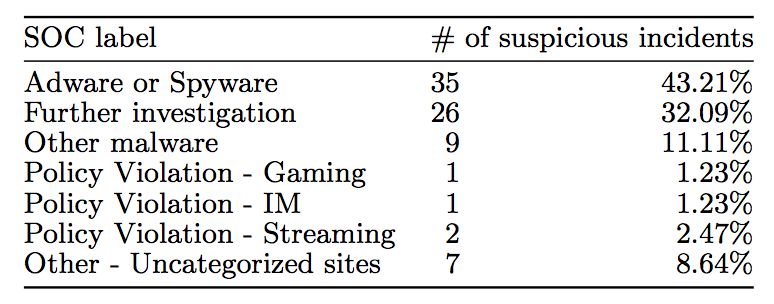
האשכולות שנמצאו עברו שני שלבים של ניתוח ידני.

ניתוח ידני ראשון התבצע על ידי צוות Beehive. ניתוח ידני של האשכולות נמצא בטבלה 6.2. 25.25% מהתקריות סווגו כחשודות או זדוניות ומתוכן 14.92% התגלו כתוכנות זדוניות, 39.41% סווגו כתקריות הנוגדות את התקנון של הארגון. 20.02% סווגו כפעילות אוטומטית כמו כריית ביטקוין אך חלק התגלו כגלישה לאתרים כבדים שיצרו המון בקשות. 8.03% מהתקריות סווגו כחיבורים שקיבלו הסכמה מהמשתמש. חיבורים אלו לא נמצאו זדוניים לכן סווגו לקטגוריית גלישה.

בשלב הסופי התקריות החשודות סווגו על ידי מומחי אבטחת מידע של הארגון. את התוצאות ניתן לראות בטבלה 6.3. יותר ממחצית מהתקריות החשודות סווגו כתוצאה של תוכנות זדוניות. 32.09% נוספים סווגו כתקריות משמעותיות שיש צורך לחקור אותן.



טבלה 6.2. קטגוריות לאחר ניתוח ידני של האשכולות [15].



טבלה 6.3. תוצאות סיווג של תקריות חשודות [15].

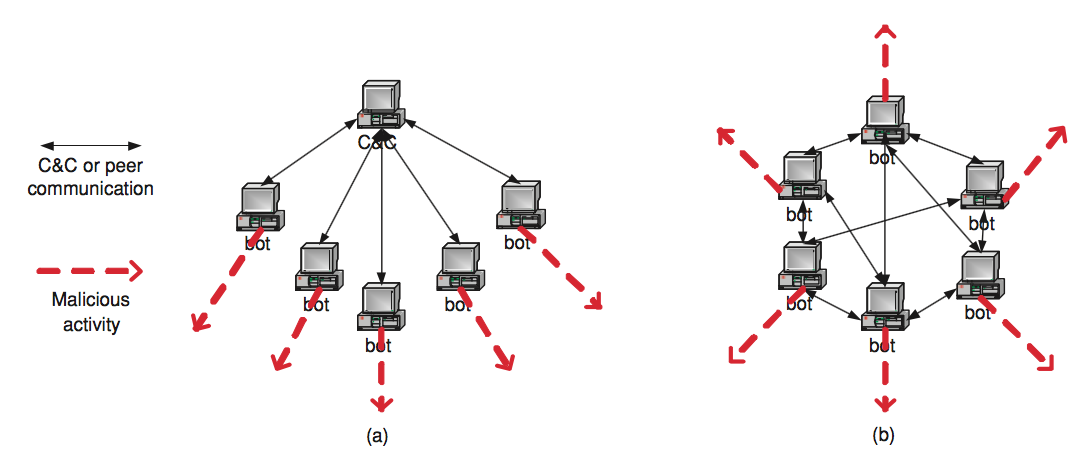
### מסקנות

מערכת Beehive סיווגה בהצלחה 784 תקריות מתוכם רובן היו רלוונטיות לאבטחת מידע של הארגון. כמה מאפיינים שהמערכת השתמשה הצליחו לגלות מחשבים הנגועים בבוטנט מבוסס DGA שמחלקת אבטחת המידע בארגון לא הצליחה לאתר בעזרת כלים קיימים.

## מערכת BotMiner [4]

### הקדמה

הרעיון מאחורי מערכת BotMiner הוא לא להיות תלוי בגורמים שמשתנים כל הזמן כמו פרוטוקול, ארכיטקטורה ושרתי שליטה (איור 32), מודל הדבקה או גיוס של מחשבים חדשים, שינוי כתובות שרתי שליטה. המערכת תפעל בשני מישורים. מישור אחד יעסוק בשאלה של מי מדבר עם מי C-Plane - Communication Traffic, זה יוכל למצוא קבוצות עם מקורות או יעדים זהים כאשר אחד מהם הוא שרת שליטה ובצדו השני המחשב הנגוע. המישור השני יהיה אחראי על אופי וסוג התקשורת שמועברת A-Plane - Activity Traffic, מישור זה יתפוס תבניות זהות באופי התקשורת בין שני הצדדים. נאחד את המידע שמתקבל משני המישורים ונוכל למצוא רשתות בוטנט.



איור 32. ארכיטקטורות בוטנט של שרת שליטה מרכזי וארכיטקטורת P2P [4].

### גבולות המערכת

נגדיר את גבולות המערכת על ידי קביעת קריטריונים לבוטנט. בוטנט יוגדר על ידי שני קריטריונים: במישור התקשורתי ובמישור הפעילות הזדונית. המערכת לא תזהה פעילות חשודה שקוראת רק באחד המישורים. לדוגמא, תוכנת שיתוף קבצים של bittorrent יכולה להיות מועמד של מישור תקשורתי משום שיכולה להיות דומה לתקשורת של בוטנט בארכיטקטורת P2P. לחילופין תוכנות זדוניות אחרות יכולות לבצע פעילות זדונית ללא תבניות תקשורתיות. תוכנות זדוניות אלה לא יחשבו כבוטנט.

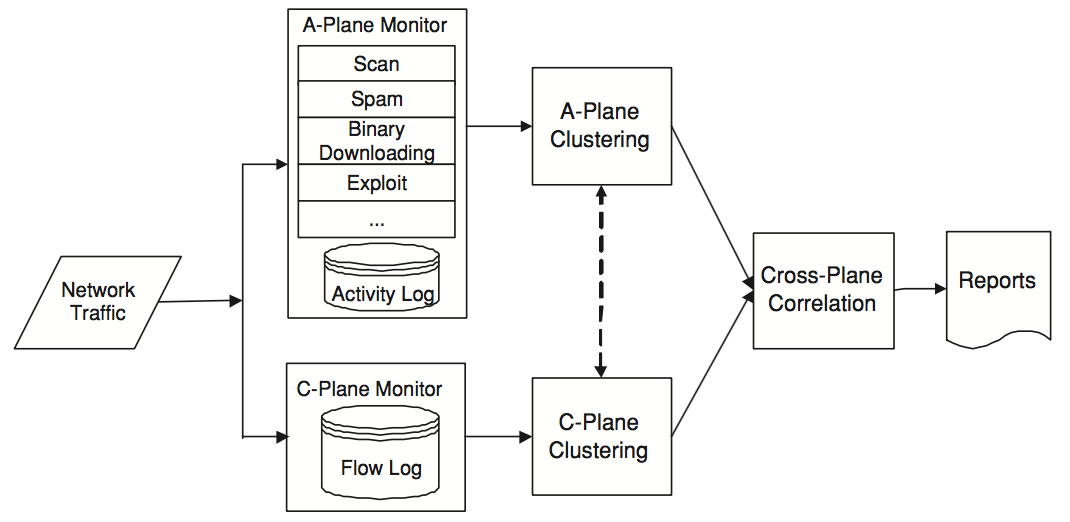
### מטרות המערכת

מטרת המערכת היא זיהוי רשת בוטנט בתוך הרשת המנוטרת על ידי המערכת. המערכת לא תמנע או תזהה הדבקה של מחשב. החיפוש יתבצע תוך כדי שמירה על התכונות הבאות:

1. מציאת הבוטנט לא תהיה תלויה בפרוטוקול, ארכיטקטורה, שינויים במיקומים של שרתי שליטה.
2. מציאת הבוטנט לא תהיה תלויה בתוכן ההודעות המועברות, הן יכולות להיות מוצפנות.
3. המערכת תשמור על מספר קטן של false positives.
4. המערכת תעבד את הנותנים בזמן סביר והזיהוי יהיה יעיל.

### ארכיטקטורת המערכת

הניטור מתבצע בנקודה בה הרשת הפנימית מתקשרת אל החיצונית. כל הנתונים הרלוונטיים יישמרו בקבצים שינותחו לאחר מכן. מבנה המערכת מתואר באיור 33. המידע של הרשת מנוטר בשני מישורים - מישור תקשורתי ומישור הפעילות הזדונית. A-Plane Monitor יאגור את המידע שקשור בפעילות הזדונית. C-Plane Monitor יאגור את המידע שקשור בתקשורת. לאחר מכן המידע האגור יעבור את שלב האשכול בכל אחד מהמישורים - A-Plane Clustering ו C-Plane clustering. לאחר מכן, יבוצע איחוד של תוצאות ב Cross-Plane Correlation וזאת במטרה להבין האם יש בוטנט ברשת.



איור 33. ארכיטקטורת המערכת BotMiner לזיהוי בוטנטים [4].

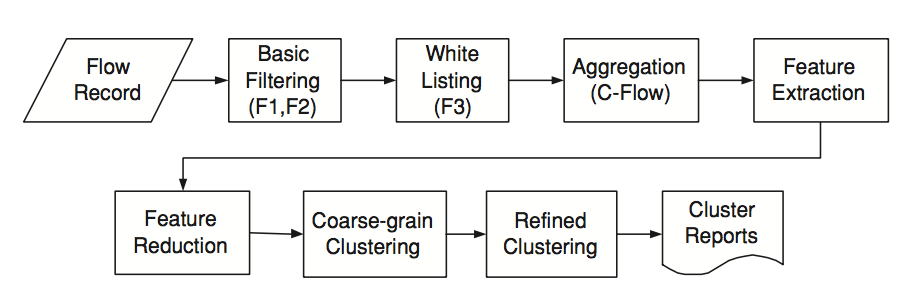
### אישכול במישור C

#### הכנת נתונים

המידע הנאגר במישור התקשורתי הוא ווקטור בעל מאפיינים הבאים:

1. זמן.
2. משך ההתקשרות.
3. כתובת IP של המקור.
4. כתובת IP של היעד.
5. פורט מקור.
6. פורט יעד.
7. מספר הבתים שהועברו.
8. מספר החבילות שהועברו.
9. פרוטוקול TCP/UDP.

תהליך האישכול מתואר באיור 34:



*איור 34. תהליך האישכול במישור התקשורתי* [4]*.*

תחילה נסנן את המידע הגולמי שקיבלנו מ C-Plane Monitor, בעזרת שני חוקי סינון:

1. נתעלם מכל המידע העובר ברשת הפנימית:
   1. מידע שמקור והיעד שלו הוא הרשת הפנימית.
   2. נתעלם מכל התקשרות המתחילה מהרשת החיצונית אל הרשת הפנימית.
2. נתעלם מהתקשרויות שהחלו באופן לא תקני (רוב ההתקשרויות הללו קשורות בסריקת רשת במישור A-Plane).

השלב של White Listing יסנן את ההתקשרויות שיעדם הוא ידוע ומוכר כדוגמת google.com, cnn.com. הסינון מתבסס על רשימת האתרים הפופולריים ביותר מ alexa.com.

בשלב הקיבוץ (Aggregation) מתבצע קיבוץ של התקשרויות בעלות אותו פרוטוקול (TCP/UDP), אותה כתובת מקור וכתובת יעד ואותו פורט שקרו בפרק זמן מסוים (בד"כ יום).

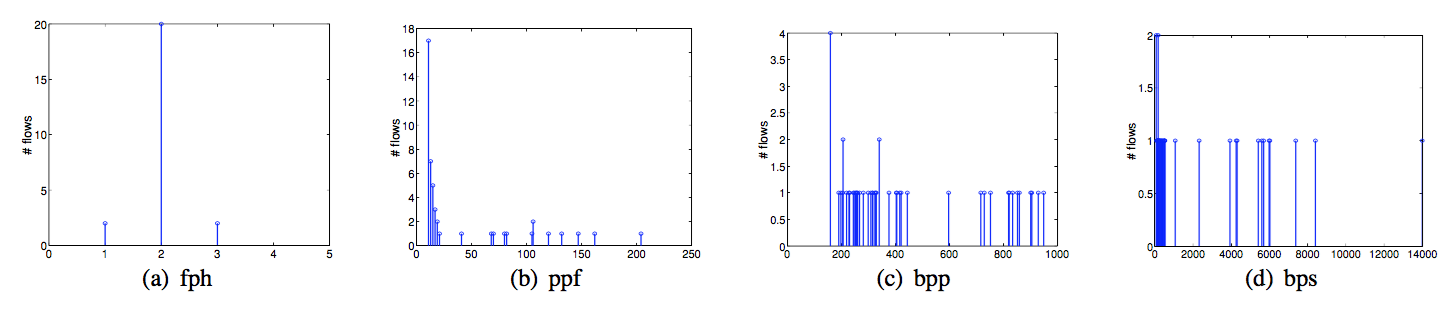
נסמן את ההתקשרויות המקובצות ב כאשר כל היא התקשרות יחידה. הסט מציג מי דיבר עם מי בפרק זמן (יום אחד).

#### ייצוג ווקטורי של התקשרויות

נגדיר ארבעה מאפיינים עבור ווקטור האשכול הקשור במישור התקשורתי. עבור כל נחלץ כמה מאפיינים סטטיסטיים ונתרגם אותם לווקטורים בעלי מימדים . נסמן את תהליך הפיכת לווקטור על ידי פונקציה . בהינתן התקשרות מקובצת נחשב התפלגות דיסטקרטית של ארבעה משתנים אקראיים:

1. מספר ההתקשרויות בשעה (fph) - מחושב על ידי ספירה של מספר ההתקשרויות TCP/IP ב בכל שעה ביום E.
2. מספר החבילות שהועברו בהתקשרות (ppf) - מחושב על ידי סכימה של סך החבילות שנשלחו בכל התקשרות TCP/IP ב .
3. מספר ממוצע של בתים בחבילה (bpp) - בכל התקשרות ב נחלק את סך כל מספר הבתים ב בסך כל מספר החבילות ב .
4. מספר הממוצע של בתים בשנייה (bps) - מחושב מסך כל מספר הבתים המועברים בכל ב מחולק באורך ההתקשרות .

דוגמא לתהליך שתיארנו ניתן לראות באיור 35.

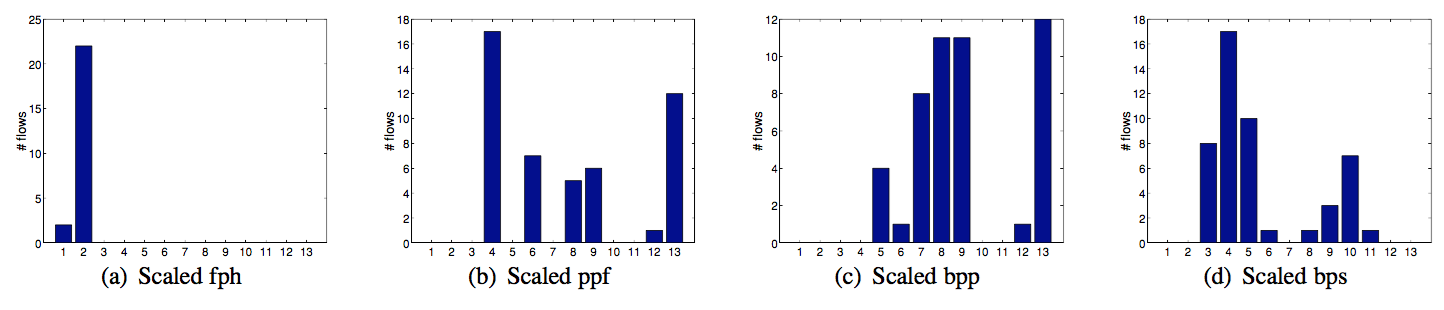


איור 35. התקשרויות ל google.com על פי ארבעת המשתנים האקראיים [4].

בעזרת שיטת ה Binning נוכל לקרב את ההתפלגויות הדיסקרטיות של ארבעת המשתנים שקיבלנו.

נחלק את ציר ה-x ל 13 מרווחים . נחשב את , , , , , , , , , , , עבור התפלגות דיסקרטית. נגדיר :

. סך הכול קיבלנו ווקטור של 52 איברים (13 ערכים עבור כל ארבעת המשתנים). את התוצאה נוכל לראות באיור 36.



איור 36. התקשרויות ל google.com על פי ארבעת המשתנים לאחר הפעלת שיטת ה Binning [4].

#### אישכול

אחרי שהגדרנו את הווקטורים עבור האישכול אפשר להתחיל בתהליך האישכול עצמו. מטרת האישכול במישור ה-C התקשורתי היא למצוא התקשרויות דומות זו לזו. גם במקרה בו הבוטנט משתמש בארכיטקטורה עם הרבה שרתי שליטה שונים ההתקשרות אליהם, הפרמטרים שמוצגים בווקטור, עדיין יסמנו לנו התקשרויות שדומות באופיין.

Gu&Perdisci [4] בחרו ב X-means כדי למצוא קבוצות התקשרויות דומות. הם בחרו ב X-means מכיוון שלעומת K-means אלגוריתם אישכול זה אינו דורש לספק K – מספר האשכולות, בנוסף X-means מראה ביצועים טובים עבור בסיסי נתונים גדולים.

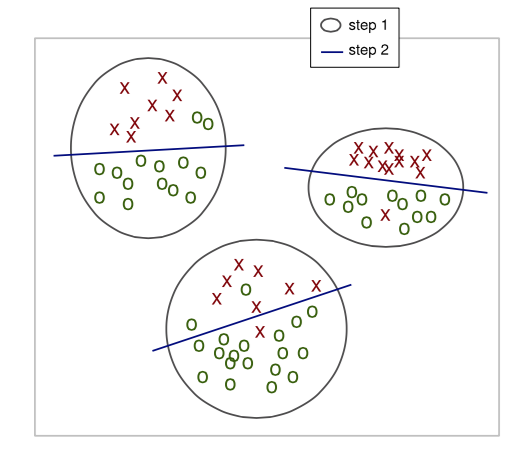
נבצע את האישכול על בסיס נתונים . הבעיה שבה אנחנו נתקלים היא הגודל של בסיס הנתונים. בגלל מספר רב של התקשרויות מספר הווקטורים מאוד גדול. בנוסף מספר הממדים בכל ווקטור גם כן מאוד גבוה.

בסיס נתונים גדול ומספר רב של ממדים מגדילים זמן ריצת X-means. לכן נבצע את האישכול בשני שלבים.

בשלב הראשון יש הרבה מאוד נתונים ולכן נצמצם את מרחב המאפיינים שלנו בווקטור מ-52 ל-8, על ידי חישוב של ממוצע ושונות של כל הפרמטרים הסטטיסטיים שחישבנו מקודם (fph, ppf, bpp,bps). נפעיל את האלגוריתם X-means. תוצאת האלגוריתם היא מספר אשכולות שמחלק את הנתונים ועוזר להתמודד עם הגודל של בסיס הנתונים.

בשלב השני נפעיל עוד הפעם את האלגוריתם על כל אשכול בנפרד, אבל הפעם נשתמש בכל 52 המאפיינים שחישבנו מקודם. נוכל לראות הדגמה באיור 37.

השימוש בשני שלבי אישכול יכול לגרום לרעש ושיבושים בתוצאות, אך הדבר אינו משמעותי באופן ניכר. דרך נוספת להימנע מהשיבושים ולחסוך בזמן האשכול היא להשתמש רק בתוצאות של לוגים אשר הגיעו ממישור A, כלומר לוגים שזוהו כפעילות חשודה. שיטה זו מהירה יותר אך גם ניתן לאבד בה מידע.



איור 37. שני שלבי האישכול. האישכול הראשון יוצר את האשכולות הגסים. האישכול השני מבוצע על כל אחד מן האשכולות הגסים [4].

### אישכול במישור A

#### הכנת נתונים

הנתונים במישור הפעילות הזדונית נאספים על ידי שימוש בתוכנות גילוי בפעילות זדונית ברשת. תוכנות עזר כמו Snort ו SCADE עוזרות בזיהוי וגילוי של פעילות זדונית ברשת. במישור A הפעילות הזדונית מתחלקת לכמה קטגוריות:

1. סריקת רשת לפרצות אבטחת מידע.
2. שליחת ספאם.
3. הורדות תוכנות זדוניות.
4. שימוש בפרצות אבטחת מידע.

#### אישכול

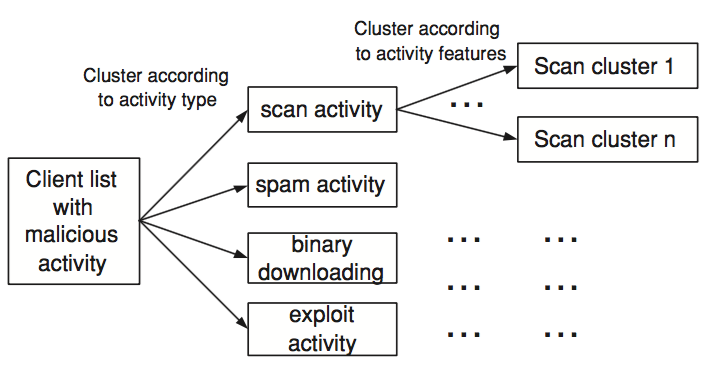
במישור A נפעל בדרך דומה כמו שהוצגה במישור C. נפעל בשני צעדים. בצעד הראשון נבצע אישכול הפעילות על פי סוג הפעילות (סריקות, הורדות תוכנות זדוניות, ספאם ועוד...). בשלב השני האישכול מתבצע על פי קריטריונים ספציפיים יותר למתקפה:

1. סריקה - טווח פורטים, טווח כתובות.
2. ספאם - יעדים חופפים של ספאם, תדירות שליחת מיילים.
3. הורדות תוכנות זדוניות - מקור זהה של הורדה (URL).
4. שימוש בפרצות אבטחת מידע - אותו מזהה של מתקפה (מתגלה על ידי Snort).

בפועל על פי התוצאות של הניטור בשטח, האשכול בוצע באופן הבא:

1. סריקה - אשכול לפי פורטים.
2. ספאם - אשכול בינארי של מחשבים שמבצעים פעילות ספאם וכאלה שלא.
3. הורדות קבצים - מורידים את החבילה הראשונה. האשכול מבוצע על פי החבילה הזו (החלק הראשוני של הקובץ).

במישור זה לא משתמשים באלגוריתם אישכול קלאסי כלשהו. עם זאת המערכת מניבה תוצאות טובות כפי שנראה בהמשך.



איור 38. אשכול במישור A [4].

### אישכול היררכי בין המישורים - Cross-plane Correlation

כאשר קיבלנו את התוצאות משני האישכולים - במישור A ומישור C, נבצע פעולת איחוד של הנתונים. נשתמש במידע שקיבלנו באישכולים הקודמים זה לצד זה על מנת לקבל ראיות חזקות יותר לפעילות בוטנט ברשת. נסמן ב כל מחשב שבו נצפתה פעילות חשודה אחת לפחות. נסנן את המחשבים שהציון שלהם נמוך מ . נמיין לקבוצות את שאר המחשבים עם פעילות חשודה על פי פונקציית דמיון שתיקח בחשבון את האישכולים של מישור A מישור C. נסמן את להיות סט של מחשבים שהתקבלו כפלט של אשכול במישור A. נסמן את המחשב . נגדיר את להיות סט של אשכולות אשר מכילים את המחשב . נגדיר את להיות סט של אשכולות אשר מכילים את המחשב . נוסחת הציון סבירות הבוטנט עבור מחשב בודד היא:

|  |  |  |
| --- | --- | --- |
|  |  | (6.9) |

– ציון סבירות לבוטנט עבור המחשב.

– משקל עבור פעילות במישור *A.*

– אשכול במישור *A* אשר מכיל את המחשב *h.*

– אשכול במישור *C* אשר מכיל את המחשב *h.*

כאשר ו . הוא סוג הפעילות הזדונית, לדוגמא סריקה או שליחת ספאם.

הוא המשקל של הפעילות הזדונית של . ככל שהפעילות חזקה או אינטנסיבית יותר המשקל יהיה גבוה יותר. ספאם או ניצול של פרצה נחשבים להיות פעילויות חזקות יותר מאשר סריקת רשת או הורדת קבצים.

בפועל נשתמש ב עבור כל ערכי ו . משמעות הדבר היא שמחשב שנמצא בשני האשכולות של מישור הפעילות הזדונית A, ו/או מישור התקשורתי C, יחשב להיות מועמד חשוד.

משמעות נוסחת הציון היא שמחשב יקבל ציון גבוה אם הוא היה מעורב בכמה פעילויות חשודות ומחשבים אחרים שנמצאו באשכול של מראים גם כמה פעילויות חשודות. אם נקח לדוגמא את שבצע פעילות סריקה ופעילות ספאם. נניח שהאשכול של הסריקה הוא והאשכול של הספאם הוא . המחשב יקבל ציון גבוה יותר ככל שיהיו יותר מחשבים זהים בחיתוך של שתי הקבוצות . המישור התקשורתי משפיע באופן דומה, ככל שיהיו יותר מחשבים חופפים בין אשכול C ואשכול A (כאשר שניהם מכילים את המחשב ) כך הציון יהיה יותר גבוה.

בהינתן ערך הסף , נחשיב את כל המחשבים כך ש , כרובוטים ברשת בוטנט. נסנן מחשבים שלא תואמים את ההגדרה הזו . נגדיר את כסט של רובוטים שזוהו. הוא סט של אשכולות במישור A כך שכל אשכול מכיל לפחות רובוט אחד . הוא סט של אשכולות במישור C כך שכל אשכול מכיל לפחות רובוט אחד . נגדיר את

להיות סט מסודר של אשכולותA ו C. נגדיר עבור כל מחשב

ווקטור בינארי כך ש . בעזרת ווקטור זה נוכל להגדיר פונקציית מרחק בין שני מחשבים:

|  |  |  |
| --- | --- | --- |
|  | כאשר , | (6.10) |

– פונקציית מרחק בין שני מחשבים.

מטרת הנוסחה היא לקבץ שני מחשבים באשכול אחד אם הם חולקים אשכול פעילות זדונית ואשכול תקשורתי אחד לפחות. בעזרת פונקציית מרחק שהגדרנו נוכל להפעיל אלגוריתם אישכול היררכי על הנתונים. נשתמש ב DB index עבור בדיקת טיב האשכול [16]. באיור 39 נוכל לראות דוגמא של דנדרוגרמה עבור אשכול היררכי, כאשר הם מחשבים וחיתוך הדנדרוגרמה נעשה ברמה 90 – לכן קיבלנו שני בוטנטים במערכת שמכילים את המחשבים הבאים: .

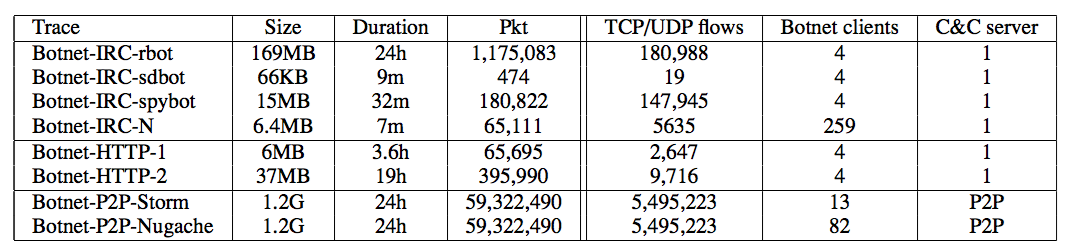
איור 39. דוגמא של אישכול היררכי בתהליך זיהוי בוטנט [4].

### תוצאות ניסוייות

נתונים

1. אוגרי הנתונים הוצבו באוניברסיטת Georgia Tech למשך 10 ימים.

2. עבור הניסוי נאספו נתונים של 8 בוטנטים. ראה טבלה 3 לסטטיסטיקות:



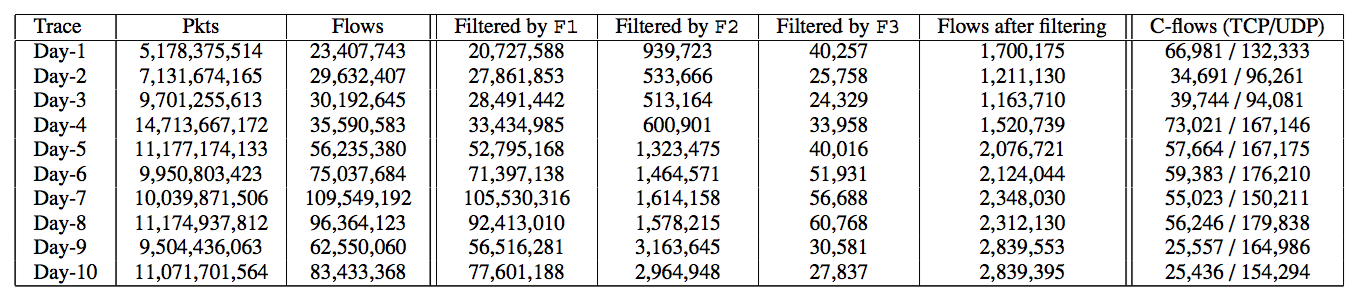
טבלה 46.. סטטיסטיקות של בוטנטים ידועים [4].

ניסוי

מניחים שהקבוצה הראשונה של הנתונים נקייה מבוטנטים. היא כוללת תעבורה מגוונת המבוססת על פרוטוקולים כמו HTTP, SMTP, POP, FTP, SSH, NetBios, DNS, SNMP, IM, P2P ו-IRC. בוטנטים שנאספו נבדקו בסביבות וירטואליות (V-Spybot V-Sdbot B-HTTP-I, B-HTTP-II, Rbot) כמו כן הגיע לידי החוקרים Trace של תקשורת Botnet-IRC-N (Phatbot) , Nugache ו Storm.

תוצאות

מהלך הרצת ה BotMiner על הנתונים האגורים מתואר בטבלה 4. במהלך 10 ימים נאספו ביליארדי חבילות ומיליוני התקשרויות. המסנן F1 הוכיח את עצמו כיעיל מאוד בסינון בסט גדול מאוד של נתונים – סוננו כ 90% מהנתונים. המסנן F2 סינן בין 0.5-3 מיליון התקשרויות כל יום. מסנן F3 סינן עשרות אלפי התקשרויות כל יום.

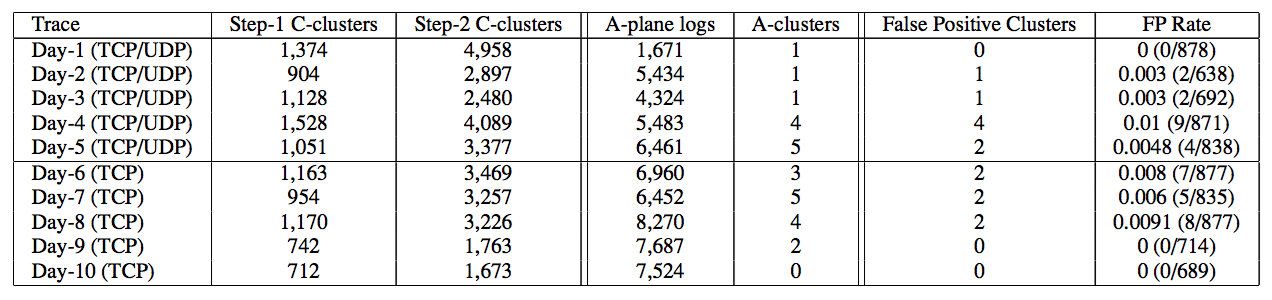


טבלה 56.. סטטיסטיקות במישור התקשרותי וסינונים של F1, F2, F3 [4].

אחרי הפילטור נותרו כ 1-3 מיליוני התקשרויות. ההתקשרויות שנותרו קובצו להתקשרויות מקובצות (C-flows) כפי שהוסבר בפרק ‏6.4.5.2. כך התקבלו כ – 170000 התקשרויות מקובצות ביום.

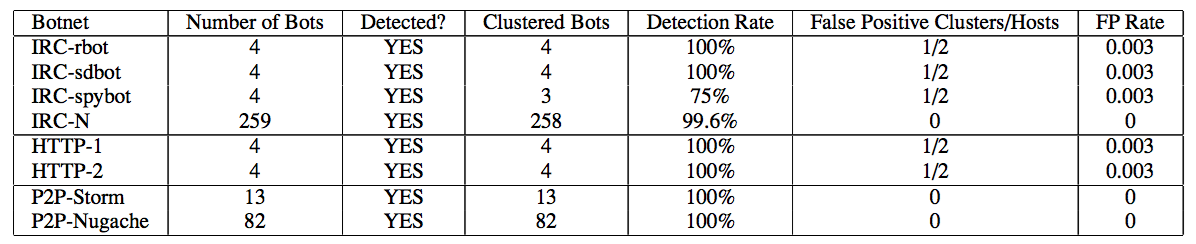
BotMiner רץ על נתונים שנאספו מהרשת. נקודת המוצא שלנו היא שהרשת לא מכילה בוטנטים זדוניים. הדבר נבדק בעזרת מערכות לאיתור בוטנטים קיימות – BotSniffer, BotHunter. הנחה זו מאפשרת לבדוק את כמות ה false positives במערכת BotMiner. כפי שניתן לראות, נוצרו אלפי לוגים במישור A. המערכת מעוניינת למצוא קבוצות של מכונות שמבצעות פעילות דומה. לכן סוננו אשכולות במישור A שהכילו רק הוסט אחד כדי להקטין כמות ה false positives. רוב האשכולות שנמצאו וסומנו כ false positives מכילים כ 2 מחשבים. ניתוח האשכולות הראה כי מרבית ה false positives נגרמו על ידי הורדת קבצים בינאריים מאתרים שלא נכללים בTOP100 של alexa.com.

טבלה 5 מציגה את הנתונים של האישכולים בשני המישורים.



טבלה 66.. תוצאות אישכול במישור A ו C [4].

על מנת לבדוק את יעילות מערכת BotMiner נשלב את נתוני ההתקשרות של 8 הבוטנטים הקיימים. בהינתן בוטים הקיימים בנתוני ההתקשרות שאספנו קודם, נבחר באופן אקראי כתובות IP ונחליף אותם בכתובות IP של בוטים. הפעולה מאפשרת לנו לשתול בוטנט בתוך מערכת מבלי לפגוע בה. בטבלה 66. נראה את תוצאות של הרצת BotMiner על הנתונים השתולים. אפשר לראות שיש אחוז מאוד גבוה של זיהוי בוטנטים. 6 בוטנים זוהו באופן מלא – כל בוטנט התקבל בתוך אשכול. IRC-N, IRC-Spybot הם בוטנטים בהם BotMiner לא הצליח לזהות רק מחשב פגוע אחד. BotMiner יצר אשכול false positive שכלל שני מחשבים שנמצאו גם בבדיקה הראשונה כ false positives ללא המידע השתול.



טבלה 76.. גילוי בוטנט על ידי BotMiner [4].

### מסקנות

מערכת BotMiner הראתה תוצאות טובות מאוד במציאת בוטנטים. כל הבוטנטים שהושתלו בניסוי נמצאו ומספר הfalse positives הוא קטן מאוד. זו מערכת שלא תלויה בפרוטוקול או ארכיטקטורה כפי שראינו במגוון הבוטנטים שנתפסו. המערכת משתמשת בהגדרות הבסיסיות של בוטנט ומאפייניו על מנת לזהות קבוצות של בוטים. הגדרה זו היא השימוש בתבניות תקשורת והתנהגות דומה עבור כל המחשבים – בוטים בבוטנט.

# סיכום

עבודה זו עסקה בפתרון בעיית זיהוי בוטנט בעזרת אלגוריתמי אשכול. הבוטנט הוא אחת הסכנות שהביאה את הקדמה הטכנולוגית במחשבים ובפרט ברשת האינטרנט. בוטנטים פוגעים באבטחה באינטרנט מבלי שרוב האנשים בכלל מודעים לבעיה. עם זאת אנחנו תלויים יותר ויותר במחשבים והרשת. כפי שניתן לראות בעבודה יש דרכים להתמודד עם הבוטנטים. הוצגו כאן 3 שיטות בזיהוי ומציאת הבוטנטים בעזרת אלגוריתמי אשכול. כל שיטה באה לפתור בעיה שונה בוואריאציה שונה. מהסיבה שהבוטנט הוא לא סטאטי, ברגע קריאת המאמר חלק מהשיטות עלולות להיות כבר לא רלוונטיות שכן הגרסאות החדשות של בוטנטים אולי כבר יודעות להתחמק מהמערכות שמזהות אותם. לכן זו מלחמה מתמשכת - השיטות הופכות להיות יותר מתוחכמות ומצד שני הבוטנטים ממשיכים להתפתח.

המכנה המשותף של שלושת השיטות הוא אלגוריתמי אישכול עליהם הן מבוססות. אלגוריתמי אישכול הם אלגוריתמים אשר באים לפתור סוג מסוים של בעיות מעולם של כריית מידע. הוצגו שלשה אלגוריתמי אישכול ובוצעה השוואה בין האלגוריתמים. בעבודה זו הוצגו האלגוריתם K-Means וממשיכו ה X-Means. לעומתם הוצג גם האלגוריתם האשכול ההיררכי שניגש לבעיה מזווית שונה ומפיק תוצאות שונות. בכל אחת מהשיטות השתמשו באלגוריתם שונה, כיוון שפתרון של בעיות שונות דורש כלים שונים לקבלת התוצאה הטובה ביותר.

# המשך מחקר

אחת הבעיות הגדולות שנמצאת בזיהוי בוטנט בעזרת אלגוריתמי אישכול היא התקפת רעש. בוטנטים מתפתחים והופכים לחכמים יותר. כאשר תוצאות מחקרים ומערכות לניטור בוטנטים הופכים להיות פומביים אפשר להניח שיוצרי הבוטנט כבר יגיבו בהתאם. התקפת רעש היא הפעולה שבוטנט עושה כדי לבלבל את המערכת או האלגוריתם שמנסה לזהות אותו. אם מערכת מסוימת מחפשת אחר התנהגות או אופי תקשורת מסוים ומבצעת אשכול על פי תכונות אלו הבוטנט, בגרסתו החדשה, יתחיל להתנהג בצורה לא צפויה וישנה לחלוטין את אופי התקשורת על מנת לא להימצא בקלות. בעקבות התקפות הרעש דיוק של מערכות מזהות בוטנט יכול להיפגע מאוד והן יפסיקו להיות אפקטיביות על גרסאות חדישות של נוזקות. יהיה צורך לבצע מחקר חדש ולמצוא את הדרך האופטימלית לטיפול בהתקפת רעש.

כמה שיטות למציאת בוטנט מסתמכות על השימוש בפרוטוקול HTTP הלא מוצפן. שיטות אלו עושות שימוש במאפיינים שנראים רק כאשר הבקשה לא מוצפנת. מאפיינים אלו הם חלק אינטגרלי באלגוריתם אשכול. בוטנטים יכולים להשתמש בפרוטוקול מוצפן HTTPS כדי לבצע התקשרות. שינוי זה יכול למוטט לחלוטין את השיטה לזיהוי הבוטנט. יש לציין שלא כל הבוטנטים יעדיפו להצפין את התקשורת באופן שרירותי בגלל מגבלות חדשות שהדבר יציב לבוטנטים עצמם – בחלק מהמערכות יש וידוי היעדים אם מדובר בפרוטוקול HTTPS.

# מקורות

|  |  |
| --- | --- |
| [1] | C. Hoffman, "What Is a Botnet?," 3 6 2014. [Online]. Available: http://www.howtogeek.com/183812/htg-explains-what-is-a-botnet/. |
| [2] | M. Antonakakis, R. Perdisci, Y. Nadji, N. Vasiloglou II, S. Abu-Nimeh, W. Lee and D. Dagon, "From Throw-Away Traffic to Bots: Detecting the Rise of DGA-Based Malware.," in *USENIX Security Symposium*, 2012. |
| [3] | M. Thomas and A. Mohaisen, "Kindred domains: detecting and clustering botnet domains using DNS traffic," in *Proceedings of the companion publication of the 23rd international conference on World wide web companion*, 2014. |
| [4] | G. Gu, R. Perdisci, J. Zhang and W. Lee, "BotMiner: Clustering Analysis of Network Traffic for Protocol-and Structure-Independent Botnet Detection," in *USENIX Security Symposium*, 2008. |
| [5] | R. Perdisci, W. Lee and N. Feamster, "Behavioral Clustering of HTTP-Based Malware and Signature Generation Using Malicious Network Traces," in *NSDI*, 2010. |
| [6] | A. Zand, G. Vigna, X. Yan and C. Kruegel, "Extracting Probable Command and Control Signatures for Detecting Botnets," 2014. |
| [7] | J. Han and M. Kamber, Data Mining Concepts and Techniques, Elsevier Inc., 2006. |
| [8] | D. Pelleg and A. W. Moore, "X-means: Extending K-means with Efficient Estimation of the Number of Clusters," in *ICML*, 2000. |
| [9] | K. Deng and A. Moore, "Multiresolution instance-based learning," in *Proceedings of IJCAI-95*, Montreal, Quebec, Canada, 1995. |
| [10] | D. Pelleg and A. Moore, "Accelerating exact k-means with geometric reasoning," Carnegie Mellon University, Pittsburgh, PA, 2000. |
| [11] | D. Müllner, "Modern hierarchical, agglomerative clustering algorithms," http://arxiv.org/, 2011. |
| [12] | H. Trevor, R. Tibshirani and J. Friedman, The Elements of Statistical Learning, Springer, 2008. |
| [13] | M. Steinbach, G. Karypis and V. Kumar, "A Comparison of Document Clustering Techniques," 2000. |
| [14] | M. Kaur and U. Kaur, "Comparison Between K-Means and Hierarchical Algorithm Using Query Redirection," *International Journal of Advanced Research in Computer Science and Software Engineering,* 2013. |
| [15] | T.-F. Yen, A. Oprea, K. Onarlioglu, T. Leetham, W. Robertson, A. Juels and E. Kirda, "Beehive: large-scale log analysis for detecting suspicious activity in enterprise networks," in *Proceedings of the 29th Annual Computer Security Applications Conference*, 2013. |
| [16] | M. Halkidi and Y. Batistakis, "On clustering validation techniques," 2001. |
| [17] | J. Newsome, B. Karp and D. Song, "Polygraph: Automatically generating signatures for polymorphic worms," *IEEE Symposium on Security and Privacy,* 2005. |
| [18] | S. Wu and U. Manber, "Agrep - a fast approximate pattern-matching tool.," vol. USENIX Technical Conference, 1992. |
| [19] | U. Bayer, P. Milani Compareti, C. Hlauschek, C. Kruegel and E. Kirda, "Scalable, behavior-based malware clustering," in *Network and Distributed System Security Symposium*, 2009. |
| [20] | J. Newsome, B. Karp and D. Song, "Polygraph: Automatically generating signatures for polymorphic worms," in *IEEE Symposium on Security and Privacy*, 2005. |

# Abstract

This paper focuses on a constantly growing and evolving web threat – botnet, and its detection using clustering algorithms. Botnet is a computer network controlled by a server or group of servers that issue commands to bot members - computers infected by malware. Usually the malware is invisible and it forces the computer to send out spam and perform DDOS attacks.

3 methods of botnet detection were presented in this paper:

* Behavioral clustering and signature generation using malicious network traces.
* Beehive system. Beehive performs large-scale analysis for detection of suspicious activity in enterprise networks.
* BotMiner system. BotMiner provides protocol- and structure-independent botnet detection.

Each one of the three methods discussed in this paper use different clustering algorithm. The choice of an algorithm is based on the character of the data analyzed by the system: database size, type of objects in the database etc. The paper presents 3 clustering algorithms: K-Means, X-Means and Hierarchical clustering, and provides comparison (by scalability, complexity, cluster validity, etc) between these algorithms.

This work proves following:

1. Clustering algorithms that were presented in this paper turned to be very useful for botnet detection.

2. Botnets change rapidly and continuously adapt themselves to new detection tools. Thus, the detection tools should change and improve frequently.

Contents

[1 Abstract 1](file:///C:/Users/Tamir/Downloads/MSc%20Final%20Assignment/25.12.2017/תוכן%20עניינים%20באנגלית.docx#_Toc481847201)

[2 Structure 1](file:///C:/Users/Tamir/Downloads/MSc%20Final%20Assignment/25.12.2017/תוכן%20עניינים%20באנגלית.docx#_Toc481847202)

[3 Botnet 1](file:///C:/Users/Tamir/Downloads/MSc%20Final%20Assignment/25.12.2017/תוכן%20עניינים%20באנגלית.docx#_Toc481847203)

[3.1 What is Botnet 1](file:///C:/Users/Tamir/Downloads/MSc%20Final%20Assignment/25.12.2017/תוכן%20עניינים%20באנגלית.docx#_Toc481847204)

[3.2 Botnet Distribution 2](file:///C:/Users/Tamir/Downloads/MSc%20Final%20Assignment/25.12.2017/תוכן%20עניינים%20באנגלית.docx#_Toc481847205)

[3.3 Botnet Types 2](file:///C:/Users/Tamir/Downloads/MSc%20Final%20Assignment/25.12.2017/תוכן%20עניינים%20באנגלית.docx#_Toc481847206)

[3.4 Botnet Architecture 4](file:///C:/Users/Tamir/Downloads/MSc%20Final%20Assignment/25.12.2017/תוכן%20עניינים%20באנגלית.docx#_Toc481847207)

[3.5 Botnet Discovery 4](file:///C:/Users/Tamir/Downloads/MSc%20Final%20Assignment/25.12.2017/תוכן%20עניינים%20באנגלית.docx#_Toc481847208)

[4 Clustering Algorithms 5](file:///C:/Users/Tamir/Downloads/MSc%20Final%20Assignment/25.12.2017/תוכן%20עניינים%20באנגלית.docx#_Toc481847209)

[4.1 Introduction 5](file:///C:/Users/Tamir/Downloads/MSc%20Final%20Assignment/25.12.2017/תוכן%20עניינים%20באנגלית.docx#_Toc481847210)

[4.2 K-means Algoright 8](file:///C:/Users/Tamir/Downloads/MSc%20Final%20Assignment/25.12.2017/תוכן%20עניינים%20באנגלית.docx#_Toc481847211)

[4.2.1 Introduction 8](file:///C:/Users/Tamir/Downloads/MSc%20Final%20Assignment/25.12.2017/תוכן%20עניינים%20באנגלית.docx#_Toc481847212)

[4.2.2 Goals 8](file:///C:/Users/Tamir/Downloads/MSc%20Final%20Assignment/25.12.2017/תוכן%20עניינים%20באנגלית.docx#_Toc481847213)

[4.2.3 Algorithm Description 8](file:///C:/Users/Tamir/Downloads/MSc%20Final%20Assignment/25.12.2017/תוכן%20עניינים%20באנגלית.docx#_Toc481847214)

[4.2.4 Algorithm Pros 9](file:///C:/Users/Tamir/Downloads/MSc%20Final%20Assignment/25.12.2017/תוכן%20עניינים%20באנגלית.docx#_Toc481847215)

[4.2.5 Algorithm Cons 9](file:///C:/Users/Tamir/Downloads/MSc%20Final%20Assignment/25.12.2017/תוכן%20עניינים%20באנגלית.docx#_Toc481847216)

[4.2.6 Algorithm Analysis 9](file:///C:/Users/Tamir/Downloads/MSc%20Final%20Assignment/25.12.2017/תוכן%20עניינים%20באנגלית.docx#_Toc481847217)

[4.3 X-means Algorithm 10](file:///C:/Users/Tamir/Downloads/MSc%20Final%20Assignment/25.12.2017/תוכן%20עניינים%20באנגלית.docx#_Toc481847218)

[4.3.1 Introduction 10](file:///C:/Users/Tamir/Downloads/MSc%20Final%20Assignment/25.12.2017/תוכן%20עניינים%20באנגלית.docx#_Toc481847219)

[4.3.2 Goals 10](file:///C:/Users/Tamir/Downloads/MSc%20Final%20Assignment/25.12.2017/תוכן%20עניינים%20באנגלית.docx#_Toc481847220)

[4.3.3 Algorithm Description 10](file:///C:/Users/Tamir/Downloads/MSc%20Final%20Assignment/25.12.2017/תוכן%20עניינים%20באנגלית.docx#_Toc481847221)

[4.3.4 K-means Acceleration 13](file:///C:/Users/Tamir/Downloads/MSc%20Final%20Assignment/25.12.2017/תוכן%20עניינים%20באנגלית.docx#_Toc481847222)

[4.3.5 Blacklisting Algorithm 16](file:///C:/Users/Tamir/Downloads/MSc%20Final%20Assignment/25.12.2017/תוכן%20עניינים%20באנגלית.docx#_Toc481847223)

[4.3.6 Accelerating Improve Structure 16](file:///C:/Users/Tamir/Downloads/MSc%20Final%20Assignment/25.12.2017/תוכן%20עניינים%20באנגלית.docx#_Toc481847224)

[4.3.7 Other Accelerations 16](file:///C:/Users/Tamir/Downloads/MSc%20Final%20Assignment/25.12.2017/תוכן%20עניינים%20באנגלית.docx#_Toc481847225)

[4.3.8 Algorithm Pros 17](file:///C:/Users/Tamir/Downloads/MSc%20Final%20Assignment/25.12.2017/תוכן%20עניינים%20באנגלית.docx#_Toc481847226)

[4.3.9 Algorithm Cons 17](file:///C:/Users/Tamir/Downloads/MSc%20Final%20Assignment/25.12.2017/תוכן%20עניינים%20באנגלית.docx#_Toc481847227)

[4.3.10 Algorithm Analysis 17](file:///C:/Users/Tamir/Downloads/MSc%20Final%20Assignment/25.12.2017/תוכן%20עניינים%20באנגלית.docx#_Toc481847228)

[4.4 Hierarchical Clustering Algorithm 18](file:///C:/Users/Tamir/Downloads/MSc%20Final%20Assignment/25.12.2017/תוכן%20עניינים%20באנגלית.docx#_Toc481847229)

[4.4.1 Introduction 18](file:///C:/Users/Tamir/Downloads/MSc%20Final%20Assignment/25.12.2017/תוכן%20עניינים%20באנגלית.docx#_Toc481847230)

[4.4.2 Goals 19](file:///C:/Users/Tamir/Downloads/MSc%20Final%20Assignment/25.12.2017/תוכן%20עניינים%20באנגלית.docx#_Toc481847231)

[4.4.3 Algorithm Description 19](file:///C:/Users/Tamir/Downloads/MSc%20Final%20Assignment/25.12.2017/תוכן%20עניינים%20באנגלית.docx#_Toc481847232)

[4.4.4 Algorithm Pros 24](file:///C:/Users/Tamir/Downloads/MSc%20Final%20Assignment/25.12.2017/תוכן%20עניינים%20באנגלית.docx#_Toc481847233)

[4.4.5 Algorithm Cons 24](file:///C:/Users/Tamir/Downloads/MSc%20Final%20Assignment/25.12.2017/תוכן%20עניינים%20באנגלית.docx#_Toc481847234)

[4.4.6 Algorithm Analysis 24](file:///C:/Users/Tamir/Downloads/MSc%20Final%20Assignment/25.12.2017/תוכן%20עניינים%20באנגלית.docx#_Toc481847235)

[5 Comparative Analysis Between Clustering Algorithms 24](file:///C:/Users/Tamir/Downloads/MSc%20Final%20Assignment/25.12.2017/תוכן%20עניינים%20באנגלית.docx#_Toc481847236)

[5.1 Introduction 24](file:///C:/Users/Tamir/Downloads/MSc%20Final%20Assignment/25.12.2017/תוכן%20עניינים%20באנגלית.docx#_Toc481847237)

[5.2 Defining Comparison Criteria 25](file:///C:/Users/Tamir/Downloads/MSc%20Final%20Assignment/25.12.2017/תוכן%20עניינים%20באנגלית.docx#_Toc481847238)

[5.2.1 Clustering Quality Criteria 25](file:///C:/Users/Tamir/Downloads/MSc%20Final%20Assignment/25.12.2017/תוכן%20עניינים%20באנגלית.docx#_Toc481847239)

[5.2.2 Algorithm Criteria 28](file:///C:/Users/Tamir/Downloads/MSc%20Final%20Assignment/25.12.2017/תוכן%20עניינים%20באנגלית.docx#_Toc481847240)

[5.3 Results 28](file:///C:/Users/Tamir/Downloads/MSc%20Final%20Assignment/25.12.2017/תוכן%20עניינים%20באנגלית.docx#_Toc481847241)

[5.3.1 K-means vs X-means 28](file:///C:/Users/Tamir/Downloads/MSc%20Final%20Assignment/25.12.2017/תוכן%20עניינים%20באנגלית.docx#_Toc481847242)

[5.3.2 K-means vs Hierarchical Agglomerative Clustering 32](file:///C:/Users/Tamir/Downloads/MSc%20Final%20Assignment/25.12.2017/תוכן%20עניינים%20באנגלית.docx#_Toc481847243)

[5.4 Summary 34](file:///C:/Users/Tamir/Downloads/MSc%20Final%20Assignment/25.12.2017/תוכן%20עניינים%20באנגלית.docx#_Toc481847244)

[6 Using Clustering Algorithms in Botnet Discovery 36](file:///C:/Users/Tamir/Downloads/MSc%20Final%20Assignment/25.12.2017/תוכן%20עניינים%20באנגלית.docx#_Toc481847245)

[6.1 Introduction 36](file:///C:/Users/Tamir/Downloads/MSc%20Final%20Assignment/25.12.2017/תוכן%20עניינים%20באנגלית.docx#_Toc481847246)

[6.2 Behavioral Clustering of HTTP-Based Botnet and Signature Generation 37](file:///C:/Users/Tamir/Downloads/MSc%20Final%20Assignment/25.12.2017/תוכן%20עניינים%20באנגלית.docx#_Toc481847247)

[6.2.1 Introduction 37](file:///C:/Users/Tamir/Downloads/MSc%20Final%20Assignment/25.12.2017/תוכן%20עניינים%20באנגלית.docx#_Toc481847248)

[6.2.2 System Goals 37](file:///C:/Users/Tamir/Downloads/MSc%20Final%20Assignment/25.12.2017/תוכן%20עניינים%20באנגלית.docx#_Toc481847249)

[6.2.3 System Description 37](file:///C:/Users/Tamir/Downloads/MSc%20Final%20Assignment/25.12.2017/תוכן%20עניינים%20באנגלית.docx#_Toc481847250)

[6.2.4 System Architecture 37](file:///C:/Users/Tamir/Downloads/MSc%20Final%20Assignment/25.12.2017/תוכן%20עניינים%20באנגלית.docx#_Toc481847251)

[6.2.5 Results 44](file:///C:/Users/Tamir/Downloads/MSc%20Final%20Assignment/25.12.2017/תוכן%20עניינים%20באנגלית.docx#_Toc481847252)

[6.3 Botnet Discovery in Enterprise Network using Log Analysis 49](file:///C:/Users/Tamir/Downloads/MSc%20Final%20Assignment/25.12.2017/תוכן%20עניינים%20באנגלית.docx#_Toc481847253)

[6.3.1 Introduction 49](file:///C:/Users/Tamir/Downloads/MSc%20Final%20Assignment/25.12.2017/תוכן%20עניינים%20באנגלית.docx#_Toc481847254)

[6.3.2 System Goals 50](file:///C:/Users/Tamir/Downloads/MSc%20Final%20Assignment/25.12.2017/תוכן%20עניינים%20באנגלית.docx#_Toc481847255)

[6.3.3 System Architecture 50](file:///C:/Users/Tamir/Downloads/MSc%20Final%20Assignment/25.12.2017/תוכן%20עניינים%20באנגלית.docx#_Toc481847256)

[6.3.4 Raw Data 50](file:///C:/Users/Tamir/Downloads/MSc%20Final%20Assignment/25.12.2017/תוכן%20עניינים%20באנגלית.docx#_Toc481847257)

[6.3.5 Normalization 50](file:///C:/Users/Tamir/Downloads/MSc%20Final%20Assignment/25.12.2017/תוכן%20עניינים%20באנגלית.docx#_Toc481847258)

[6.3.6 Features 51](file:///C:/Users/Tamir/Downloads/MSc%20Final%20Assignment/25.12.2017/תוכן%20עניינים%20באנגלית.docx#_Toc481847259)

[6.3.7 Clustering 53](file:///C:/Users/Tamir/Downloads/MSc%20Final%20Assignment/25.12.2017/תוכן%20עניינים%20באנגלית.docx#_Toc481847260)

[6.3.8 Results 54](file:///C:/Users/Tamir/Downloads/MSc%20Final%20Assignment/25.12.2017/תוכן%20עניינים%20באנגלית.docx#_Toc481847261)

[6.3.9 Conclusions 55](file:///C:/Users/Tamir/Downloads/MSc%20Final%20Assignment/25.12.2017/תוכן%20עניינים%20באנגלית.docx#_Toc481847262)

[6.4 BotMiner System 55](file:///C:/Users/Tamir/Downloads/MSc%20Final%20Assignment/25.12.2017/תוכן%20עניינים%20באנגלית.docx#_Toc481847263)

[6.4.1 Introduction 55](file:///C:/Users/Tamir/Downloads/MSc%20Final%20Assignment/25.12.2017/תוכן%20עניינים%20באנגלית.docx#_Toc481847264)

[6.4.2 System Limits 55](file:///C:/Users/Tamir/Downloads/MSc%20Final%20Assignment/25.12.2017/תוכן%20עניינים%20באנגלית.docx#_Toc481847265)

[6.4.3 System Goals 55](file:///C:/Users/Tamir/Downloads/MSc%20Final%20Assignment/25.12.2017/תוכן%20עניינים%20באנגלית.docx#_Toc481847266)

[6.4.4 System Architecture 56](file:///C:/Users/Tamir/Downloads/MSc%20Final%20Assignment/25.12.2017/תוכן%20עניינים%20באנגלית.docx#_Toc481847267)

[6.4.5 C Plane Clustering 56](file:///C:/Users/Tamir/Downloads/MSc%20Final%20Assignment/25.12.2017/תוכן%20עניינים%20באנגלית.docx#_Toc481847268)

[6.4.6 A Plane Clustering 59](file:///C:/Users/Tamir/Downloads/MSc%20Final%20Assignment/25.12.2017/תוכן%20עניינים%20באנגלית.docx#_Toc481847269)

[6.4.7 Hierarchical Cross-plane Correlation 60](file:///C:/Users/Tamir/Downloads/MSc%20Final%20Assignment/25.12.2017/תוכן%20עניינים%20באנגלית.docx#_Toc481847270)

[6.4.8 Results 61](file:///C:/Users/Tamir/Downloads/MSc%20Final%20Assignment/25.12.2017/תוכן%20עניינים%20באנגלית.docx#_Toc481847271)

[6.4.9 Conclusions 63](file:///C:/Users/Tamir/Downloads/MSc%20Final%20Assignment/25.12.2017/תוכן%20עניינים%20באנגלית.docx#_Toc481847272)

[7 Summary 63](file:///C:/Users/Tamir/Downloads/MSc%20Final%20Assignment/25.12.2017/תוכן%20עניינים%20באנגלית.docx#_Toc481847273)

[8 Future Work 64](file:///C:/Users/Tamir/Downloads/MSc%20Final%20Assignment/25.12.2017/תוכן%20עניינים%20באנגלית.docx#_Toc481847274)

[9 References 65](file:///C:/Users/Tamir/Downloads/MSc%20Final%20Assignment/25.12.2017/תוכן%20עניינים%20באנגלית.docx#_Toc481847275)

**The Open University of Israel**

**Department of Mathematics and Computer Science**

**Botnet Discovery with**

**Clustering Algorithms**

Final Paper submitted as partial fulfillment of the requirements

towards an M.Sc. degree in Computer Science

The Open University of Israel

Computer Science Division

By

**Tamir Aminov**

Prepared under the supervision of **Dr. Maya Herman**

December 2017

1. DGA (Domain Generation Algorithm) – אלגוריתם שמחולל שמות דומיינים ששרתי שליטה ישתמשו בחלקם. [↑](#footnote-ref-2)
2. Data Warehouse – מחסן נתונים. זהו בסיס נתונים המשמש מערכות תומכות החלטה. [↑](#footnote-ref-3)
3. לוג-נראות – הלוגריתם הטבעי של פונקציית הנראות. [↑](#footnote-ref-4)
4. Spherical Gaussian - משטח גאוסיאני כדורי. [↑](#footnote-ref-5)
5. ב[מדעי המחשב](https://he.wikipedia.org/wiki/%D7%9E%D7%93%D7%A2%D7%99_%D7%94%D7%9E%D7%97%D7%A9%D7%91), עץ kd הוא [עץ חיפוש](https://he.wikipedia.org/wiki/%D7%A2%D7%A5_%D7%97%D7%99%D7%A4%D7%95%D7%A9) המיועד לאחסון יעיל של נקודות במרחב k ממדי (ומכאן נגזר שמו). [↑](#footnote-ref-6)
6. מרחק לוינשטיין מנורמל – מרחק בין שתי מחרוזות ו ששווה למספר מינימלי של פעולות על תווים (הכנסה, מחיקה, החלפה) שדרושות כדי להפוך מחרוזת אחת לשנייה, מחולק ב . [↑](#footnote-ref-7)
7. מרחק Jaccard - מרחק בין שתי קבוצות A ו-B שמוגדר בצורה הבאה: . [↑](#footnote-ref-8)