# פרויקט סיכום בקורס מבוא לאופטימיזציה – פתרון בעיית שיבוץ מערכת שעות

אלן ברונשטיין 206228751, נעה עבו 208523514

המחלקה למדעי המחשב, אוניברסיטת בר אילן

https://github.com/Noabbo/Optimization_course_scheduling

## הקדמה

סטודנטים רבים מדי שנה נתקלים בצורך לשבץ את הקורסים שהם מחויבים לקחת וקורסים לבחירתם במערכת שעות סופית ומוגבלת בכמות השעות הקיימות ביום לימודים. בפרויקט זה בחרנו לבחון ולפתור את בעיית שיבוץ הקורסים – תהליך שיבוץ כלל הקורסים שנלקחים באותו הסמסטר ע"י סטודנט כלשהו למערכת שעות סמסטריאלית אופטימלית כפי שיוגדר בהמשך.

כדי למצוא פתרון אופטימלי לבעיה יש ראשית להגדירה היטב.

## הגדרת הבעיה

כדי להגדיר את הבעיה היטב נחלק את תיאורה לשלושה חלקים:

***הנחות והגדרות***

- <u>ראשית נגדיר הנחות כלליות ובסיסיות:</u>
  - לכל קורס קיימים לפחות סט אחד של קבוצות (כלומר לקורס קיים רק הרצאה) ולכל היותר שני סטים (הקורס מכיל הרצאה ותרגול), ובכל סט ישמר מספר הקבוצה.
  - משך כל קורס יהיה מספר שלם, מינימום שעה ומקסימום כמספר השעות שנותרו עד סוף היום בו הוא מתקיים.
  - יום למידה מוגדר כך:
    - בימים א'-ה' בין השעות 08:00-21:00
    - ביום ו' בין השעות 08:00-16:00
    - ביום שבת אין קורסים

    כלומר סה"כ ישנן 73 שעות למידה שבועיות.
  - שעות הבוקר יוגדרו בין השעות 08:00-14:00.
  - שעות הערב יוגדרו בין השעות 14:00-21:00.
  - נגדיר <u>התנגשות מלאה</u> בין שתי קבוצות כאשר הן חלות באותו היום ושעות ההתחלה והסיום שלהן זהות.
  - נגדיר <u>התנגשות חלקית</u> בין שתי קבוצות כאשר הן חלות באותו היום, אבל שעת ההתחלה ו/או הסיום שלהן שונות.
  - נגדיר <u>חפיפה מלאה</u> בין שני קורסים כאשר כל הקבוצות של שני הקורסים מתנגשים באופן מלא.
  - נגדיר <u>חפיפה חלקית</u> בין שני קורסים כאשר קיימות לפחות שתי קבוצות משני הקורסים (אחת מכל קורס) אשר מתנגשים באופן חלקי.

- במקרה של חפיפות בין קורסים נניח את הדברים הבאים:
  - לעולם לא תהיה חפיפה מלאה בין שני קורסי חובה – ניתן להניח זאת מכיוון שלא ניתן לסיים תואר ללא לקיחה וקבלת ציון עובר של כל קורסי החובה, לכן קיימת מערכת הכוללת רק את קורסי החובה של אותו הסמסטר ללא התנגשויות ביניהם.
  - במצב של חפיפה מלאה בין קורס חובה לקורס בחירה, העדיפות תינתן תמיד לקורס החובה.
  - במצב של חפיפה מלאה בין שני קורסי בחירה, הקורס המומלץ יותר (המספר המייצג את ההמלצה עליו יהיה גבוה יותר) יבחר. אם שני הקורסים מומלצים במידה שווה אז אחד מהקורסים יבחר רנדומלית.
- קורסי בחירה לעולם לא יהיו קורסי קדם לקורסים אחרים.

- <u>שנית נגדיר את מבנה של כל קורס:</u>
  - מספר הקורס – מספר שלם, חיובי וייחודי לכל קורס עד חמש ספרות.
  - שם הקורס.
  - האם מדובר בקורס חובה או בחירה – משמש כדי ליצור עדיפות לקורסי החובה על הבחירה. בנוסף, קורסי החובה יהיו מיוצגים על ידי השנה בה לוקחים אותם (א', ב', ג' או ד') וקורסי בחירה יהיו מיוצגים על ידי השנה 0.
  - מספר השעות של אותו הקורס – מספר שלם וחיובי בטווח [1,13] (הטווח המגביל את כמות השעות של כל קורס מוסבר בהנחות הבסיסיות לעיל).
  - תיאור הקבוצות – רשימה הבנויה ממחרוזות, כך שכל מחרוזת מורכבת מהיום בו הקבוצה מתקיימת, שעת ההתחלה-שעת הסיום, לדוגמה "Sunday 8-12".
  - המלצה – מספר שלם וחיובי בטווח [1,10] כך שציון 1 הוא הנמוך ביותר בעוד שציון 10 הוא הגבוה ביותר.
  - רשימת קורסי קדם – רשימה של מספרי קורסים אותם המשתמש היה צריך לסיים בהצלחה לפני שיבוץ קורס זה.
- <u>לבסוף נגדיר את המערכות וקידוד הקורסים:</u>
  - <u>נגדיר קורס כמחרוזת בינארית באורך X:</u>
    - מכיוון שלכל קורס יש לכל היותר 73 קבוצות, נזדקק ל-7 ביטים על מנת לייצג את כל הקבוצות. כלומר, אנו מקבלים כי X=7. לדוגמה:
      - לא נבחרה אף קבוצה – 0000000.
      - נבחרה קבוצה 01 – 0000001.

- נבחרה קבוצה 02 – 0000010.
- נבחרה קבוצה 73 – 1001001.

נשים לב! כי המחרוזת החל מ 1001010 עד 1111111 אינן מחרוזות חוקיות.

- <u>נגדיר מערכת באופן הבא:</u>
  - מכיוון שיתכנו לכל היותר 73 קורסים, נשרשר את המחרוזות 73 פעמים. סה"כ נקבל כי מערכת בנויה מ-511 ביטים – 511 ביטים אלו מייצגים מערכת מסוימת. למשל:
  - 00000000000000.....0000000 – המערכת בה לא נבחרה אף קבוצה.
  - 00000010000001.....0000001 – המערכת בה עבור כל קורס נבחרה קבוצה 01.
- <u>נגדיר סוגים של מערכות (סה"כ נגדיר 4 סוגים):</u>
  - <u>מערכת חוקית</u> ⇔ עבור כל קורס חובה בחרנו קבוצה קיימת מהטווח 01 עד 73, ועבור כל קורס בחירה בחרנו קבוצה קיימת מהטווח 00 עד 73.
  - <u>מערכת רעה</u> ⇔ המערכת חוקית, וגם מתקיימת חפיפה חלקית או מלאה בין לפחות שתי קבוצות של שני קורסים שונים שנבחרו.
  - <u>מערכת טובה</u> ⇔ המערכת חוקית, וגם לא מתקיימת חפיפה בין כל הקבוצות של הקורסים שנבחרו.
  - <u>מערכת אופטימלית</u> ⇔ מערכת טובה שבה אנו ממקסמים את פונקציות המטרה (יפורטו בהמשך).

***מגבלות***

<u>את מגבלות הבעיה חילקנו לשתי קטגוריות:</u>

- מגבלות קבועות, כלומר מגבלות שלא תלויות בקלט שהמשתמש מכניס למערכת:
  - לא ניתן לאפשר התנגשות בין שתי קבוצות קורסים שונים.
  - לא ניתן לבחור קורס מסוים אם המשתמש לא סיים בהצלחה את כל קורסי הקדם של אותו הקורס.
- מגבלות המשתמש, כלומר מגבלות שהמשתמש מכניס למערכת ותלויות בו לחלוטין:
  - ימים ו/או שעות בהן הוא לא יכול ללמוד, כלומר לא ניתן לשבץ קורסים בימים ובשעות הללו.

***פונקציות מטרה***

<u>את מטרות הבעיה חילקנו לשתי קטגוריות:</u>

- מטרות קבועות, כלומר מטרות שלא תלויות בקלט שהמשתמש מכניס למערכת:
  - שיבוץ הקורסים בצורה שכמות החלונות תהיה מינימלית.
- מטרות המשתמש, כלומר מטרות שהמשתמש מכניס למערכת ותלויות בו לחלוטין:
  - מקסום שיבוץ הקורסים בשעות הבוקר/הערב/בימים שלמים (כלומר המטרה היא להגיע למינימום כמות ימי למידה).

## שלבי הפתרון

***הקלט***

המשתמש יתן למערכת כקלט את השנה בה הוא לומד, קורסי חובה עליהם הוא צריך לחזור (אם יש כאלה), קורסי הבחירה שבחר, בחירת פונקציית מטרה וימים ו/או שעות בהן הוא לא יכול ללמוד.

***הפלט***

המערכת תחזיר מערכת שעות אופטימלית לפי פונקציות המטרה שהוגדרו ולפי בחירות המשתמש במידה וקיימת, אחרת תחזיר שלא קיימת מערכת שעות אופטימלית.

<u>נחלק את האלגוריתם לשני שלבים:</u>

***שלב ראשון – Pre-processing (שלב סינון הקורסים)***

לאחר שהמשתמש יזין את הנתונים וההעדפות שלו, המערכת תסנן את הקורסים לפי הקריטריונים הבאים:

- <u>סינון קורסי קדם:</u> עוברים על כל קורסי הבחירה ועל הקורסים שצריך לחזור עליהם – אם נמצא קורס בחירה שאחד מקורסי הקדם שלו הוא קורס שצריך לחזור עליו, נמחק את הקורס מהרשימה.
- <u>סינון העדפות משתמש:</u> אם נמצא קורס בחירה שנמצא ביום ואו שעה שהוגדרו על ידי המשתמש שלא ניתן ללמוד בהם, נמחק את הקורס מהרשימה.
- <u>סינון קורסי בחירה ראשוני:</u> אם קיימים מעל 74 קורסי חובה, נזרוק את המערכת, מכיוון שלא ניתן לסדר מערכת שעות לפי ההנחות וההגדרות שהגדרנו.

- <u>סינון חפיפות מלאות בין קורסים:</u> במצב בו קורס חובה וקורס בחירה חופפים בצורה מלאה, נמחק את קורס הבחירה, ובמצב בו שני קורסי בחירה חופפים בצורה מלאה וציון ההמלצה שלהם זהה גם הוא, אז נמחק את אחד הקורסים באופן שרירותי.

<u>מסוף שלב זה מתקיימים הדברים הבאים:</u>

- האלגוריתם לא יוכל ליצור מערכות לא חוקיות.
- המערכת תחשיב את כל הקורסים שהמשתמש בחר לאחר הסינון כקורסי חובה – המשמעות היא שאו שתוחזר מערכת אופטימלית עבור <u>כל</u> הקורסים, או שלא תוחזר מערכת בכלל.

***שלב שני – שלב הבחירה***

עבור כל פונקציית מטרה (קבועה ולפי בחירת המשתמש) המערכת תחזיר את קבוצת קורסים שתואמת לבחירות המשתמש ומניבה את הציון המקסימלי עבור השילוב של הפעלת פונקציות המטרה.

## הדרך למציאת הפתרון

בשלב זה נתאר את התהליך והאלגוריתמים שהגענו אליהם בדרך למציאת האלגוריתם היעיל ביותר לפתרון הבעיה:

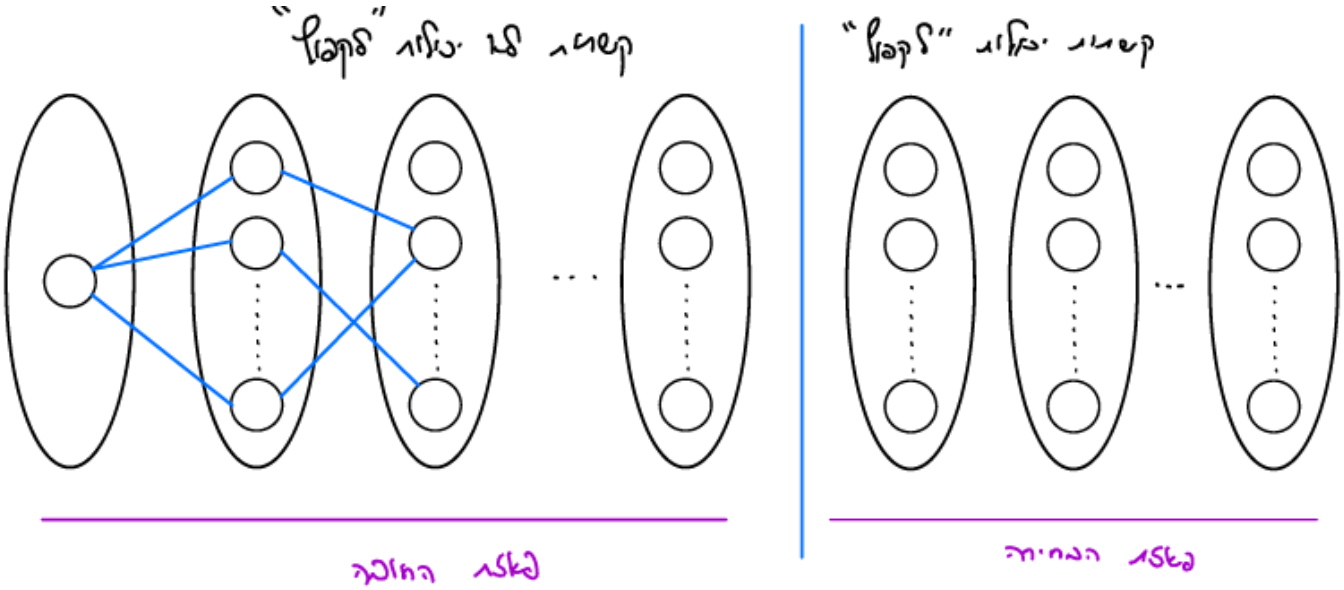
***פתרון נאיבי***

נעבור על כל הסידורים האפשריים של כל קבוצות הקורסים ונבחר את האופטימלי ביותר. אופן חישוב כמות הסידורים האפשריים שקול לבעיית סידור אובייקטים בשורה, לכן המערכת תעבור על !73 אפשרויות, כלומר זמן הריצה יהיה $\Theta(73!)$ – סדר גודל גדול ולא אופטימלי.

***פתרון עם גרף שכבות***

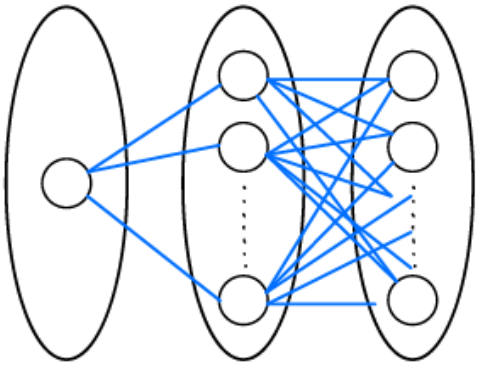
נגדיר גרף שכבות, כך שכל שכבה מייצגת סט של קורס, כל קודקוד מייצג קבוצת קורס וקיימת קשת המחברת שני קודקודים אמ"מ הקשת היא בין שכבה $V_i$ ל-$V_{i+1}$, וששתי הקבוצות לא חופפות בזמנים. נשים לב כי מספר השכבות הוא לכל היותר 146 (לפי ההנחה שלכל קורס יש לכל היותר שני סטים, הרצאה ותרגול, כלומר פי שניים ממספר הקורסים האפשריים לשיבוץ) ומספר הקודקודים הוא לכל היותר 73 (כמספר הקורסים האפשריים לשיבוץ). נגדיר שצריך קודקוד התחלה יחיד בגרף השכבות, כלומר יהיו לכל היותר 73 גרפים שונים לאחר פיצול השכבה הראשונה כך שבכל גרף יהיה קודקוד התחלה יחיד. הגרף יהיה מחולק לשני חלקים: החלק הראשון יהיה מורכב מקורסי החובה, כך שבחלק זה קשתות מתחברות רק לשכבות עוקבות (כלומר לא ניתן לחבר קשת בין שכבה $V_i$ ל-$V_{i+k}$ כאשר k > 1), והחלק השני יהיה מורכב מקורסי הבחירה, כך שבחלק זה קשתות יכולות "לדלג" לשכבות

הבאות (נשים לב שעדיין נשמר סדר מעבר הקשתות – מהנמוך לגבוה):




האלגוריתם יסכום את משקלי הקשתות (שיכילו בתוכן את שקלול הציונים של פונקציות המטרה) ויחזיר את המסלול בעל המשקל המקסימלי, כלומר המערכת האופטימלית בעלת הציון הגבוה ביותר עבור פונקציות המטרה.

<u>הבעיה</u> עם פתרון זה הוא שהקשתות הן "דינמיות", כלומר הגרף מניח באופן שגוי שהמסלול $(u, v_1, w)$ שקול למסלול $(u, v_2, w)$.



בנוסף, אם נרצה לוודא שהקשתות לא יהיו "דינמיות", אז נצטרך ליצור "קליק" בגרף, כלומר ליצור קשת בין כל קודקוד לכל קודקוד בשכבה שאחריו:

<u>הבעיה</u> עם נסיון שיפור זה הוא שזמן הריצה אפילו גבוה מזה של הנאיבי – $(73^{73})$.

***פתרון תכנות דינמי***

בניסיון לייעל את הפתרון באמצעות גרף השכבות, הצענו פתרון המחזיר את המערכת האופטימלית באופן הבא: במקום להריץ את האלגוריתם כל פעם עד השכבה האחרונה, האלגוריתם יזכור את מה שכבר נסרק, הוא יעצור בקודקוד ויבדוק האם חישב את תת העץ הזה בעבר, כלומר באופן תיאורטי הגרף יסרק פחות פעמים בכך שעבור כל קודקוד בשכבה גבוהה נוכל בשלב מסוים למצוא מערכת אופטימלית ב-O(1) (נשים לב שהאלגוריתם לא יזכור אלו קורסים נבחרו, אלא אלו שעות נבחרו).

<u>להלן הפסאודו קוד:</u>

*כל עוד ניתן לבחור:*

*הזכר אם ראית מערכת כזו בעבר:*

*אם כן, המשך לבחור מאותה הנקודה שאתה זוכר.*

*אחרת, בחר באקראי קורס בתנאי שהמערכת חוקית.*

*זכור את המערכת החדשה.*

*אם המערכת טובה יותר מהמערכת הקודמת, שמור אותה.*

<u>הבעיה</u> עם פתרון זה היא שנצטרך לזכור $2^{73}$ מצבים שונים. אומנם, האלגוריתם יכול לרוץ בצורה טובה על קלטים בגודל "הגיוני" – האלגוריתם מדמה את ההתנהגות האנושית בבחירת הקורסים בכך שהוא זוכר מערכות שנתקלנו בהן בעבר ומנסה לשפר אותן.

## הפתרון – אלגוריתם אבולוציוני (גנטי)

***הסבר השיטה***

<u>השיטה מחולקת לחמישה שלבים:</u>

1. **שלב הבריאה:** ניצור כמות מסוימת של מערכות – כל מערכת בשלב זה תהיה חוקית, אך לא בהכרח תהיה אופטימלית או אפילו טובה.
2. **שלב הבחירה (המבול):** בשלב הזה באמצעות פונקציה מסוימת שנגדיר בהמשך, נבחר אלו מהפתרונות יוכל להמשיך לדור הבא, ואלו מהפתרונות ימחקו. הבחירה תתבצע כך: נפעיל פונקציה או "אסון טבע" כלשהו, מי שישרוד את אסון הטבע יוכל להמשיך לדור הבא, ומי שלא ימחק ולא יזכה להמשיך לדור הבא. בכך יובטח לנו שרק המערכות הכי טובות ימשיכו לדור הבא.
3. **שלב הזיווג:** בשלב זה נבחר את המערכות שנזווג, כתוצאה מהזיווג יוולדו מערכות חדשות (שלב זה הוא רק בחירת הזוגות).
4. **שלב היצירה:** בשלב זה נוצרות המערכות החדשות מלקיחת גנים מההורים שלהם.
5. **שלב המוטציה:** בשלב זה נעבור על כל אחד מהילדים ובהסתברות מסוימת ונבצע בו מוטציה. שלב זה פוטנציאלי ליצור לנו פתרונות חדשים ואף טובים יותר שלא יכלנו למצוא משלב היצירה.
6. **שלב הלידה:** בשלב זה נוסיף את המערכות החדשות שנוצרו לאוכלוסיה הקיימת.

***פסאודו קוד***

נגדיר את מספר הבריאות להיות מספר הפעמים שנרצה להריץ את האלגוריתם.

נגדיר את אחרית הימים להיות זמן או מספר דורות שניתן לאלגוריתם לרוץ.

1. בצע את שלב הבריאה.
2. בצע את שלב המבול.
3. בצע את שלב הזיווג.
4. בצע את שלב היצירה.
5. בצע את שלב המוטציה.
6. בצע את שלב הלידה.
7. בצע את שלבים 2 עד 6 עד מציאת הפתרון האופטימלי, או אחרית הימים, מה שמגיע קודם.
8. כל עוד לא ביצענו את שלב 1 כמספר הבריאות המקסימלי, חזור לשלב 1.

***הסבר האלגוריתם***

1. **ביצוע שלב הבריאה:** שלב זה הוא יחסית פשוט – ראשית נבחר מתוך הקורסים שנבחרו קבוצות באופן אקראי אבל מוכוון לפונקציית המטרה, לדוגמה אם המשתמש בחר שהוא מעדיף ללמוד בבוקר אז הבחירה האקראית תתבצע מבין הקבוצות שמתקיימות בבוקר. לאחר מכן, ניתן לאלגוריתם לבחור כמה מערכות הוא יוצר. אם קורה והמערכות נכחדות (הכחדה מוגדרת כאשר נותרה מערכת אחת באוכלוסיה לכל היותר), ובהנחה ואסונות הטבע היו סבירים, אנו ניצור בריאה חדשה עם אוכלוסיה התחלתית גבוהה מהאוכלוסיה הקודמת. חשוב לציין – באלגוריתם שלנו נעדיף לבצע פעם אחת את הבריאה ולהמנע כמה שיותר מהכחדות מלאה או הרוב, אך נשאיר את האופציה לבריאות נוספות במידה וכן הגענו להכחדות הנ"ל.
2. **ביצוע שלב המבול:** בשלב זה המבול מקדם אותנו לפונקציית המטרה. המבול נועד לשמור על יחס ילודה-תמותה, בנוסף לשיטת ממשל-סין בשלב 4 – אם נראה כי הילודה גבוהה מידי נפעיל מבול חזק יותר, וההפך אם התמותה גבוהה מידי.
3. **ביצוע שלב הזיווג:** בשלב זה נמיין את האוכלוסייה שלנו מהחזק לחלש. הזיווג יתבצע לפי שיטת "החזקים לחזקים" – כל אחד יזווג למי שהכי קרוב אליו מבחינת חוזק, בהנחה שכבר לא זווג למישהו אחר.

4. **ביצוע שלב היצירה:** בשלב זה אנו בוחרים שני דברים – כמה ילדים יהיה לכל זוג ואלו גנים נקח לאב ולאם. בחירת כמות הילדים תהיה אקראית ותעבוד לפי שיטת ממשל-סין. נרצה כי מדור לדור ישמר יחס ילודה-תמותה מסוים – אם הילודה תהיה גבוהה מידי לא נאפשר לזוג ליצור הרבה ילדים, ואם אחוז התמותה יהיה גבוה נדחוף זוגות ליצור הרבה ילדים. אופן בחירת הגנים יתבצע בצורה אקראית.
5. **ביצוע שלב המוטציה:** את שלב זה נבצע על כל הילדים החדשים שנוצרו בשלב היצירה האחרון, כך שהאלגוריתם יכול לשנות כל גן בהסתברות מסוימת.
6. **ביצוע שלב הלידה:** שלב פשוט שבו נוסיף את כל הילדים החדשים לאוכלוסיה הקיימת.

***הסבר נכונות***

נראה כי תמיד נגיע לפתרון האופטימלי אם נריץ מספיק זמן או מספיק דורות, כלומר בהסתברות גבוהה נגיע לפתרון אופטימלי או אחד שמאוד קרוב אליו. נשים לב שמדור לדור אנו מקבלים כי ה"אלפא" מתחזק או שווה לעוצמה של ה"אלפא" בדור הקודם (מכיוון שבשלב המבול אנחנו מוחקים את החלשים ומשאירים את החזקים, בפרט את ה"אלפא", או הכי חזק נשאר). הדרך היחידה שה"אלפא" ימחק היא אם המבול מחק את כולם, וזה בסתירה לכך שאנו רוצים כי המבול יהיה "מבול חלש".

***פונקציות מתן הציונים למערכות***

- ציון הניתן לכל מערכת – 0-100:
  - 100 מתקבל עבור מערכת אופטימלית.
  - 60-99 מתקבל עבור מערכת טובה.
  - 0-59 מתקבל עבור מערכת רעה.
- הציונים ניתנים ביחס לשאר המערכות שניבנו באותו הדור.
- עבור מערכת רעה, האלגוריתם מחשב את כמות ההתנגשויות ונותנת למערכת ציון לפי הטווח – אם אין התנגשויות הציון המקסימלי שיכולה המערכת לקבל הוא 59.
- עבור מערכת טובה, האלגוריתם מחשב את כמות השעות/ימים העונים על דרישת פונקציית המטרה התלויה בהעדפת המשתמש (לדוגמה כמה שעות מתוך כלל שעות הקורסים מתרחשות בבוקר עבור משתמש המעדיף ללמוד בבוקר), מחשב את כמות החלונות הקיימים במערכת ומחבר את תוצאות הציונים היחסיים לפי יחס של 30-70 לטובת פונקציית המטרה לפי בחירת המשתמש.

***הפרמטרים ששימשו אותנו בהרצת האלגוריתם***

- **מספר הבריאות:** 100.
- **מספר הדורות:** 40.
- **כמות האוכלוסיה ההתחלתית:** 60.
- **קבלת מוטציה:** 9%.
- פיצוץ אוכלוסין: פי 3.5 מהאוכלוסיה בבריאה.
- סכנת הכחדה: 0.75 מהאכלוסיה בבריאה.
- מקדם ילודה מצב רגיל: 1-5 ילדים לזוג.
- מקדם ילודה פיצוץ אוכלוסין: 1-3 ילדים לזוג.
- מקדם ילודה בסכנת הכחדה: 3-5 ילדים לזוג.
- מקדם מבול מצב רגיל: נבחר 50% טובים ביותר.
- מקדם מבול פיצוץ אוכלוסין: נבחר 70% טובים ביותר.
- מקדם מבול בסכנת הכחדה: נבחר 30% טובים ביותר.
- מנגנון לקיחת גנים מההורים לפי הטלת מטבע, עבור כל גן: עץ עבור אב , פלי עבור אם.

***בדיקת קבועים למציאת מערכת אופטימלית ביותר***

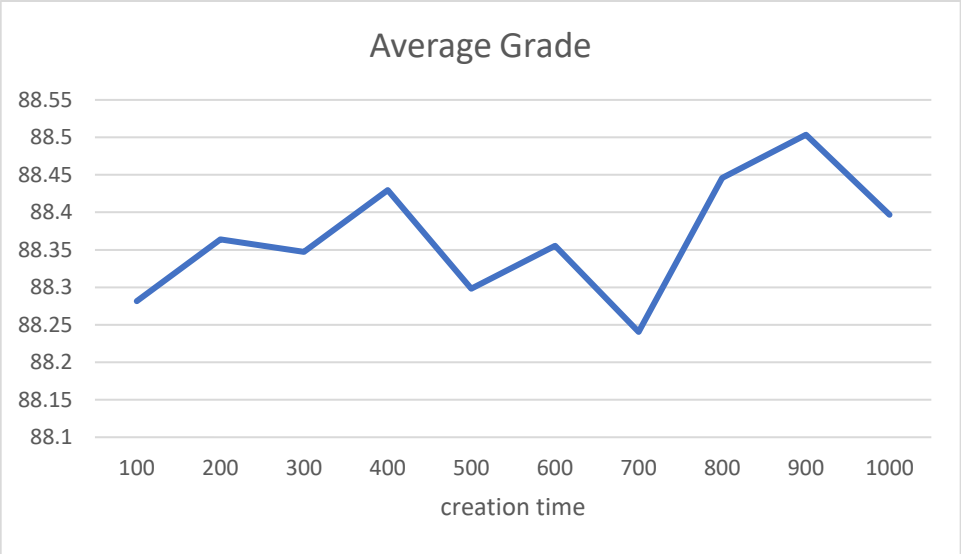
על מנת למצוא את הפרמטרים המניבים את המערכת האופטימלית ביותר, הרצנו בדיקות על המערכת שלנו עבור הפרמטרים הבאים:

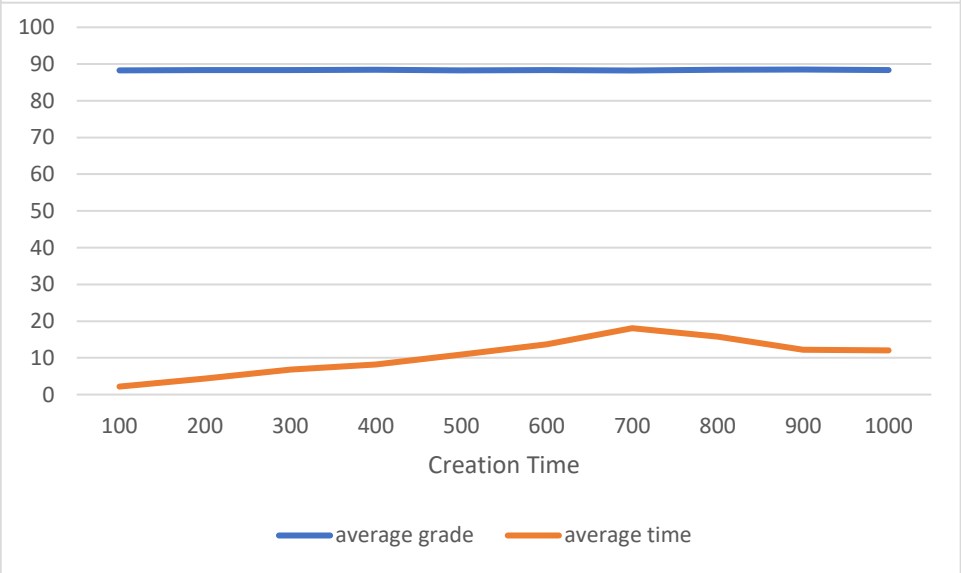
- מספר הבריאות (ערך ברירת מחדל – 100).
- מספר הדורות באלגוריתם (ערך ברירת מחדל – 10).
- כמות האוכלוסיה ההתחלתית (ערך ברירת מחדל – 10).
- אחוז המוטציה (ערך ברירת מחדל – 10).

עבור כל בדיקה נבחר פרמטר אחד לבדיקת ערכים בתחום מסוים ואת שאר הפרמטרים נשאיר קבועים. בנוסף עבור כל ערך בתחום הנבדק נריץ 100 בדיקות וניקח את התוצאות הממוצעות.

<u>להלן תוצאות הבדיקות:</u>

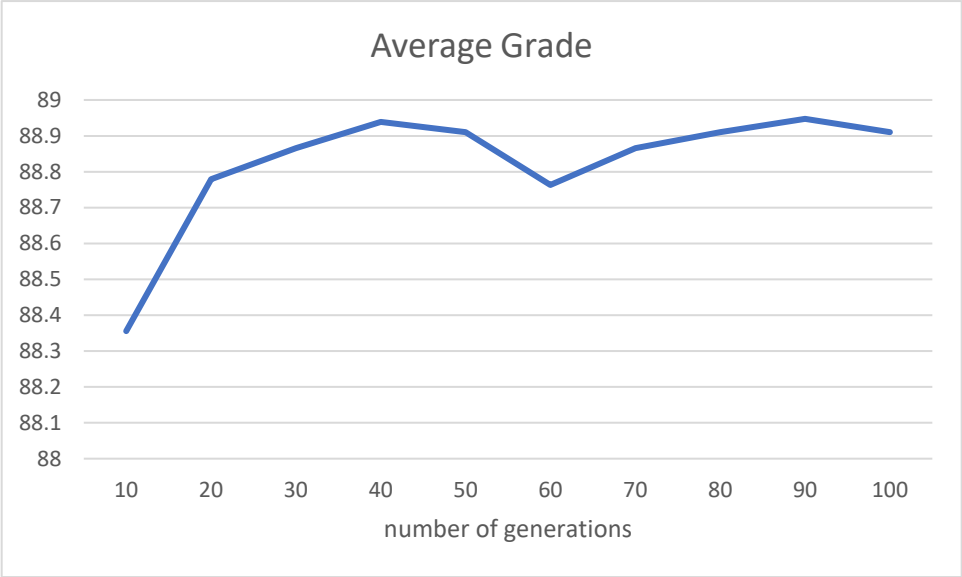
- עבור מספר הבריאות בתחום 100-1000 (בקפיצות של 100):

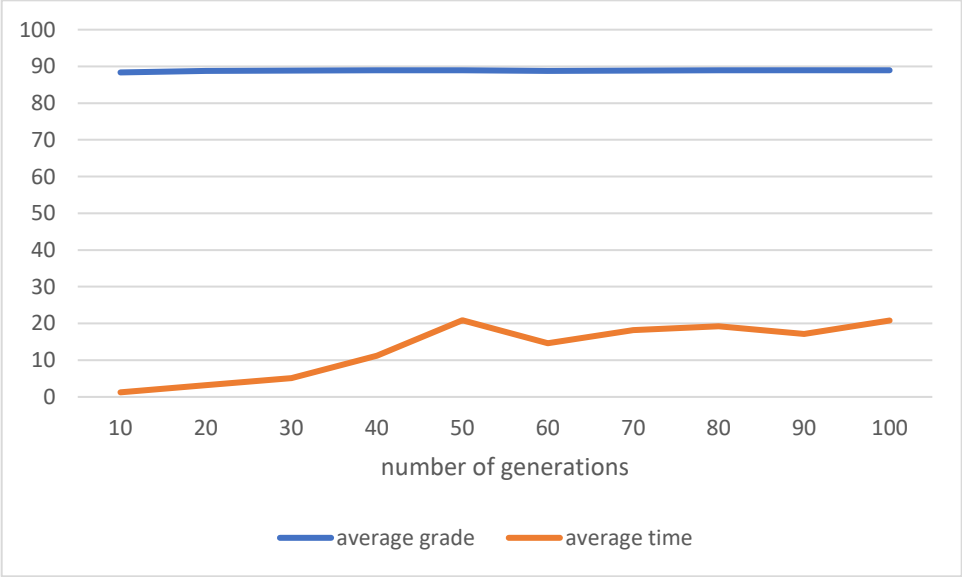



ניתן לראות כי הציון לא מושפע באופן דרסטי מהגדלת מספר הבריאות (הירידות עבור חלק מהמספרים נוצרו בגלל עליה במספר החלונות במערכת מ-0 לחלון אחד או שניים לרוב), אבל זמן הריצה עולה דרמטית.
לכן הערך האופטימלי עבור מספר הבריאות לפי הבדיקות הוא 900 (הרצה עבור 900 בריאות יקח בממוצע 10 שניות, כלומר הרבה זמן, לכן השתמשנו ב-100 בריאות, המספק ציון קרוב מאוד לציון הגבוה ביותר).

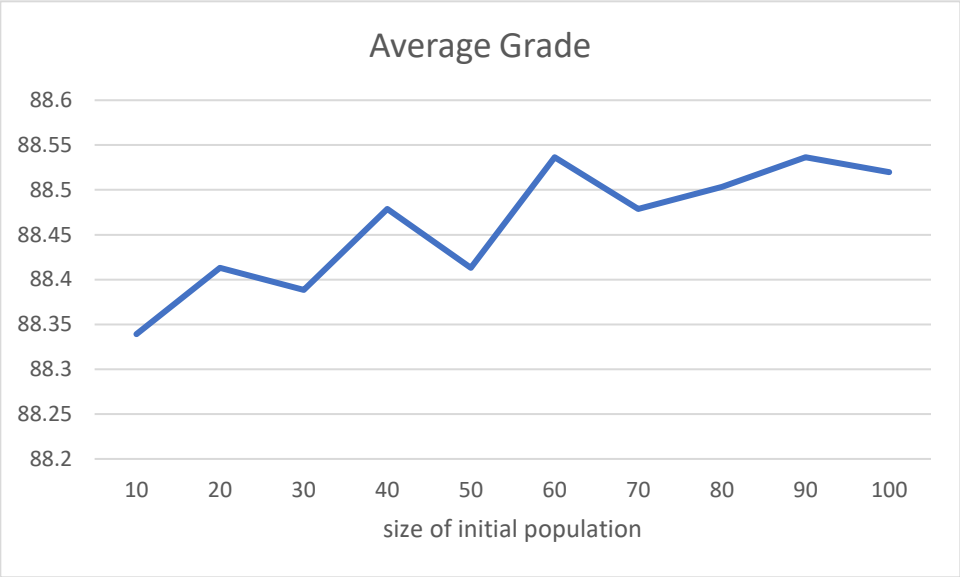
- עבור מספר הדורות בתחום 10-100 (בקפיצות של 10):

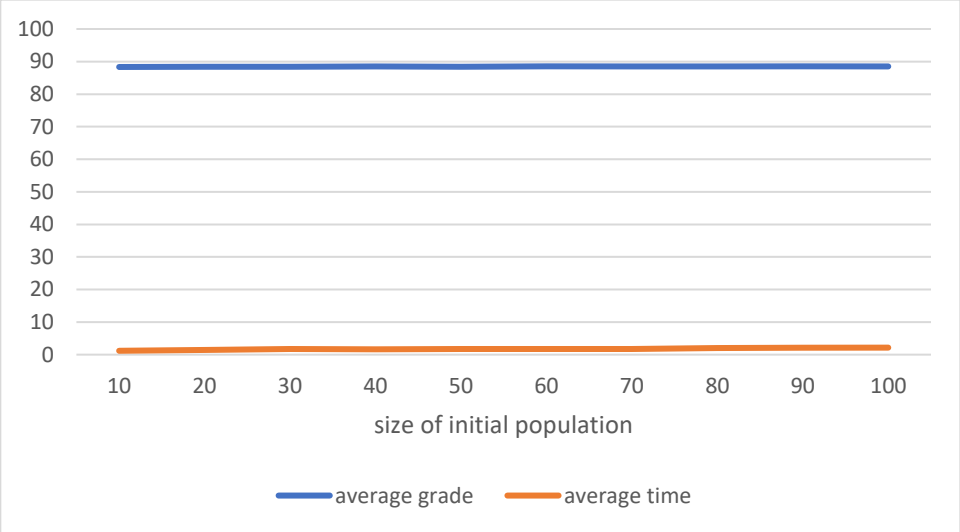



ניתן לראות כי גם ציון המערכת האופטימלית ביותר וגם זמן הריצה עולים, לכן הערך האופטימלי של מספר הדורות לפי הבדיקות הוא 90 (בגלל העליה המשמעותית בזמן הריצה, לקחנו את הערך עם הציון הקרוב ביותר למקסימלי, כלומר עבור 40 דורות).

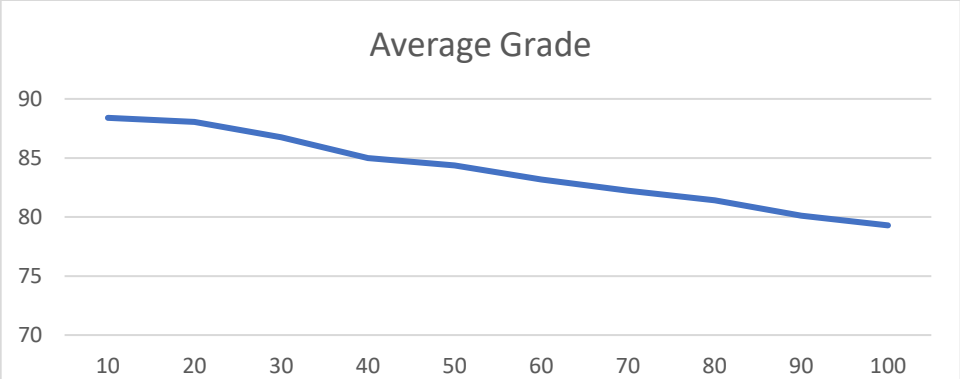
- עבור כמות האוכלוסיה ההתחלתית בתחום 10-100 (בקפיצות של 10):

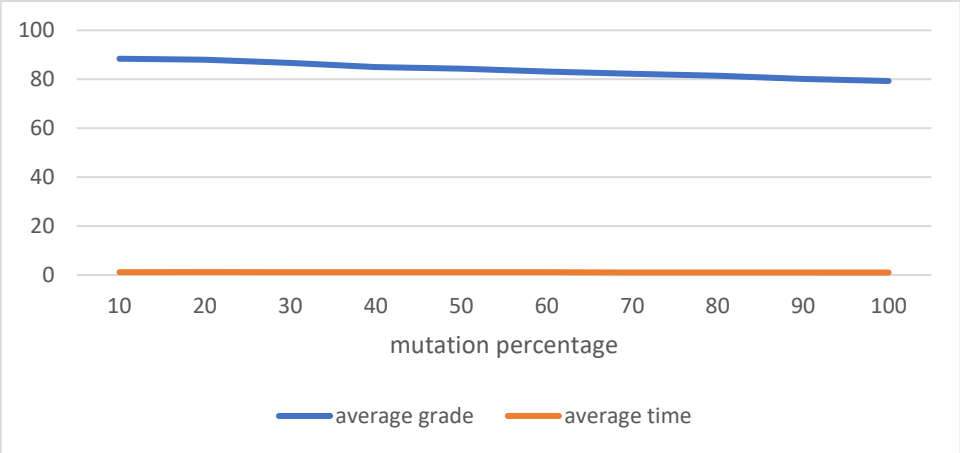


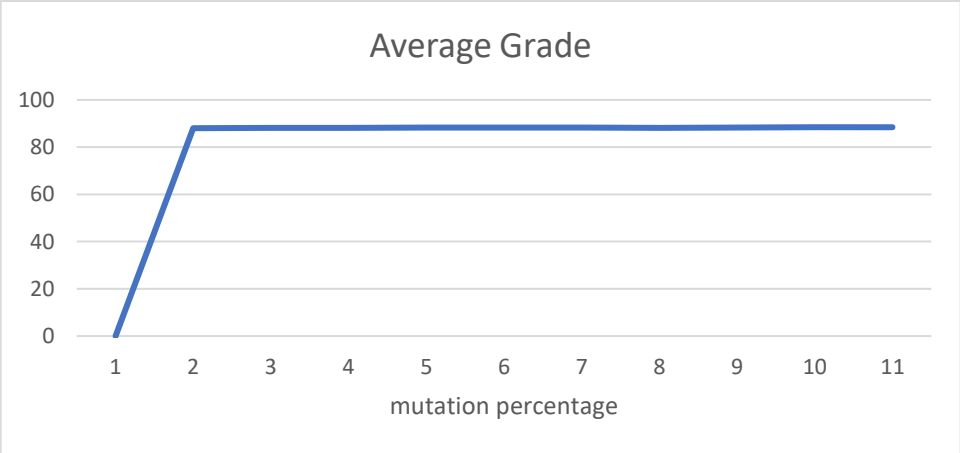




ניתן לראות כי ציון המערכת האופטימלית ביותר במגמת עליה (הירידות עבור חלק מהמספרים נוצרו בגלל עליה במספר החלונות במערכת מ-0 לחלון אחד או שניים לרוב), בעוד שזמן הריצה לא משתנה משמעותית, לכן הערך האופטימלי של כמות האוכלוסיה ההתחלתית לפי הבדיקות הוא 60.

- עבור אחוז המוטציה בתחום 10-100 (בקפיצות של 10):




ניתן לראות שזמן הריצה לא משתנה דרמטית, אבל ציון המערכת האופטימלית ביותר במגמת ירידה, לכן בדקנו גם את ערכי הציונים עבור אחוז מוטציה בין 0-10 (עבור 0 אחוז הוא לא מוצא מערכת אופטימלית), המתוארים בגרף התחתון – מכיוון שגרף זה במגמת עלייה, אז הערך האופטימלי ביותר לאחוז המוטציה לפי הבדיקות הוא 9%.

# ביבליוגרפיה

https://towardsdatascience.com/genetic-algorithm-implementation-in-python-5ab67bb124a6

https://towardsdatascience.com/using-genetic-algorithms-to-schedule-timetables-27f132c9e280

https://en.wikipedia.org/wiki/Genetic_algorithm

https://github.com/ahmedfgad/GeneticAlgorithmPython