



AI Final Project

Exam Scheduling and Advisor

תוכן עיניים

2	מבוא
3	סקירה קצרה של האלגוריתמים
4-11	מימוש האלגוריתמים עבור בעיית שיבוץ המבחנים
12-14	מימוש האלגוריתמים עבור בעיית שיבוץ האולמות
15-16	מימוש האלגוריתמים עבור יעוץ על ההחלטה לגבי מועד ב
17	פתרון שיבוץ המבחנים למשבצות זמן באמצעות תורת המשחקים
18-41	השוואה בין אלגוריתמים
18-28	בעיית המבחנים למשבצות זמן
29-30	תורת המשחקים למול חיפוש
31-39	בעיית שיבוץ האולמות
40-41	בעיית היעוץ ID3
42-43	דיון בתוצאות והסקת מסקנות
44-45	סיכום והצעות לשיפור
46-47	הוראות הרצת התוכנית
48	קיבצי התוכנית
49	תמונות ונספחים

מבוא

בפריקט זה החלטנו לעסוק בנושא של תקופת המבחנים, בחלק מעיסוק זה בחרנו להתמקד בשתי בעיות עיקריות האחת הינה שיבוץ לוח מבחנים הוגן והשנייה הינה מתן ייעוץ לסטודנטים האם לגשת למועד ב או לא.

חלק ראשון:

המטרה שלנו בחלק זה הינה לשבץ לוח מבחנים הוגן הן עבור הסטודנטים והן עבור הסגל. בחרנו בנושא זה מכיוון שחוויתנו קושי בניהול הזמנים במהלך תקופת הבחינות עקב שיבוץ צפוף של בחינות רלוונטיות. כדי לפתור את הבעיה בחרנו למדל אותה באופן הבא: צמצמנו את הבעיה לשיבוץ הבחינות עבור סמסטר מסוים ושנה מסוימת (למשל, שיבוץ מבחנים לשנה א', סמסטר א') ובנוסף התעמקנו במבחנים המתקיים בקמפוס גבעת רם. לאחר שיבוץ המבחנים בתאריכים שיבצנו בנוסף כיתות למבחנים עבור המידול הקודם. נקטנו בשתי גישות על מנת לפתור את הבעיה האחת הינה באמצעות תורת המשחקים והשנייה בעזרת מידול של הבעיה כבעיית CSP ושימוש באלגוריתמים הבאים:

- CSP, WCSP
- Simulated Annealing (SA)
- Gradient Descent (GD), Randomized Gradient Descent (RGD)
- Genetic Algorithm (GA)

אנו חושבים ששימוש ב-CSP מציג צורת חשיבה טבעית בפתרון בעיה מסוג זה מכיוון שיש צורך להתייחס לאילוצים גם של הסטודנטים וגם של הסגל. בנוסף, עקב כך שמרחב החיפוש עצום אלגוריתמים מסוג Gradient descent ו-Genetic algorithm יהוו פתרון טוב יותר לבעיה הנ"ל.

מנקודת מבט שונה שימוש בתורת המשחקים יכול להראות לנו כיצד הסטודנטים יכולים לפעול יחד כדי כך שכול חוג יוכל להגיע לשיבוץ המיטבי עבורו. ננסה להציג את הגישה הזו על ידי יצירת קואליציות בין החוגים השונים.

חלק שני:

בחלק זה התמקדנו בבעיה של המלצה לסטודנט האם לגשת למועד ב או לא בהינתן גורמים שונים. המטרה שלנו בחלק זה היא לעזור לסטודנטים עם התלבטות זו, שאנו מאמינים שיצא לכול סטודנט לחוות לפחות פעם אחת בתואר. כדי לעשות זאת יצרנו סוכן שמטרתו לעזור לסטודנט לקבל את ההחלטה על הסוגייה. בבעיה זו אספנו מידע על ידי סקר של סטודנטים מחוגים שונים ולאחר מכן השתמשנו ב3ID כדי לייצר עץ החלטה על מנת להגיע לתוצאה.

סקירה קצרה של האלגוריתמים

1. Constraint Satisfaction Problem
בעיות של השמת ערכים למשתנים כך שיש אילוצים מסוימים בין ערכים של משתנים מסוימים. בבעיה הזו קיימים משתנים, השמות (הצבות של משתנים), דומיין (הערכים האפשריים למשתנה ספציפי) ואילוצים (פונקציית שהערכים של המשתנים צריכים לקיים).
2. Weighted Constraint Satisfaction Problem
הבעיה זו היא הכללה של הבעיה הקודמת (מסעיף 1) כך שחלק מהאילוצים יכולים להיות מופרים. בנוסף, קיימת פונקציית משקל שמקבלת משתנים ואילוצים ומחזירה את המשקל שלהם (מספר רציונלי). בצורה זו ניתן לתת העדפה לפתרון מסוים על פני פתרון אחר לפי פונקציית המשקל.
3. Lecture Simulated Annealing
בפרויקט זה השתמשנו בווריאציה של האלגוריתם Lecture Simulated Annealing כאשר האלגוריתם שלנו עושה צעד אקראי בכל פעם ומתקדם אם יש שיפור במידה ואין שיפור הוא יבצע את הצעד בהסתברות מסוימת התלויה בטמפרטורה.
4. Gradient Descent
שיטת חיפוש מקומי כאשר מתחילים ממצב רנדומלי ובכל צעד מתקדמים לעבר השכן הכי "נמוך" של המצב הנוכחי. במידה וכל השכנים גבוהים או שווים למצב הנוכחי, נחזיר את המצב הנוכחי או במידה ונגיע למספר האיטרציות הנקבע מראש. אנו מודדים את "הגובה" (איכות) של כל המצבים לפי פונקציית יוריסטיקה. האלגוריתם הזה לעיתים מחזיר פתרון אופטימלי חלקי, מכיוון שהוא נתקע במינימום מקומי.
1. Randomized Gradient Descent
הבעיה הזו היא הכללה של האלגוריתם Gradient descent המאפשרת "בריחה" ממינימום מקומי על ידי כך שמדי פעם אנו מאפשרים לעשות צד "רע" (צעד שלא מקדם אותנו לכיוון המינימום).
5. Genetic Algorithm
אלגוריתם חיפוש, מידול ומיטוב אשר משלב אלמנטים אפשריים לפתרון הבעיה (הכלאה מוטציה וברירה טבעית) ומפעילים הליכים של ברירה מלאכותית כדי לבחור את המועמדים שיעברו לשלבים הבאים. רעיון תכנותי בסיסי זה מושפע מההצלחה של האבולוציה בפתרון בעיות אמיתיות.
6. ID3
האלגוריתם מייצר עץ החלטה באמצעות מערך נתונים המועבר. אלגוריתם פועל באופן הבא: מחשב אנטרופיה עבור כל תכונה של כל קודקוד בעץ, מחלק את התשובות לתתי קבוצות עם התוצאה המקסימלית, מייצר עץ החלטה המכיל את התכונה לעיל, חוזר על התהליך עבור התכונות הנותרות. התהליך יסתיים כאשר נגיע למצב שקיימת החלטה בכל עלה.

מימוש האלגוריתמים עבור פתרון בעיית שיבוץ המבחנים

2. Constraint Satisfaction Problem

הגדרת הבעיה:

Variables – (v_1, \dots, v_n) when n is the amount of exams

Domains – $\{v_i: (t_1, \dots, t_k) \text{ for } i \text{ in } n\}$ when (t_1, \dots, t_k) is a subset of all the valid time slots for the variable v_i

Constraints:

- $(v_i,)$ for i in n s.t $\forall i \exists j$ s.t $v_i: t_j$ and $\forall i \forall j v_i: t_j \rightarrow \forall k \neq j \neg(v_i: t_k)$
- (v_i, v_j) for i, j in n s.t $\forall i, j \forall k v_i: t_k \rightarrow \neg(v_j: t_k)$
- (v_i, v_j) for i, j in n s.t v_i is the first attempt of $v_j \rightarrow t_{v_j} - t_{v_i} \geq 14$
- (v_i, v_j) for i, j in n s.t v_i is moed a and v_j is moed b $\rightarrow t_{v_j} > t_{v_i}$
- (v_i, v_j) for i, j in n s.t v_i and v_j in the same faculty $\rightarrow \text{abs}(t_{v_j} - t_{v_i}) < d_{f_k}$

כאשר d_{f_k} הינו ההפרש המינימלי של ימים בין המבחנים בפקולטה f_k

האילוצים במילים

- לכול מבחן יש שיבוץ אחד בדיוק
- אין שני מבחנים שמשובצים באותה משבצת זמן
- ההפרש המינימלי בין מועד א למועד ב באותו מקצוע הוא שבועיים
- כל מועדי הא משובצים לפני מועדי הב
- מבחנים מאותה פקולטה משובצים במרחק גדול או שווה להפרש המינימלי שהוחלט עבור אותה פקולטה.

מימוש האלגוריתם:

תחילה בנינו את המשתנים והדומיין של הבעיה, בתחילת הפתרון הדומיין של כל המשתנים זהה והוא כל האפשרויות לשבץ מבחנים בתקופת המבחנים (יום בתקופה זו בשעות 9:00, 13:30 ו17:00 בימים ראשון עד חמישי וכן בשעה 9:00 ביום שישי).

אחרי בניית הדומיין בנינו את האילוצים כפי שמפורט לעיל. לאחר שהגדרנו את הבעיה פתרנו אותה באמצעות מספר יורסטיקות שראינו בשיעור.

Arc3:

יורסטיקה זו מהווה מעין עיבוד מקדים לבעיה שכן תפקידה הוא להוריד צלעות שיוצרות חוסר עקביות עם האילוצים שנקבעו בהגדרת הבעיה עוד לפני שיבוץ הבעיה, למשל עבור הבעיה שלנו במבחנים שהם מועד ב נוכל להסיר את 14 הימים הראשונים שכן כל הצבה שם תוביל לכך שמועד ב יהיה פחות מ14 ימים לאחר מועד הא שלו. אופן הפעולה של יורסטיקה זו הינה מעבר על כל המשתנים ועדכון הדומיין שלהם בהתאם לאילוצים שנקבעו.

Backtracking:

צורת פתרון זו היא אינה יורסטיקה אלה צורת הפתרון הבסיסית שראינו בביתה, בגדול היא פועלת כשמה, בכול פעם אנו משבצים מבחן במשבצת זמן ספציפית ואם נתקלנו בבעיה (כלומר לוח הזמנים שלנו לא עומד באילוצים) נחזור למצב הקודם וננסה שיבוץ אחר. פתרון זה ברעיונו מבזבזי מאוד שכן יכול להיות שנדע כבר שאנו נתקלים בבעיה הרבה לפני שניתקל בה ובכול זאת נמשיך בחיפוש ונגלה אותה רק בהמשך (למשל שיבצנו מועד א במקום שלא יאפשר לנו לשבץ את כל מועדי הב) אך במקרה שלנו לא היה הרבה הבדל בין פתרונות עם יורסטיקות יופרט על כך בהמשך

:Minimum Remaining Variables

באוריסטיקה זו נבחר לשבץ את המשתנה שיש לו הכי מעט משבצות זמן חוקיות בדומיין. הרעיון מאחורי שיטה זו הינו שככל הנראה שיבוץ של המשתנה הכי מוגבל יגרום לבעיות מוקדם יותר, ולכן מספר הפעמים שנאלץ לחזור יהיה מועט יותר. וגם הסבירות לבעיות בצורת שיבוץ זו קטנה יותר, למשל עם למבחן מסויים נשארו שתי מסגרות זמן חוקיות אז אם נשבץ מבחן אחר לפניו בסבירות גובהה יחסית נשבץ אותו בתאריך שיחסום את המבחן שנשארו לו שתי מסגרות זמן. אך יוריסטיקה זו לא הניבה תוצאות טובות יותר מbacktracking בשילוב עם 3arc בבעיה שלנו. אנו סבורים כי סיבה אפשרית לכך הינה שמרחב החיפוש שלנו גדול יחסית, ולכן לא יהיה צמצום משמעותי בבחירת המשתנים בצורה זו.

: Degree Heuristic

באוריסטיקה זו נבחר לשבץ את המשתנה שיש לו הכי הרבה אילוצים. הרעיון מאחורי שיטה זו הינו שבשיבוץ של משתנה שיש לו יותר אילוצים לפני משתנה שיש לו פחות אילוצים, אנו מאפשרים מרחב גדול יותר של שיבוצים חוקיים עבור משתנה זה (מכיוון שכאשר נשבץ אותו יהיו יותר משבצות זמן פנויות) ולכן הסבירות שנתקע ללא שיבוץ עבור משתנה זה קטנה. גם האוריסטיקה הזאת לא עבדה טוב יותר מbacktracking רגיל בנוסף ל3arc אך כן ראינו שיפור בביצועים על סדר שונה של הקורסים (איך שהם הגיעו מקבוצ האקסל) ובשימוש ללא 3arc.

:Least Constraining Value

באוריסטיקה זו נבחר לשבץ את הערך שמוביל לכך שלאחר שיבוצו מספר ההגבלות על המשתנים יהיה מינימאלי. הרעיון מאחורי יוריסטיקה זו הינו שבבחירה של הערך עם המספר הנמוך ביותר של אילוצים הסבירות לבעיות בשיבוץ עתידי יורדת (מעצם זה שיש פחות אילוצים יהיה יותר קל לעמוד בהם). גם האוריסטיקה הזאת לא עבדה טוב יותר מbacktracking רגיל בנוסף ל3arc. אנו מאמינים כי הסיבה לכך הינה שכמות האילוצים בבעיה גדולה, ולכן גם הבחירה של הערכים שלאחר שיבוצם מספר ההגבלות הוא נמוך ביותר עדיין מספר ההגבלות שנוצר יהיה גבוה.

:Degree Heuristic and Least Constraining Value

שילוב בין שתי האוריסטיקות לעיל, כלומר נבחר גם את המשתנה שיש לו הכי הרבה אילוצים וגם את הערך שמוביל לכך שלאחר שיבוצו מספר ההגבלות על המשתנים יהיה מינימאלי. באופן כללי החלטנו לשלב בין שתי האוריסטיקות הנ"ל מכיוון ש Degree Heuristic נתנה לנו את התוצאות הטובות ביותר עבור יוריסטיקה על המשתנים ו Least Constraining Value הינה האוריסטיקה היחידה שלנו על הערכים ומכאן הנחנו ששילוב בין שתיהן יניב תוצאה טובה יותר. בפועל גם האוריסטיקה הזאת לא עבדה טוב יותר מbacktracking רגיל בנוסף ל3arc.

מסקנות:

לאחר שניסינו מספר יוריסטיקות, ולא ראינו שיפור ניכר בתוצאות (לא הצלחנו לשבץ יותר מ20 מבחנים, 10 מועדי א ו 10 מעודי ב), הגענו למסקנה כי השיטה של Pure CSP אינה מתאימה לפתרון הבעיה של שיבוץ מבחנים. אנו סבורים שהסיבה הינה מרחב החיפוש הגדול ומספר האילוצים שנוצרים כתוצאה מכך. בנוסף יתכן כי ישנן יוריסטיקות שונות שיכולות לפתור את הבעיה בצורה זו, אך העדפנו לנסות גישה שונה לפתרון הבעיה. הדרך הבאה שאותה ניסינו הייתה ע"י מידול הבעיה כבעיית Weighted Constraint Satisfaction Problem.

3. Weighted Constraint Satisfaction Problem

הגדרת הבעיה:

הבעיה זו היא הכללה של הבעיה הקודמת (מסעיף 1) כך שחלק מהאילוצים יכולים להיות מופרים/ כלומר, חלוקה של האילוצים לאילוצים "קשים" ואילוצים "רכים". בבעיה הנ"ל, האילוצים הקשים הנם האילוצים שהוצגו בסעיף הקודם וכן התווסף אילוץ רך שהוגדר בצורה הבאה:

(v_i, t_j) for i in n s.t t_j is an assignment of v_i and v_i is a math course $\rightarrow t_j = 9:00$

כלומר, עבור כל שיבוץ של מבחן לזמן מסויים, כך שהמבחן שייך לפקולטת מתמטיקה, נרצה כי המבחן יתקיים ב-9 בבוקר. לאחר מכן, הוגדרה פונקציית ערך על האילוצים. כלומר, לכל הפרה של אילוץ מסויים ישנה עלות. העלות עבור הפרה של אילוץ קשה הנה אינסוף (מאחר ואנו לא נאפשר בשום אופן הפרה שלו) ועבור הפרה של אילוץ רך יוחזר ערך ממשי כלשהי שמתאר את חומרת ההפרה. מאחר ובבעיה הנ"ל ישנו אילוץ רך אחד בלבד, יוחזר ערך קבוע.

מימוש האלגוריתם:

branch and bound:

את הבעיה הנ"ל ניסינו לפתור בעזרת אלגוריתם *branch and bound*. אלגוריתם זה מהווה מעין וריאציה של *backtracking* אך עם כמה שיפורים מיוחדים. תחילה, במידה ובעת חיפוש הפתרון ישנה הפרה של אילוץ קשה כלשהו, נעצור את החיפוש בענף הנוכחי ונחזור אחורה. לאחר מכן, במידה ואנו מגיעים לפתרון של הבעיה כך שאין בידינו פתרון נוסף, נשמור אותו יחד עם הערך שלו המתקבל כתוצאה מהפרה של האילוצים הרכים. כעת, נמשיך בחיפוש ובכל פעם שנגיע לענף שהערך שלו גבוה מהערך השמור, נפסיק את המשך החיפוש בענף זה. כמו כן, במידה ונגיע לפתרון טוב יותר מהפתרון הנוכחי, נעדכן ונשמור את הפתרון הטוב ביותר שהושג. לבסוף נחזיר את הפתרון שמתקבל.

מסקנות:

שיטה זו לא עבדה כמו שצריך גם כן, אנו משערים שהסיבה לכך היא מאחר ומרחב החיפוש הנו רחב מאוד ולא הצלחנו לצמצם אותו באופן משמעותי. לאחר חוסר ההצלחה בשיטה הזו, ניסינו למדל את הבעיה בצורה שונה לגמרי והיא מידול הבעיה בבעיית *informed search*.

1. Simulated Annealing

הגדרת הבעיה:

הגדרנו מצב (state) בתור שיבוץ חוקי של לוח בחינות, כאשר שיבוץ חוקי מוגדר להיות שיבוץ העומד בכול האילוצים הקשים.

בנוסף הגדרנו פונקציית ערך המודדת את טיב הפתרון בהתאם לעמידתו באילוצים הרכים.

פתרון חוקי הינו פתרון העומד באילוצים הקשים אשר מוגדרים באופן הבא בבעיה זו

Hard Constraints:

- $(v_i,)$ for i in n s.t $\forall i \exists j$ s.t $v_i: t_j$ and $\forall i \forall j v_i: t_j \rightarrow \forall k \neq j \neg(v_i: t_k)$
- (v_i, v_j) for i, j in n s.t $\forall i, j \forall k v_i: t_k$ and v_j are on the same faculty $\rightarrow \neg(v_j: t_k)$
- (v_i, v_j) for i, j in n s.t v_i is the first attempt of $v_j \rightarrow t_{v_j} - t_{v_i} \geq 14$
- $(t_i,)$ for i in m s.t num of students in $t_i \leq \text{maximum amount of students}$

Soft Constraints:

- (v_i, v_j) for i, j in n s.t v_i and v_j are on the the same faculty $\rightarrow \text{abs}(t_{v_j} - t_{v_i}) < d_{f_k}$
- $(t_i,)$ for i in m s.t t_i is an evening slot $\rightarrow \forall j \neg(v_j: t_i)$

כך שח הוא מספר משבצות הזמן החוקיות בבעיה (הדומיין ההתחלתי המוגדר בבעיית CSP)

האילוצים הקשים במילים:

- כל מבחן משובץ במשבצת זמן אחת בלבד
- לא קיימים שני מבחנים באותה פקולטה באותה משבצת זמן
- ישנו הפרש של לפחות שבועיים בין מועד א למועד ב באותו הקורס
- מספר הסטודנטים המשובצים באותה משבצת זמן קטן או שווה למספר הסטודנטים המקסימאלי (המספר הוגדר להיות כגודל הקורס הגדול ביותר באותו הסמסטר).

האילוצים הרכים במילים:

- מבחנים מאותה פקולטה משובצים במרחק גדול או שווה להפרש המינימאלי שהוחלט עבור אותה פקולטה.
- אין מבחנים המשובצים בשעות הערב.

מימוש האלגוריתם:

תחילה מתוך קובץ csv נתון חילצנו את הקורסים, לאחר מכן בנינו את "הדומיין" כלומר את כל משבצות הזמן החוקיות עבור בעיה זו (כמו שמוגדר בבעיית CSP).

אנו בונים מצב התחלתי ולאחר מכן אנו רצים בלולאה למספר איטרציות המתקבל בבנאי ובכול איטרציה אנו פועלים באופן הבא: בוחרים שכן אקראי של המצב הנוכחי, בודקים האם הערך שלו נמוך מהערך של השכן הנוכחי, במידה וכן נתקדם לשכן הנבחר אחרת נתקדם לשכן זה בהסתברות הנקבעת על ידי טמפרטורה ופונקציית קירור. בסוף האלגוריתם נחזיר את המצב האחרון שהתקבל.

פונקציית קירור *Cooling Function*:

פונקציית הקירור שבחרנו פועלת באופן הבא:

$$\text{temp} = \text{initial temp} \cdot \alpha^t$$

כאשר α הוא קבוע כלשהו $0 < \alpha < 1$ הינו מספר האיטרציה.

בכול איטרציה אנו מעדכנים את הטמפרטורה לפי פונקציית הקירור הנ"ל.

חישוב ההסתברות לצעד "לא טוב":

חישוב ההסתברות למתי נעשה צעד "לא טוב" נעשית באופן הבא

$$p < e^{-\Delta \text{temp}_t}$$

כאשר p הוא משתנה מקרי שמתפלג אחיד בין 0 ל 1

Δ היא ההפרש בין המצב הנוכחי למצב הקודם

ו temp_t הינה הטמפרטורה באיטרציה t .

ניתן להבחין כי ככל שמספר האיטרציות עולה כך הסבירות שנעשה צעד "רע" יורדת.

בחירת שכן באקראי:

בחירת שכן באקראי נעשית בשלושה אופנים:

1. בחירה אונארית – צעד שמגריל האם לזוז קדימה או אחורה. אם נבחר קדימה ננסה לשבץ את המבחן במשבצת זמן שבטווח בין משבצת הזמן שאחרי משבצת הזמן שהמבחן שובץ בה לבין משבצת הזמן שהמבחן שובץ בה ועוד מספר משבצות זמן קבוע. אם נבחר אחורה אז השינוי זהה עד כדי הכפלה במינוס 1 בכיוון התנועה. במידה והשיבוץ תקין נחזיר אותו אחרת נמשיך למשבצת הזמן הבאה אם ניסנו את כל משבצות הזמן ואף אחת מהן אינה חוקית נחזיר את המצב עצמו.
2. בחירה בינארית – צעד שבוחר באקראי מבחן נוסף שאינו המבחן עצמו או מועד ב שלו ומחליף את השיבוצים שלהם. במידה וההחלפה חוקית נחזיר אותה אחרת נחזיר את הילד עצמו ללא מוטציה.
3. בחירה אקראית – בחירה שבהסתברות מסוימת בוחרת בחירה אונארית ובהסתברות מסוימת בוחרת בחירה בינארית.

אנו מגדירים מצב טוב יותר לפי פונקציית הערך שמתקבלת באופן הבא: הוספת המקסימום בין 0 להפרש בין מספר הימים הרצוי למספר הימים בשיבוץ הקיים של קורסים מאותה פקולטה, כאשר קנסנו הפרשים שגדולים מהפרש שראינו כמינימאלי ההגיוני. בנוסף הוספנו את כמות המבחנים ששובצו בערב כפול קבוע.

מסקנות:

בשיטה זו הצלחנו להגיע לתוצאות טובות יותר מאשר עם *pure CSP* אך עדיין בהשוואה לאלגוריתמי החיפוש האחרים אנו משיגים ערכים גבוהים יותר לפי פונקציית הערך שקבענו סיבה אפשרית לכך היא האקראיות הרבה באלגוריתם. דבר נוסף שיכול להוות בעיה הינו שפונקציית הקירור שבחרנו דועכת בקצב מהיר יחסית ולכן האלגוריתם לא מצליח לברוח ממינימום מקומי, נציין כי תופעה דומה מתרחשת בכול פונקציות הקירור שניסינו.

הגדרת הבעיה זהה ל-Simulated Annealing

מימוש האלגוריתם:

תחילה מתוך קובץ csv נתון חילצנו את הקורסים, לאחר מכן בנינו את "הדומיין" כלומר את כל משבצות הזמן החוקיות עבור בעיה זו (כמו שמוגדר בבעיית CSP).

במימוש שלנו בחרנו את הגרסה של GD אשר עוצרת או בהתכנסות (כלומר הערך הטוב ביותר שמצאנו לא מביא לשיפור) או כשהגענו למספר איטרציות מקסימאלי. כשאר אנו מתחילים משיבוץ חוקי רנדומי.

בכול איטרציה אנו זזים בכיוון הצעד המקומי הנמוך ביותר, כאשר צעד מקומי מוגדר להיות שינוי שיבוץ מסגרת הזמן עבור קורס יחיד. כלומר בכול איטרציה נעבור על כל אפשרות של שינוי תאריך של כל אחד מהקורסים בהינתן השיבוץ הקודם ונבחר את השינוי שיוביל לערך הנמוך ביותר בפונקציית הערך שלנו (זהה לזאת שמתוארת ב-SA).

הגדרת שכן:

עבור אלגוריתם זה שכן מוגדר להיות כל מצב שבו משנים שיבוץ של קורס אחד. כפי שמתואר באלגוריתם בכול שלב אנו בוחרים את השינוי של הקורס למשבצת זמן שנותן את הערך הנמוך ביותר.

מסקנות:

ניתן היה לראות כי בהרצות שונות התקבלו ערכים שונים, ללא תלות במספר האיטרציות (הערכים השונים הגיעו מנקודות התכנסות שונות) ומכיוון שהדבר הרנדומי היחיד באלגוריתם זה הוא נקודת ההתחלה ניתן להסיק כי האלגוריתם נתקע במינימום מקומי ולכן ככל הנראה אלגוריתם סוכסטי כמו GAI ו-RGD יוכל לעזור לנו לקבל תוצאות טובות יותר.

הגדרת הבעיה זהה ל-Gradient Descent

מימוש האלגוריתם:

מימוש האלגוריתם דומה מאוד לזה של GD. ההבדל המרכזי במימוש האלגוריתם הוא שב-SA אנו מאפשרים צעד אקראי (לא הצעד המקומי הכי טוב) במקרים הבאים

1. בהסתברות הנקבעת מראש ומהווה את Learning Rate של האלגוריתם.
2. כאשר האלגוריתם מגיע למינימום (כלומר כל שכן של המצב הנוכחי יותר גדול ממנו).

הצעד האקראי במקרה הראשון הינו מעבר לשכן אקראי של המצב, ואילו הצעד במקרה השני הינו שינוי המצב לנקודה התחלתית שונה וחזרה על התהליך. בנוסף הגדרת השכן לזו של GD והצעדים האקראיים זהים לאלו של SA.

מסקנות:

על אף שנראה כי GD הגיע למינימום מקומי לא היה שינוי משמעותי בתוצאות בין שני האלגוריתמים אנו סבורים כי במקרה זה אין יתרון בשימוש באלגוריתם זה שכן זמן הריצה שלו ארוך יותר והוא אינו מניב תוצאות טובות יותר.

הגדרת הבעיה לזה Gradient Descent**מימוש האלגוריתם:**

כעיבוד מקדים חילצנו מתוך קובץ csv נתון את הקורסים, לאחר מכן בנינו את "הדומיין" כלומר את כל משבצות הזמן החוקיות עבור בעיה זו (כמו שמוגדר בבעיית CSP).

בתחילת האלגוריתם הגרלנו "דור" שלם רנדומלי בגודל שנקבע מראש שהינו גודל האוכלוסייה. כל פרט בדור מתאר שיבוץ זמנים חוקי בהינתן שיבוץ הזמנים שניתן. לכול מבחן ניתן אינדקס מסוים, ואינדקס זה נשמר לכול השיבוצים, כלומר למבחנים יש סדר קבוע בכול השיבוצים. בכל דור שבא אחריו אנו יוצרים אוכלוסייה חדשה על ידי רבייה של הדור הנוכחי תוך כדי אפשרות למוטציות. תהליך יצירת הדור מתרחש כך: רצים בלולאה עד שכמות הילדים בדור החדש היא ככמות הילדים הרצויה. בכול איטרציה של הלולאה אנו בוחרים באקראי שני הורים ולאחר מכן בהסתברות מסויימת אנו בוחרים אם הם יתרבו או שנבחר את ההורה הראשון (מיכון שבחירת הזוגות היא אקראית סברנו כי אין צורך לבחור באקראי גם את ההורה שאנו מכניסים). אם בחרנו שזוג ההורים יתרבה אז בהסתברות מסוימת נבחר אם לילד שיצא תהיה מוטציה או לא.

רבייה *reproduce*:

ההתרבות נעשית באמצעות crossover כלומר אנו בוחרים מקום רנדומי בשיבוץ (מבחן מסוים) והמבחן הראשון ועד אליו אנו לוקחים את המידע מההורה הראשון וממנו ועד למבחן האחרון אנו לוקחים את השיבוץ של ההורה השני. לאחר ההכלאה במידה ושיבוץ הזמנים הוא תקין אנו מחזירים אותו במידה ולא אנו מנסים שוב, הפעם עם נקודת החלפה שונה. אם זיווג נכשל לאחר מספר קבוע מראש של ניסיונות, הזיווג נכשל ולא ייוצר ילד באיטרציה זו.

מוטציה *mutation*:

- מוטציה מתקיימת בהסתברות מסוימת שנקבעה מראש. בכול פעם שאנו עושים מוטציה אנו מגרילים בין רבע לחצי מהמבחנים בשיבוץ ועל כל מבחן אנו בוחרים באקראי איזה מבין המוטציות הבאות לבצע:
- מוטציה אונארית – מוטציה שבוחרת באקראי כמה קורסים לשנות ולאחר מכן מגרילה האם לזוז קדימה או אחורה. אם נבחר קדימה היא תנסה לשבץ את המבחן במשבצת זמן שבטווח בין משבצת הזמן שאחרי משבצת הזמן שהמבחן שובץ בה לבין משבצת הזמן שהמבחן שובץ בה ועוד מספר משבצות זמן קבוע. אם נבחר אחורה אז השינוי זהה עד כדי הכפלה במינוס 1 בכיוון התנועה. במידה והשיבוץ נחזיר אותו אחרת נמשיך למשבצת הזמן הבאה אם ניסנו את כל משבצות הזמן ואף אחת מהן אינה חוקית נחזיר את הילד עצמו ללא מוטציה.
 - מוטציה בינארית – מוטציה שבוחרת באקראי מבחן נוסף שאינו המבחן עצמו או מועד ב שלו ומחליפה את השיבוצים שלהם. במידה וההחלפה חוקית נחזיר אותה אחרת נחזיר את הילד עצמו ללא מוטציה.

האלגוריתם ימשיך עד שנגיע למספר דורות קבוע מראש (מתקבל בבנאי) כאשר בכול דור הוא שומר את הפריט בעל הערך הנמוך ביותר באוכלוסייה, כשאר מדידה של ערך של פריט זהה למדידה של מצב ב GD

מסקנות:

ניתן לראות כי אלגוריתם זה משיג בדרך כלל תוצאות טובות יותר מGD בזמן קצר יותר הסיבה לכך היא ככל הנראה הרנדומליות הרבה באלגוריתם המאפשרת "לברוח" ממינימום מקומי ולהגיע לתוצאות טובות יותר. לגבי זמן הריצה רוב הצעדים של האלגוריתם הגנטי הם מקומיים ונבחרים באקראי ולכן אין צורך לעבור על כל הצעדים האפשריים כדי למצוא את הפתרון הטוב ביותר ומכאן שזמן הריצה קטן בצורה משמעותית.

מימוש האלגוריתמים עבור פתרון בעיית שיבוץ האולמות

1. Gradient Descent

הגדרת הבעיה:

הגדרנו מצב (state) בתור שיבוץ חוקי של לוח בחינות, כאשר שיבוץ חוקי מוגדר להיות שיבוץ העומד בכול האילוצים הקשים.

בנוסף הגדרנו פונקציית ערך המודדת את טיב הפתרון בהתאם לעמידתו באילוצים הרכים. פתרון חוקי הינו פתרון העומד באילוצים הקשים אשר מוגדרים באופן הבא בבעיה זו

Hard Constraints:

- $(v_i,)$ for i in n s.t $s.t \{v_i: s_j\}$ and $\{v_i: t_m\} \rightarrow \forall h \in s_j \forall k \neg(h \text{ is a hall of } v_k) \text{ or } \neg(\{v_k: t_m\})$
- $(v_i,)$ for i in n s.t $\{v_i: s_j\} \rightarrow \sum_{h \in s_j} (capacity(h)) \geq students(v_i)$
- $(v_i,)$ for i in n s.t $\{v_i: s_j\} \rightarrow \forall h \in s_j type(h) = type(v_i)$

Soft Constraints:

- $(v_i,)$ for i in n s.t $\{v_i: s_j\} \rightarrow \forall h_1, h_2 \in s_j chair_type(h_1) = chair_type(h_2)$
- $(v_i,)$ for i in n s.t $\{v_i: s_j\} \rightarrow \forall h_1 \in s_j type(h_1) = regular$
- $(v_i,)$ for i in n s.t $\{v_i: s_j\} \rightarrow \forall h_1, h_2 \in s_j area(h_1) = area(h_2)$
- $(v_i,)$ for i in n s.t $\{v_i: s_j\} \rightarrow \sum (capacity(s_j)) \leq 1.25 * students(v_i)$

כך ש s_j הינה תת קבוצה של אולמות, n הוא מספר הקורסים ו h_j מתייחס לאולם ה j .

האילוצים הקשים במילים:

- לכול אולם משובץ מבחן אחד בלבד למשבצת זמן ספציפית.
- הקיבולת של האולמות שמשובצים למבחן גדולה או שווה לכמות הנבחנים
- לכול מבחן משובצים אולמות המתאימים לסוג המבחן

האילוצים הרכים במילים:

- כל האולמות שמשובצים עבור מבחן מסוים הם בעלי אותו סוג כיסאות
- כל האולמות המשובצים הם בעלי כיסאות רגילים
- כל האולמות שמשובצים למבחן מסוים נמצאים באותו אזור
- הקיבולת של האולמות שמשובצים למבחן מסוים לא עולה על 1.25 כפול מספר הנבחנים

מימוש האלגוריתם:

תחילה מתוך קובץ csv נתון חילצנו את האולמות, ולאחר מכן שלחנו את האולמות ושיבוץ אולמות למשבצות זמן תקין לאלגוריתם.

במימוש שלנו בחרנו את הגרסה של GD אשר עוצרת או בהתכנסות (כלומר הערך הטוב ביותר שמצאנו לא מביא לשיפור) או כשהגענו למספר איטרציות מקסימאלי. כשאר אנו מתחילים משיבוץ חוקי רנדומי.

בכול איטרציה אנו זזים בכיוון הצעד המקומי הנמוך ביותר, כאשר צעד מקומי מוגדר להיות שינוי שיבוץ אולם עבור קורס יחיד. כלומר בכול איטרציה נעבור על כל אפשרות של שינוי אולם של כל אחד מהקורסים בהינתן השיבוץ הקודם ונבחר את השינוי שיוביל לערך הנמוך ביותר בפונקציית הערך שלנו (זהה לזאת שמתוארת בSA).

הגדרת שכן:

עבור אלגוריתם זה שכן מוגדר להיות כל מצב שבו משנים שיבוץ של קורס אחד. כפי שמתואר באלגוריתם בכול שלב אנו בוחרים את השינוי של אולם בתוך שיבוץ האולמות של קורס מסוים שנותן את הערך הנמוך ביותר.

פונקציית הערך לבחירת שכן עם ערך נמוך יותר מוגדרת באופן הבא: סכום היחסים בין מספר האולמות עם סוג הכיסאות שמופיע הכי מעט בשיבוץ לכלל האולמות של כל מבחן בנוסף לחלק היחסי של אולמות עם כיסאות סטודנט מכלל האולמות ששובצו לכלל הבחינות ועוד סכום של סכום המרחקים מהחציון של כלל האולמות ששובצו למבחן מסוים עבור כלל המבחנים כפול קבוע. ועוד מספר המבחנים שבהם יש קיבולת של פי קבוע גדול מ1 ממספר הסטודנטים שנבחנו במבחן.

מסקנות:

ניתן היה לראות כי האלגוריתם עוצר לאחר מספר איטרציות קטן ממספר האיטרציות שניתן בבנאי ומכאן אנו חושדים כי האלגוריתם נתקע במינימום מקומי ולכן ככל הנראה אלגוריתם סוכסטי כמו GA יוכל לעזור לנו לקבל תוצאות טובות יותר.

2. Genetic Algorithm **זהה ל'Gradient Descent'**

מימוש האלגוריתם:

כעיבוד מקדים חילצנו מתוך קובץ csv נתון את האולמות, בנוסף שמרנו את פתרון הבעיה הקודמת, שהינו שיבוץ חוקי של מבחנים למשבצות זמן (כלומר בעיה זו הינה שיבוץ אולמות למבחנים בהינתן שיבוץ של מבחנים למשבצות זמן).

בתחילת האלגוריתם הגרלנו "דור" שלם רנדומלי בגודל שנקבע מראש שהינו גודל האוכלוסייה. כל פרט בדור מתאר שיבוץ אולמות חוקי בהינתן שיבוץ הזמנים שניתן. לכול מבחן ניתן אינדקס מסוים, ואינדוקס זה נשמר לכול השיבוצים, כלומר למבחנים יש סדר קבוע בכול השיבוצים. בכל דור שבא אחריו אנו יוצרים אוכלוסייה חדשה על ידי רבייה של הדור הנוכחי תוך כדי אפשרות למוטציות. תהליך יצירת הדור מתרחש כך: רצים בלולאה עד שכמות הילדים בדור החדש היא ככמות הילדים הרצויה. בכול איטרציה של הלולאה אנו בוחרים באקראי שני הורים ולאחר מכן בהסתברות מסויימת אנו בוחרים אם הם יתרבו או שנבחר את ההורה הראשון (מיכון שבחירת הזוגות היא אקראית סברנו כי אין צורך לבחור באקראי גם את ההורה שאנו מכניסים). אם בחרנו שזוג ההורים יתרבה אז בהסתברות מסוימת נבחר אם לילד שיצא תהיה מוטציה או לא.

רבייה *reproduce*:

ההתרבות נעשית באמצעות crossover כלומר אנו בוחרים מקום רנדומי בשיבוץ (מבחן מסוים) והמבחן הראשון ועד אליו אנו לוקחים את המידע מההורה הראשון וממנו ועד למבחן האחרון אנו לוקחים את השיבוץ של ההורה השני. לאחר ההכלאה במידה ושיבוץ האולמות הוא תקין אנו

מחזירים אותו במידה ולא אנו מנסים שוב, הפעם עם נקודת החלפה שונה. אם זיווג נכשל לאחר מספר קבוע מראש של ניסיונות, הזיווג נכשל ולא יוצר ילד באיטרציה זו.

מוטציה mutation:

מוטציה מתקיימת בהסתברות מסוימת שנקבעה מראש. בכל פעם שאנו עושים מוטציה אנו מגרילים בין חצי מהמבחנים לכלל המבחנים בשיבוץ ועל כל מבחן אנו בוחרים באקראי איזה מבין המוטציות הבאות לבצע:

- א. מוטציה אונארית – מוטציה שבוחרת באקראי כמה אולמות לשנות בשיבוץ הבחירה הינה בין שליש לחצי מהאולמות. לאחר מכן אנו מסדרים באקראי את כל האולמות שאינם משובצים במשבצת הזמן של המבחן הנוכחי ונוסיף מהם אולמות עד שנגיע לשיבוץ חוקי למבחן. במידה ולא הצלחנו להגיע לשיבוץ חוקי לאחר שעברנו על כל האולמות הפנויים נחזור על התהליך. המוטציה תסתיים כאשר הגענו לשיבוץ חוקי או לאחר מספר ניסיונות קבוע מראש.
- ב. מוטציה בינארית – מוטציה שבוחרת באקראי מבחן נוסף מאותו הסוג (ממוחשב או בכתב) לאחר מכן אנו בוחרים כמות של אולמות להחליף, מספר בין 0 לכמות האולמות של המבחן עם כמות האולמות הקטנה יותר, ולבסוף אנו בוחרים באקראי אולמות משני הקורסים כמספר האולמות שבחרנו מקודם ומחליפים בניהם. במידה ולאחר ההחלפה הפתרון תקין נחזיר אותו אחרת נחזור על התהליך. במידה ולאחר מספר קבוע של פעמים לא הגענו לפתרון חוקי המוטציה תיכשל.

בכול מקרה בו המוטציה נכשלה (לא הגענו לפתרון חוקי) נחזיר את הצאצא של ההתרבות ללא המוטציה.

האלגוריתם ימשיך עד שנגיע למספר דורות קבוע מראש (מתקבל בבנאי) כאשר בכול דור הוא שומר את הפריט בעל הערך הנמוך ביותר באוכלוסייה, כשאר מדידה של ערך של פריט זהה למדידה של מצב ב GD

מסקנות:

ניתן לראות כי אלגוריתם זה משיג בדרך כלל תוצאות טובות יותר מGD בזמן קצר יותר הסיבה לכך היא ככל הנראה הרנדומליות הרבה באלגוריתם המאפשרת "לברוח" ממינימום מקומי ולהגיע לתוצאות טובות יותר. לגבי זמן הריצה רוב הצעדים של האלגוריתם הגנטי הם מקומיים ונבחרים באקראי ולכן אין צורך לעבור על כל הצעדים האפשריים כדי למצוא את הפתרון הטוב ביותר ומכאן שזמן הריצה קטן בצורה משמעותית.

מימוש האלגוריתמים עבור יעוץ לסטודנטים על ההחלטה לגבי מועד ב:

1. ID3

הגדרת הבעיה:

אספנו מידע מסטודנטים מפקולטות שונות לפי השאלות הבאות:

- האם נכשלת במועד א? (כן, לא)
 - מה היה ציון הבחינה במועד א? (פחות מ55, 55-69, 60-69, 70-79, 80-89, 90-100)
 - מה היה הציון הסופי בקורס? (פחות מ60, 60-69, 70-79, 80-89, 90-100)
 - האם עשית מועד ב בעבר? (כן, לא)
 - האם עשית מועד ב בעבר ושיפרת את ציון מועד א? (כן, לא)
 - האם אתה בטוח שתוכל לשפר את ציון הבחינה? (כן, לא, לא יודע)
 - כמה ימים יש לך ללמוד לבחינה? (1, 2, 3, 4, 5, 6, יותר מ7)
 - איך היית מגדיר את קושי הבחינה? (קל, בינוני, קשה, קשה מאוד)
 - מה סוג הבחינה? (פתוח, אמריקאי, שילוב של השניים)
 - מה היא צורת ההיבחנות? (ממוחשב, בכתב)
 - האם ניתן פקטור במועד א? (כן, לא)
 - האם בדרך כלל מועדי הא קשים יותר ממועדי הב במקצוע זה? (כן, לא, לא יודעת)
- כאשר השאלה הסופית הינה label שאנו רוצים לחזות והיא האם הסטודנט ניגש בסוף למועד הב.

העץ מוגדר באופן הבא: כל קודקוד מלבד העלים הוא שאלה והעלים הינם תשובה על האם כדאי להיבחן במועד ב או לא.

מימוש האלגוריתם:

תחילה אספנו מידע מסטודנטים באמצעות סקר, לאחר מכן מכיוון שהסקר בעברית ולנו היה נוח יותר לעבוד בתוכנה באנגלית תרגמנו את המידע לאנגלית באמצעות שימוש בספרייה pandas של python.

לאחר עיבוד המידע יצרנו את העץ. תהליך יצירת העץ נעשה באמצעות ID3 באופן הבא:

1. תחילה חישבנו את Information Gain של כל שאלה עשינו זאת על ידי חישוב הentropy של הטבלה מחיסור הentropy של כל שאלה בטבלה.
2. לאחר מכן בחרנו את השאלה שלה יש את Information Gain המקסימלי וממנה החלטנו לפצל את העץ לענפים לפי התשובות, כלומר נצמצם את המידע שלנו בכול ענף רק לאנשים שענו את התשובה המתאימה לענף (למשל אם נבחרה השאלה מה סוג הבחינה? אז נפצל את העץ לשלושה ענפים כך שבענף הראשון יהיו רק תשובות בהן הבחינה היא פתוחה בשני תשובות בהן הבחינה אמריקאית ובשלישי תשובות בהן הבחינה היא שילוב של השניים).
3. עבור כל ענף נחזור על התהליך כאשר נסיים לחקור ענף מסוים כשהוא מסתיים בתשובה ולא בשאלה.
4. התהליך יסתיים כאשר כל הענפים מסתיימים בתשובה.

$$\begin{aligned}\text{Entropy} - H(S) &= \left(-\frac{n}{n+p} \cdot \log_2 \frac{n}{n+p}\right) + \left(-\frac{p}{n+p} \cdot \log_2 \frac{p}{n+p}\right) \\ \text{Attribute Entropy} - H(S|A) &= \sum_{i \in [m]} \left(-\frac{n_{c_i}}{n_{c_i}+p_{c_i}} \cdot \log_2 \frac{n_{c_i}}{n_{c_i}+p_{c_i}}\right) + \left(-\frac{p_{c_i}}{n_{c_i}+p_{c_i}} \cdot \log_2 \frac{p_{c_i}}{n_{c_i}+p_{c_i}}\right) \\ \text{Information Gain} - IG(S, A) &= (H(S) - H(S|A))\end{aligned}$$

כאשר n מתייחס לכול המשיבים שבחרו לא להיבחן ו p מתייחס לכול הנבחנים שבחרו להיבחן בנוסף m הן מספר התשובות של השאלה A למשל עבור כמה ימים יש ללמוד לבחינה m יהיה 7 ו c_i הינה התשובה i של השאלה למשל c_1 עבור השאלה הקודמת יהיה כל המשיבים שענו 1. ו n_{c_i}, p_{c_i} הם

כלל המשיבים שענו את התשובה ה- c_i ולא לקחו את הבחינה וכלל המשיבים שענו את התשובה ה- c_i ולקחו את הבחינה בהתאמה.

לאחר יצירת העץ אנו עוברים לשלב היעוץ בשלב זה אנו שואלים את הסטודנט את כל השאלות שאספנו ואז לפי העץ שנוצר בשלב הקודם אנו נחשוף בפניו את ההחלטה שהאלגוריתם הגיע אליה. במידה ואין בידנו מידע מספק כדי לתת תשובה (אף אחד לא השיב כמוהו בסקר) נעודד את הנשאל למלא את הסקר ולהרחיב את המידע שלנו.

פתרון שיבוץ המבחנים למשבצות זמן באמצעות תורת המשחקים

הגדרת הבעיה:

שיטה נוספת שבחרנו לפתרון הבעיה היא באמצעות תורת המשחקים. כאשר לצורך מידול הבעיה הגדרנו את המשחק הבא: השחקנים הם החוגים השונים (מתמטיקה, מדעי המחשב, ביולוגיה וכו'...), לוח המשחק הוא הימים של תקופת המבחנים כאשר 1 מציין את היום הראשון 2 את השני וכן הלאה.

המשחק מתנהל באופן הבא: בכול פעם החוג הגדול ביותר בוחר בתורו יום בו הוא ישבץ מבחן אחד מרשימת המבחנים שלו באותו סמסטר (נשים לב כי בדומה להגדרה בבעיית החיפוש אנו מתייחסים לקורסים של שנה ספציפית וסמסטר ספציפי למשל שנה ב סמסטר א).

הבחירה של איזה קורס לשבץ נעשית לפי פונקציית הערך הבאה:

$$V(\{x_0, \dots, x_j\}) = \min_{0 \leq i \leq j-1} \frac{1}{z_{i+1} + z_i} (x_{i+1} - x_i)$$

כאשר x_i הינו שיבוץ של המבחן ה- i , j הינו מספר המבחנים ועוד 1 ו- z_i הינו כמות נקודות הזכות של הקורס ה- i

כחלק מהגדרת הבעיה בחרנו לעסוק בקואליציות, שהם תת קבוצות של חוגים כך שחיתוך הקואליציות הינו כלל החוגים ואין חוג שנמצא ביותר מקואליציה אחת. הסיבה שעסקנו בקואליציות הינה כדי לבדוק האם לחוגים קטנים יותר יש יתרון ביצירת קואליציות כדי לשפר את לוח הבחינות שלהם. דוגמא אפשרית לכך הינו משחק בו משתתפים החוגים מדעי המחשב עם 40 סטודנטים, מתמטיקה עם 35 סטודנטים, כימיה עם 25 סטודנטים וביולוגיה עם 20 סטודנטים. נבחין כי במשחק בלי קואליציות ביולוגיה תמיד יבחרו אחרונים ואילו אם ביולוגיה וכימיה יקימו קואליציה הם יוכלו לבחור שיניים.

הגדרה פורמאלית של הבעיה ודוגמא מפורטת להקמת קואליציות ניתן לראות בקובץ המצורף על תורת הקבוצות.

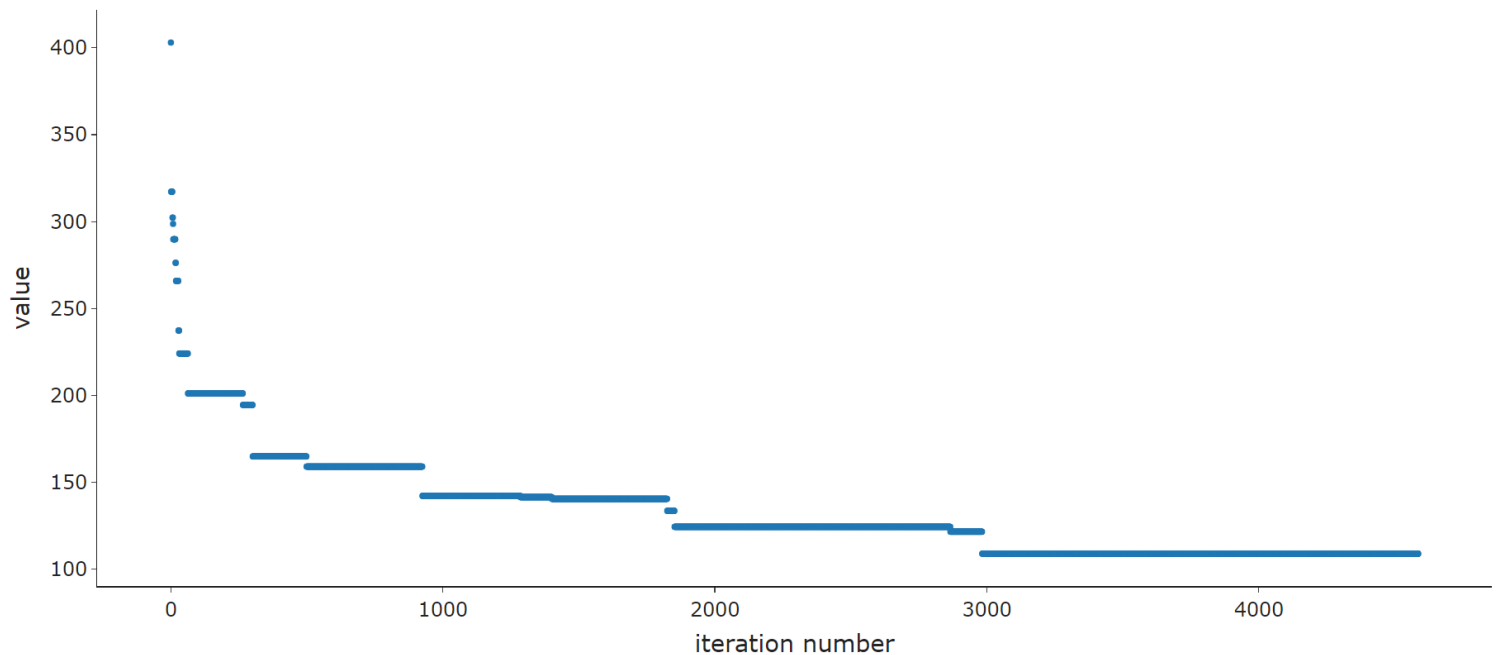
השוואה בין האלגוריתמים וניתוח תוצאות:

השוואה בין האלגוריתמים בעיית שיבוץ מבחנים למשבצות זמן:

גרפים ותוצאות:

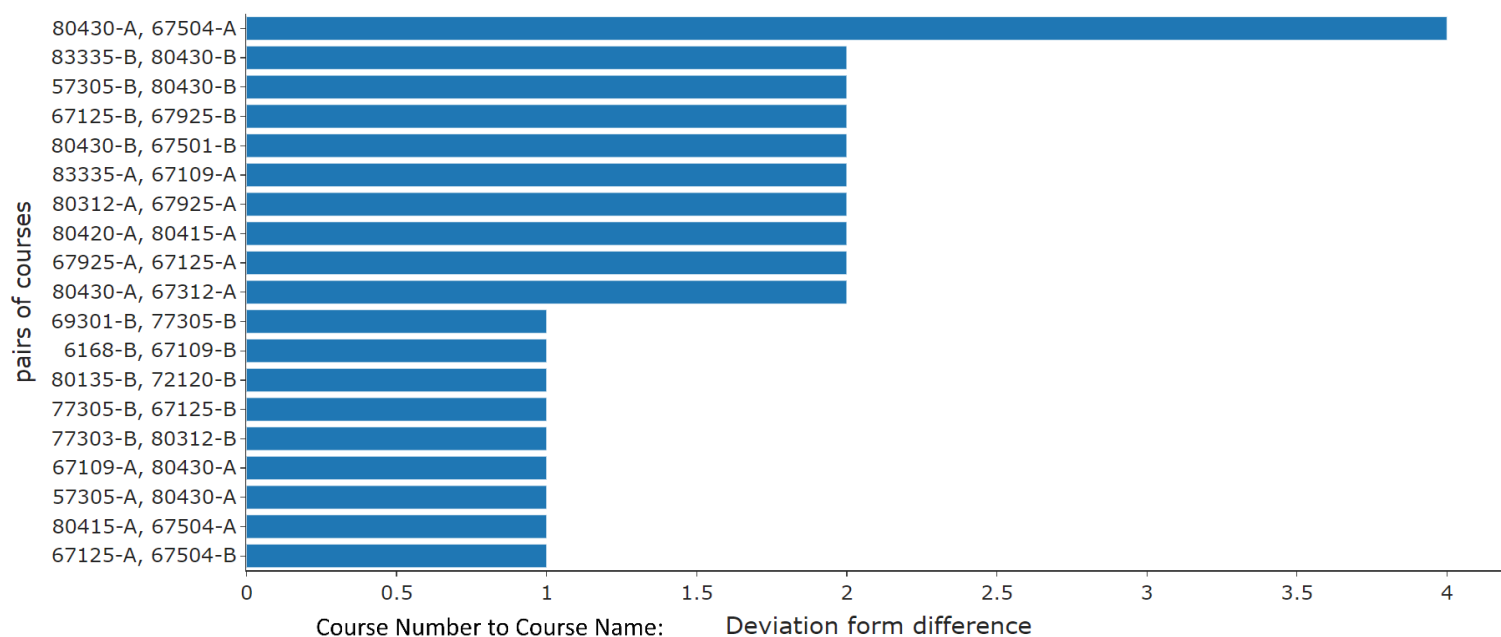
כל הגרפים הבאים מוצגים על פתרון בעיית שיבוץ האולמות למשבצות זמן עבור שנה ב סמסטר א בין התאריכים 15.1.2022-8.3.2022 נציין כי זו הייתה תקופת המבחנים של שנה ב סמסטר א עבורנו ולכן זוהי תקופת זמן הגיונית לשיבוץ הבחינות. בנוסף לקחנו את המידע של הקורסים לפי השנתון
:Simulated Annealing

SA Values as a Function of Iteration Number



בגרף זה ניתן לראות את ערך פונקציית הערך שלנו (זו שמוגדרת בעמוד 8) כאשר הקבוע שאנו מכפילים במבחני הערב הוא 0.75 ובנוסף ה"קנס" שאנו נותנים על מבחנים שההפרש בניהם קטן בשלושה ימים או יותר מההפרש הרצוי בניהם הינו כפל ב2. כפונקציה של מספר האיטרציות כאשר אלפא מוגדר להיות 0.85. ניתן לראות בגרף כי כפי שאנו מצפים ככל שמספר האיטרציות עולה כך הערכים יורדים. בנוסף ניתן לראות כי ישנם איטרציות רבות בהן הערך נשאר קבוע (למשל בין 1,800 ל2,800), סיבה אפשרית לכך הינה שבסבירות גבוהה ככל שאנו מתקדמים לערך טוב יותר הסיכוי שצעד אקראי ישפר את הערך נמוך יותר. דבר מפתיע שניתן לראות בגרף הינו שהוא מונוטוני יורד. הסיבה שזה מפתיע אותנו הינה שהאלגוריתם SA אמור לאפשר צעדים "רעים" מידי פעם, הסבר אפשרי לתופעה הינו שהטמפרטורה יורדת בקצב מהיר מאוד ולכן ככל שאנו מתקדמים בגרף הסיכוי שנעשה צעד שאינו מקדם אותו שואף ל0. ניתן לראות כי הערך הכי טוב עומד על 109

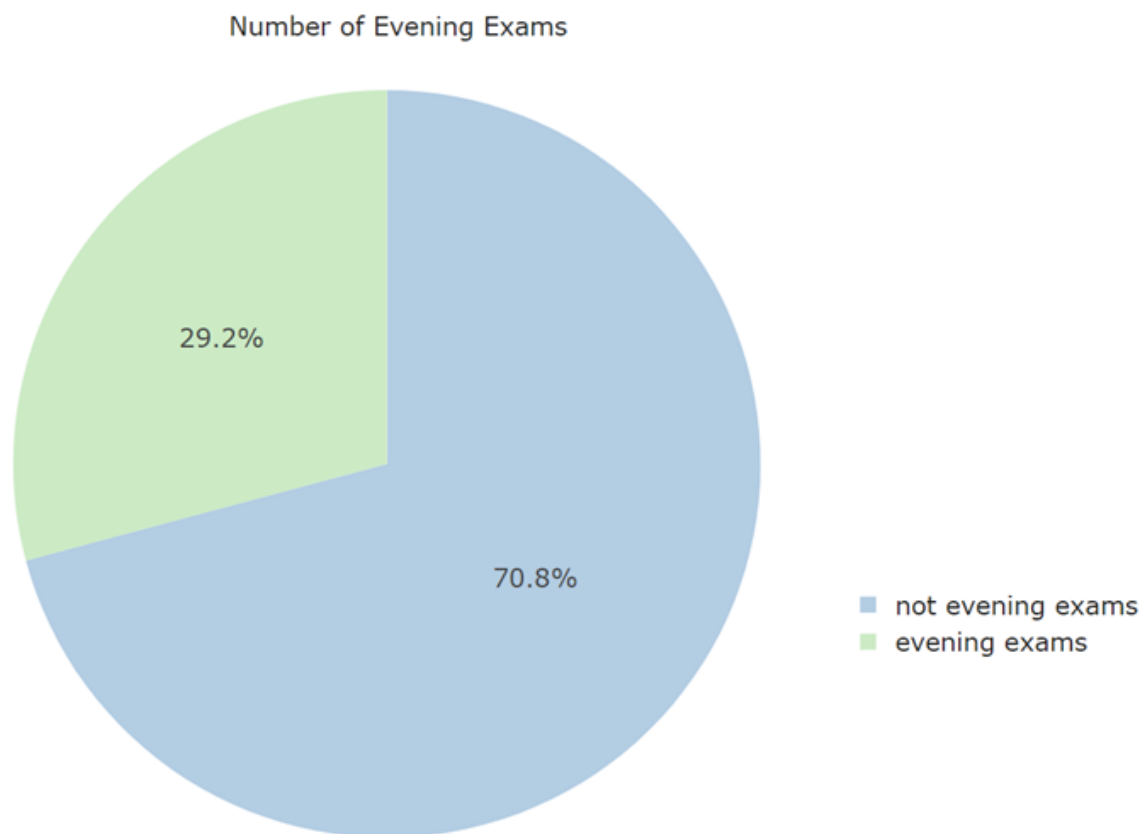
Deviation from Desired Deference between Two Courses



Course Number to Course Name: Deviation form difference

- | | |
|--|--------------------------------|
| ❖ 80430 – Introduction to Probability and Statistics | ❖ 67504 – Algorithms |
| ❖ 67501 – Introduction to Mathematical Logic for Programmers | ❖ 67125 – OOP |
| ❖ 57305 – Intermediate Macroeconomics | ❖ 67925 – NAND |
| ❖ 80312 – Probability Theory and Applications | ❖ 57307 – Price Theory 1 |
| ❖ 72120 – Biochemistry of The Cell | ❖ 77305 – Waves and Optics |
| ❖ 67312 – Programming Workshop in C/C++ | ❖ 80135 – Linear Algebra 2 |
| ❖ 6168 – Neurobiology of development and learning | ❖ 67109 – Data Structures |
| ❖ 83335 – Introduction to Electrical Engineering | ❖ 69301 – Physical Chemistry B |
| ❖ 80415 – Infinitesimal Calculus 3 | |
| ❖ 80420 – Probability Theory 1 | |
| ❖ 77303 – Analytical Mechanics | |

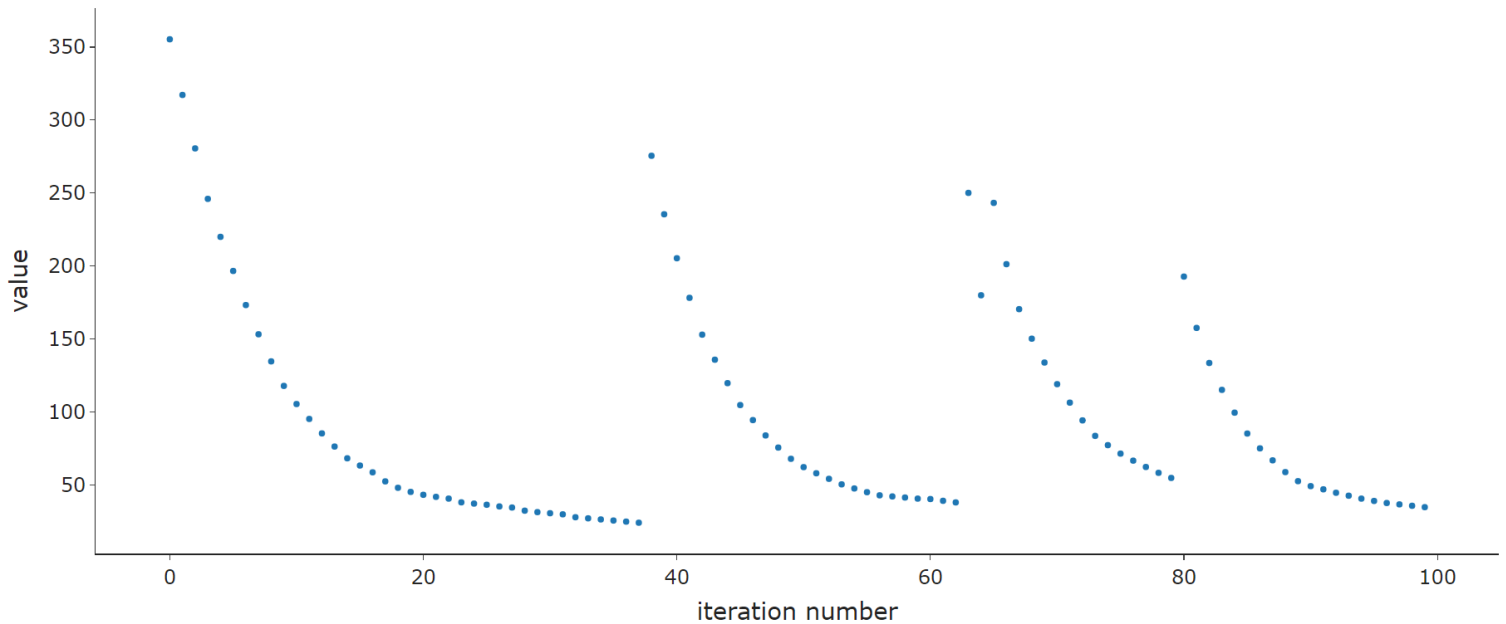
בגרף זה ניתן לראות את זוגות הקורסים שההפרשים בניהם קטנים מההפרשים הרצויים כאשר ציר הא מתאר את ההפרש בין ההפרש הרצוי בין הקורסים להפרש הקיים בשיבוץ. כפי שניתן לראות ישנם קורסים רבים שאינם עומדים באילוץ הקל של ההפרשים בשיבוץ זה. למשל עבור הסתברות ואלגוריתמים ההפרש עומד על 4 ומכיוון שההפרש הרצוי בין מבחנים במדעי המחשב הינו 6 נוצר מצב כי יש שני ימים ללמוד לאלגוריתמים מצב שהוא פחות מאופטימאלי.



בגרף זה ניתן לראות את היחס בין מספר המבחנים שמשובצים הערב לבין מספר המבחנים שלא משובצים בערב. שכן נתנו העדפה לשיבוץ שלא משבץ מבחנים בערב. ניתן לראות כי כאן כמעט 30 אחוז מהמבחנים משובצים בערב תוצאה גבוהה ביחס לשאר האלגוריתמים.

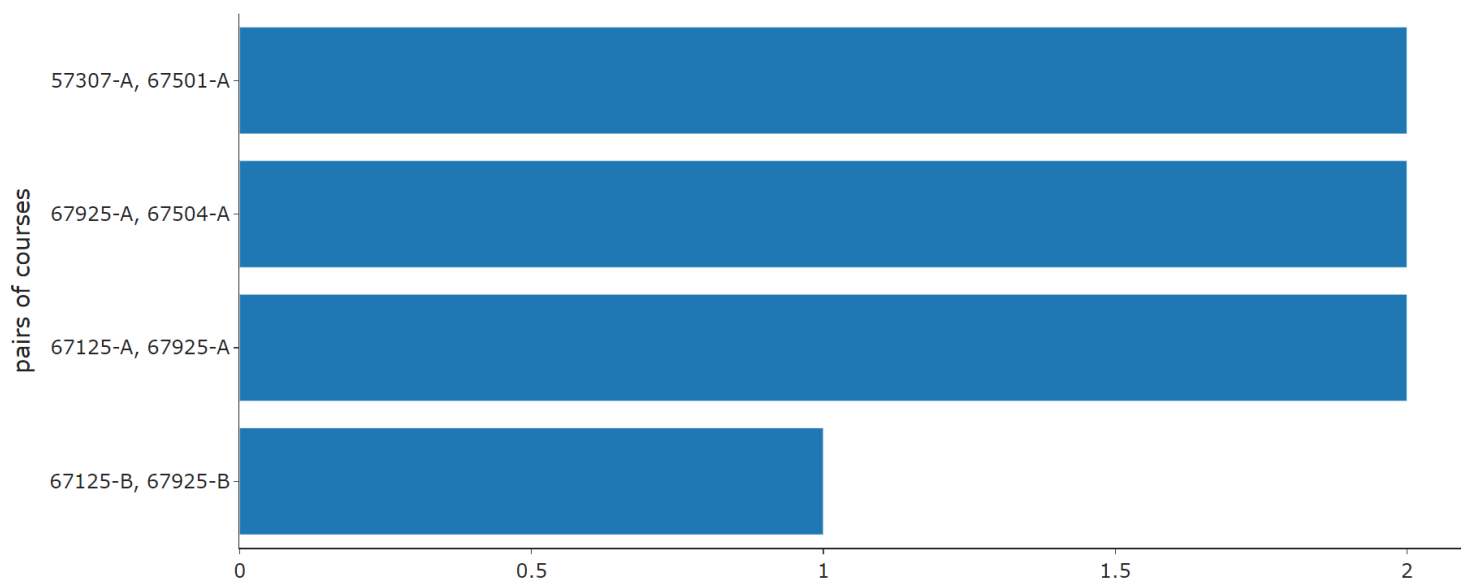
:Random Gradient Descent

RGD Values as a Function of Iteration Number



בגרף זה ניתן לראות את ערך פונקציית הערך שלנו (זו שמוגדרת בעמוד 8) כאשר הקבוע שאנו מכפילים במבחני הערב הוא 0.75 ובנוסף ה"קנס" שאנו נותנים על מבחנים שההפרש בניהם קטן בשלושה ימים או יותר מההפרש הרצוי בניהם הינו כפל ב2. כפונקציה של מספר האיטרציות כשאר קצב הלמידה של האלגוריתם עומד על 0.009. ניתן לראות כי יש מגמת ירידה בכל שמספר האיטרציות עולה כפי שהיינו מצפים. בנוסף ניתן לראות כי אחרי 40, 64, 80 איטרציות יש קפיצות גדולות יחסית של הערכים, קפיצות אלו ככל הנראה קורות כאשר האלגוריתם הגיע למינימום והוא מתחיל מחדש ממקום אקראי במרחב החיפוש. בנוסף ניתן לראות קפיצה ב65 שהיא כלל הנראה תוצאה של הצעד האקראי. ניתן לראות כי התוצאה הטובה ביותר של האלגוריתם הינה 30.71.

Deviation from Desired Deference between Two Courses



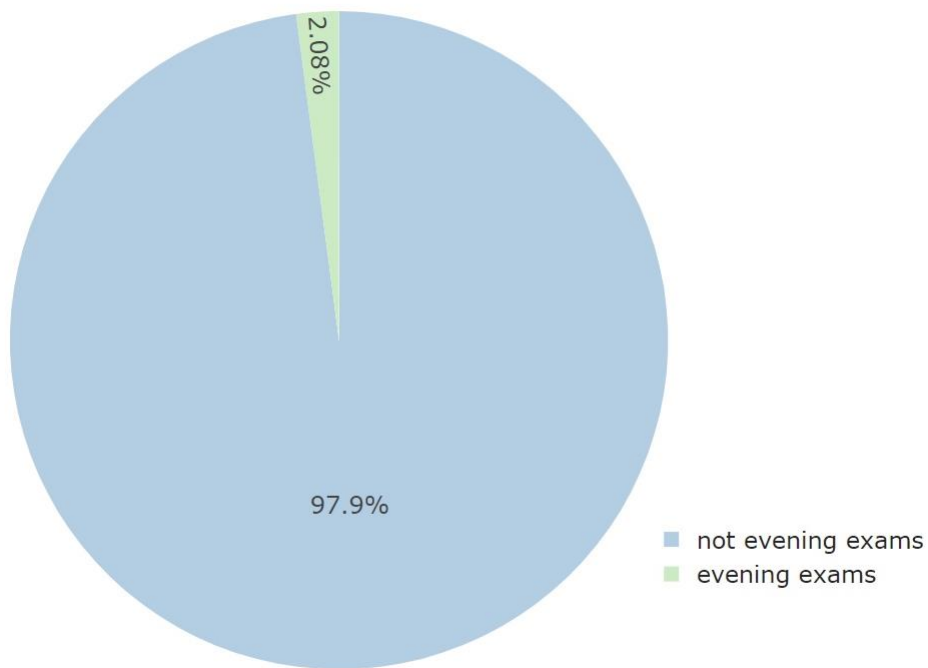
Course Number to Course Name:

Deviation form difference

- ❖ 67501 – Introduction to Mathematical Logic for Programmers
- ❖ 67504 – Algorithms
- ❖ 67125 – OOP
- ❖ 67925 – NAND
- ❖ 57307 – Price Theory 1

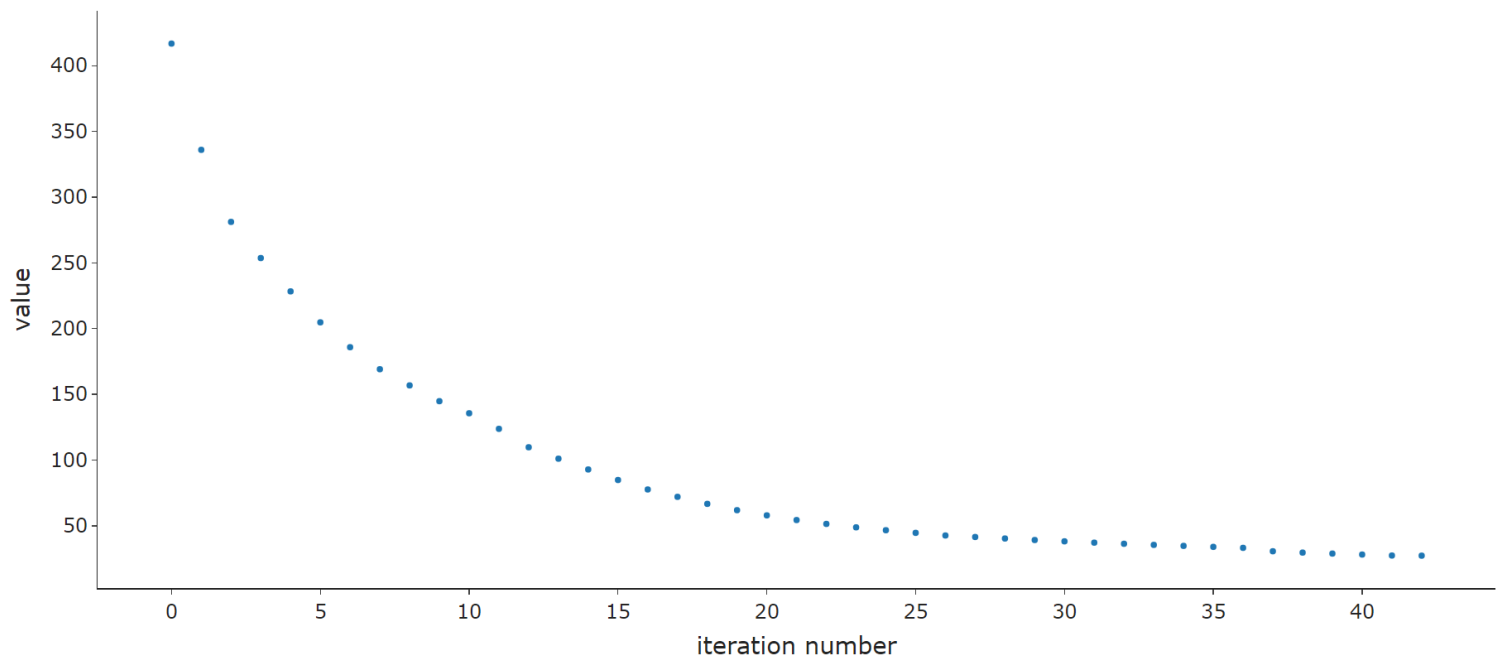
בגרף זה ניתן לראות את זוגות הקורסים שההפרשים בניהם קטנים מההפרשים הרצויים כאשר ציר הא מתאר את ההפרש בין ההפרש הרצוי בין הקורסים להפרש הקיים בשיבוץ. ניתן לראות כי בשיבוץ זה ישנם רק חמישה מבחנים שלא עומדים באילוץ הקל של ההפרשים וכי ההפרשים בניהם הם קטנים באופן יחסי. ניתן לראות כי ההפרש הגבוה ביותר הוא בין לוגיקה למתכנתים ומיקרו כלכלה והינו הפרש של שני ימים מכון שההפרש הרצוי עבור מבחנים של הנדסת חשמל הינו 6 זה אומר שיש הפרש של 4 ימים בין מיקרו כלכלה ללוגיקה למתכנתים.

Number of Evening Exams



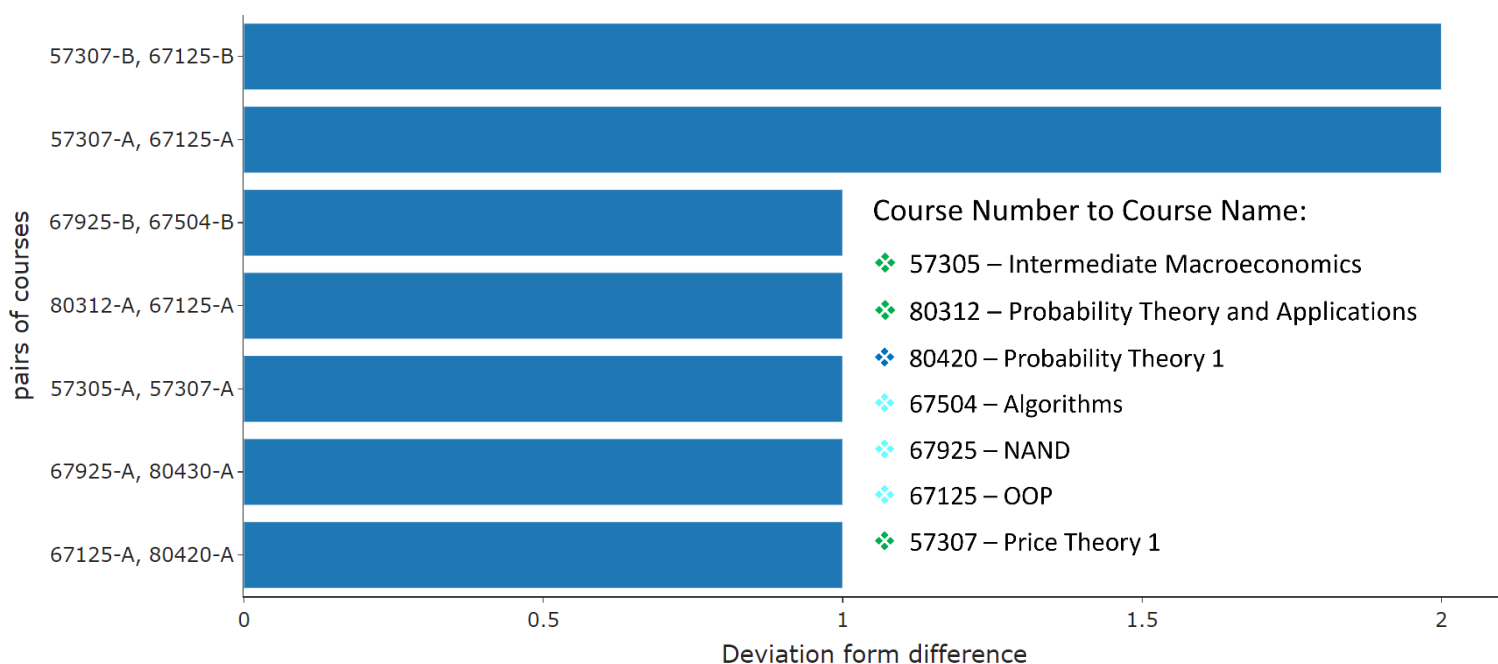
בגרף זה ניתן לראות את היחס בין מספר המבחנים שמשובצים בערב לבין מספר המבחנים שלא משובצים בערב. שכן נתנו העדפה לשיבוץ שלא משבץ מבחנים בערב. ניתן לראות כי רק שני אחוז מהמבחנים משובצים בערב עבור שיבוץ זה, דבר ששם את השיבוץ הזה ככמעט אידיאלי מהבחינה הזו.

GD Values as a Function of Iteration Number

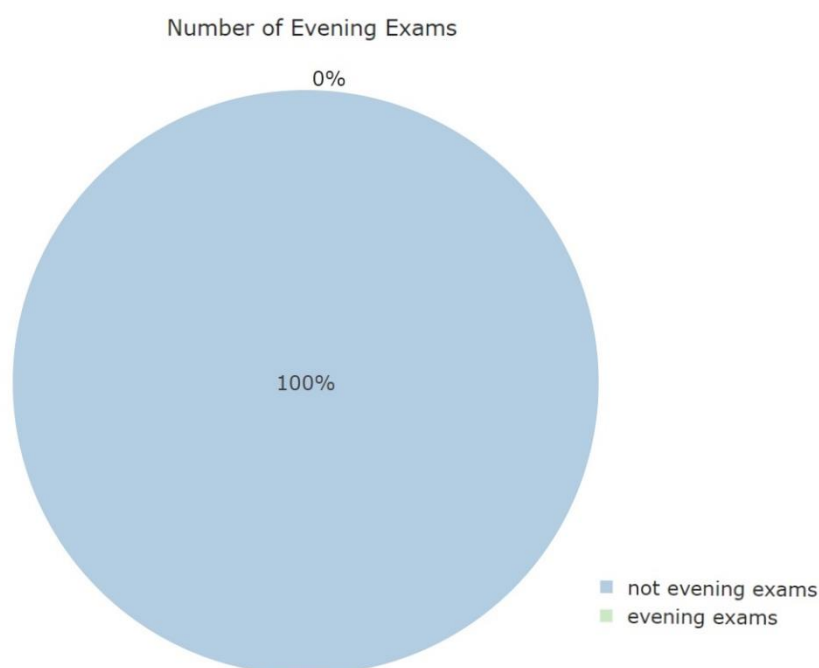


בגרף זה ניתן לראות את ערך פונקציית הערך שלנו (זו שמוגדרת בעמוד 8) כאשר הקבוע שאנו מכפילים במבחני הערב הוא 0.75 ובנוסף ה"קנס" שאנו נותנים על מבחנים שההפרש בניהם קטן בשלושה ימים או יותר מההפרש הרצוי בניהם הינו כפל ב2, כפונקציה של מספר האיטרציות. כפי שאנחנו מצפים ניתן לראות כי ששל שאנו עולים בכמות האיטרציות כך הערך יורד וכי הערך הטוב ביותר הוא בסביבות 25. בנוסף נתנו לאלגוריתם 100 איטרציות לרוץ ונראה כי הוא סיים לאחר 45 איטרציות. נתון זה מעיד על כך שהוא הגיע למינימום ולכן לא משנה כמה איטרציות היינו נותנים לאלגוריתם הוא לא היה משתפר. ראינו בנוסף כי בהרצות שנות התקבלו תוצאות שונות ולכן ניתן להגיד כי האלגוריתם מגיע למינימום לוקאלי ואינו מגיע לגלובאלי. ניתן לראות כי הערך הנמוך ביותר הינו 29.4.

Deviation from Desired Deference between Two Courses



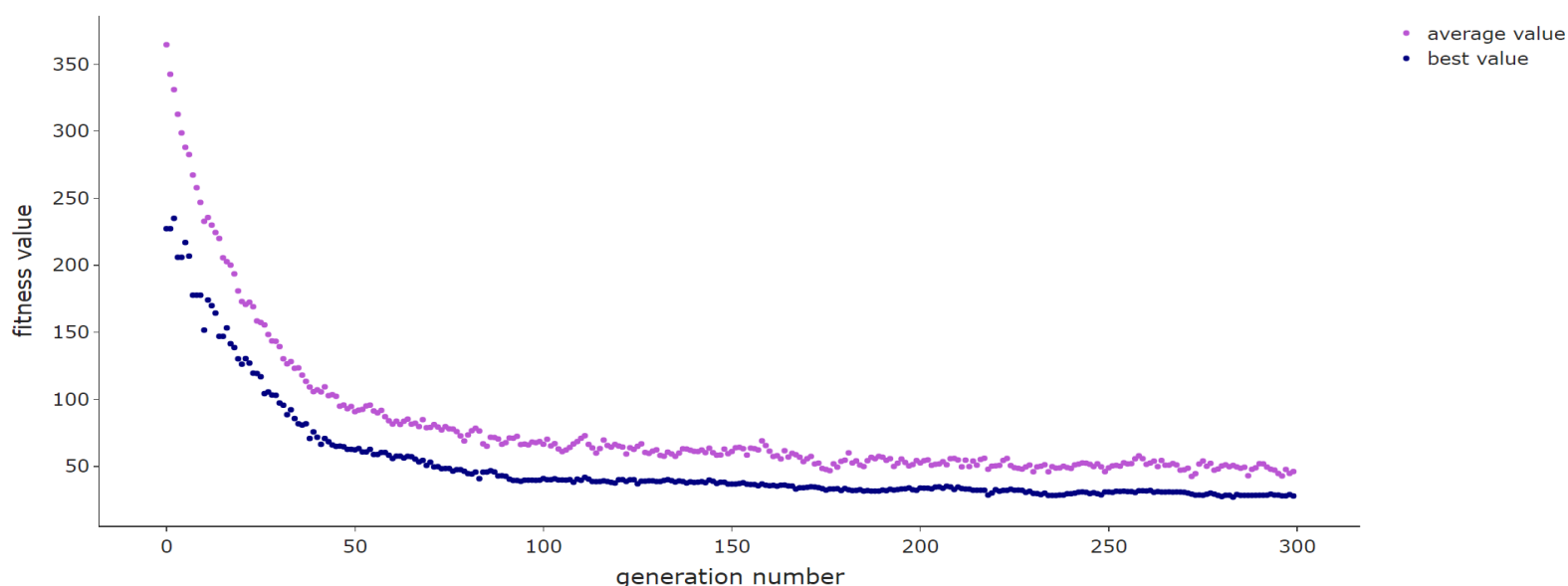
בגרף זה ניתן לראות את זוגות הקורסים שההפרשים בניהם קטנים מההפרשים הרצויים כאשר ציר ה-
מתאר את ההפרש בין ההפרש הרצוי בין הקורסים להפרש הקיים בשיבוץ. ניתן לראות כי בשיבוץ זה ישנם
רק שישה מבחנים שלא עומדים באילוף הקל של ההפרשים המהווים שבעה זוגות וכי ההפרשים בניהם הם
קטנים באופן יחסי. בנוסף ההפרש הגדול ביותר הוא הבין מועד ב של מיקרו כלכלה למועד הב בתכנות
מונחה עצמים, ההפרש הינו 2 מכיוון שהפרש הרצוי בין מועדי ב של הנדסת חשמל הינו 4 ימים ישנם יומיים
ללמידה לתכנות מונחה עצמים.



בגרף זה ניתן לראות את היחס בין
מספר המבחנים שמשובצים בערב
לבין מספר המבחנים שלא משובצים
בערב. שכן נתנו העדפה לשיבוץ שלא
משבץ מבחנים בערב. ניתן לראות כי
כל המבחנים אינם משובצים בערב
כלומר זהו שיבוץ אידיאלי מהבחינה
הזאת.

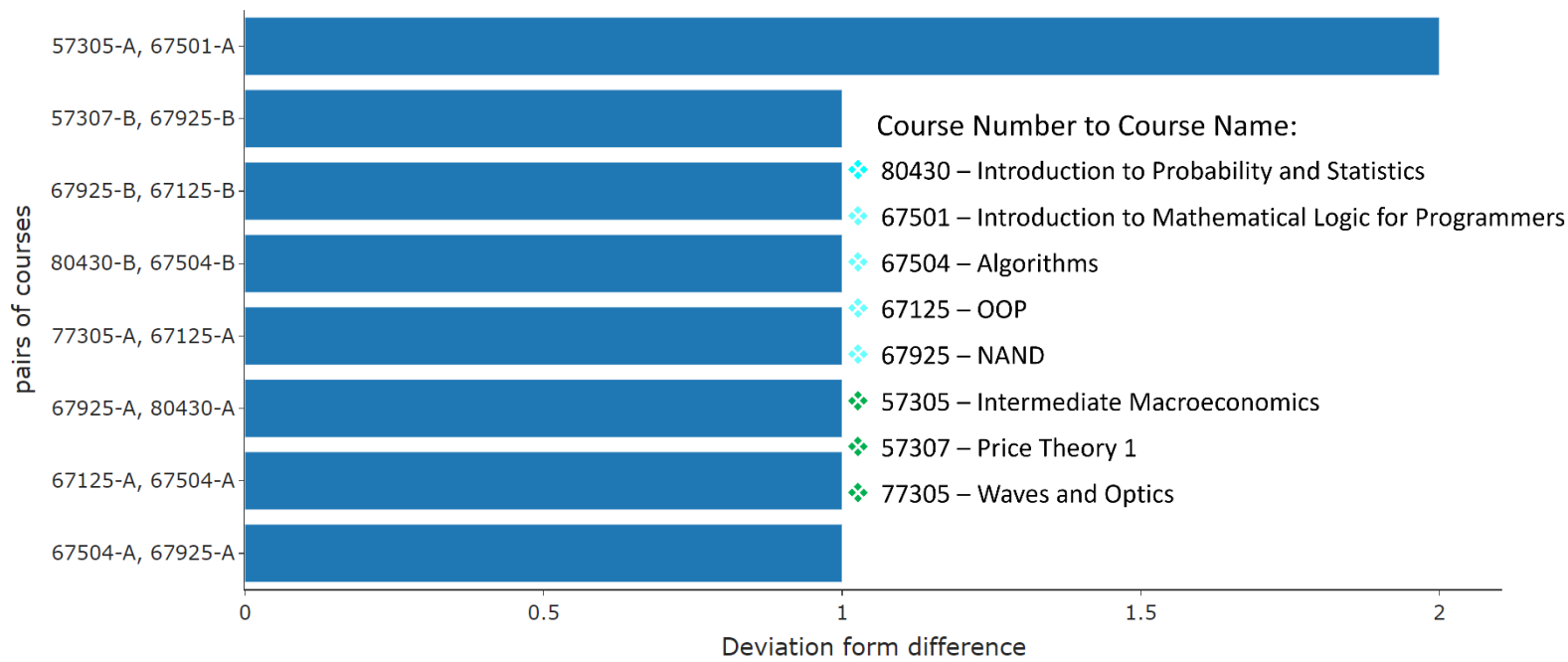
:Genetic Algorithm

GA Average and Best as a Function of Gen Number in courses



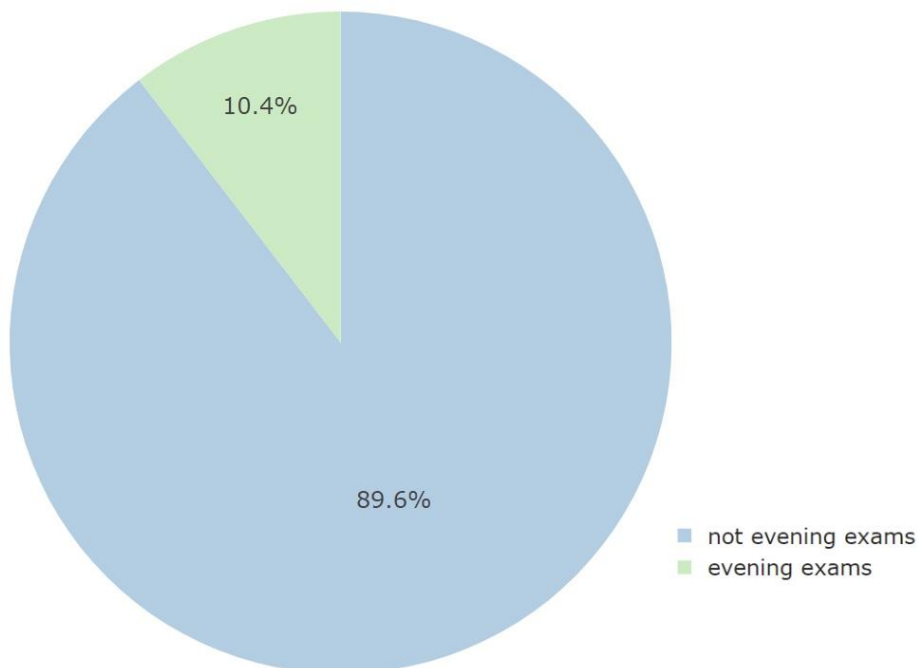
בגרף זה ניתן לראות את ערך ההתאמה של השיבוץ הכי טוב ושל ממוצע השיבוצים בדור כפונקציה של מספר הדור. ערך ההתאמה נקבע על ידי פונקציית הערך שלנו (זו שמוגדרת בעמוד 8) כאשר הקבוע שאנו מכפילים במבחני הערב הוא 0.75 ובנוסף ה"קנס" שאנו נותנים על מבחנים שההפרש בניהם קטן בשלושה ימים או יותר מההפרש הרצוי בניהם הינו כפל ב-2. בבעיה זו אנו מחפשים את ערך ההתאמה הנמוך ביותר. ניתן לראות כי כפי שהיינו מצפים ככל שאנו עולים במספר הדורות אנו רואים מגמת ירידה כללית של ערך ההתאמה הן בפריט הטוב ביותר והן בממוצע הפריטים. בניגוד לאלגוריתם הGD כאן הפונקציה אינה מונוטונית יורדת, הסיבה לכך היא האקראיות הרבה באלגוריתם, כאשר יש אקראיות רבה ישנו גם סיכוי "להרוס" את הפתרון הטוב ביותר ולהגיע לפתרונות פחות טובים. ניתן לראות כי בערך לאחר 100 דורות מגמת הירידה מתמתנת וכי הערך הטוב ביותר עומד על 27.15.

Deviation from Desired Deference between Two Courses



בגרף זה ניתן לראות את זוגות הקורסים שההפרשים בניהם קטנים מההפרשים הרצויים כאשר ציר הא מתאר את ההפרש בין ההפרש הרצוי בין הקורסים להפרש הקיים בשיבוץ. ניתן לראות כי בשיבוץ זה ישנם רק שמונה מבחנים שלא עומדים המהווים שמונה זוגות באילוץ הקל של ההפרשים וכי ההפרשים בניהם הם קטנים באופן יחסי. בנוסף ההפרש הגדול ביותר הינו בין מיקרו כלכלה לוגיקה למתכנתים והינו 2.

Number of Evening Exams



בגרף זה ניתן לראות את היחס בין מספר המבחנים שמשובצים בערב לבין מספר המבחנים שלא משובצים בערב. שכן נתנו העדפה לשיבוץ שלא משבץ מבחנים בערב. ניתן לראות כי 10 אחוז מהמבחנים משובצים בערב שזה אחוז יחסית גדול לשאר האלגוריתמים.

השוואה בין אלגוריתמי החיפוש:

השוואה בין ערכי פונקציית הערך שקבענו (עמוד 8) באלגוריתמים השונים:

ראשית ניתן לראות כי עבור SA ערך ההחזרה הינו בסביבות 100 שהוא גבוה באופן משמעותי מערכי ההחזרה של שאר האלגוריתמים ולכן כפי שמימשנו אותו הוא פחות רלוונטי לפתרון בעיה זו, ככל הנראה הסיבה לכך הינה שפונקציית הקירור דועכת מהר מידי ולכן האלגוריתם לא מאפשר צעדים "רעים" ומכאן שהוא נתקע במינימום מקומי. לעומתו האלגוריתמים GD, RGD, GA מחזירים ערכים משמעותיים יותר נמוכים (בסביבות 30) כשאר GA הגיעה לתוצאה הטובה ביותר של 27.15.

השוואה בין ההפרשים הרצויים למצויים בשיבוץ האופטימאלי:

ניתן לראות כי עבור SA מספר זוגות הקורסים עם הפרות הוא גדול ביחס לשאר האלגוריתמים העומד על 19 וכן ההפרש המקסימאלי הוא 4 לעומת 2 בשאר האלגוריתמים. שלושת האלגוריתמים הבאים GA, GD, RGD מחזירים תוצאות הדומות מאוד בערכן עם ההבדלים הבאים בפתרון שמחזיר RGA ישנם פחות מבחנים עם הפרות אך ההפרות בניהן יותר גדולות, לעומתו בGA ישנם יותר מבחנים עם הפרות אך רוב ההפרות קטנות יותר ולכן הפתרון של GA מעט טוב יותר כי נעדיף 8 מבחנים שההפרש בניהם הוא 5 במקום 6 מאשר שלושה מבחנים שההפרש בניהם הוא 4 לעומת 6 שרצוי.

השוואה בין מספר המבחנים בערב:

גם כאן SA לוקח עם כמעט 30% מהמבחנים שמשובצים בערב, לעומתו GA במקום השני עם 10% של מבחנים המשובצים בערב וGD, RGD המובילים עם 2% ו0% של מבחנים בערב. למרות שכאן הפתרון של GD ו RGD מביאים לערך טוב יותר עדיין בערך הכולל הפתרון של GA טוב יותר ולכן ניתן להסיק כי לנתון זה אין השפעה רבה על התוצאה הסופית.

השוואה כללית בין האלגוריתמים:

אפילו שמבחינת זמני ריצה SA לוקח את שאר האלגוריתמים הוא לא מגיע לתוצאות שמתקרבות אליהם ולכן האלגוריתם של SA כמו שמימשנו אותו עם פונקציית הקירור שלנו אינו מתאים לפתרון הבעיה. ניתן לראות כי מבחינת תוצאות אין הבדל משמעותי בין GD, RGD, GA אך מבחינת זמני ריצה GA הינו הטוב ביותר כאשר הוא רץ בזמן של כחצי שעה לעומת זמן של 50-60 דקות שבהם רצים GD, RGD. בנוסף בבעיה זו RGD לא היווה יתרון על GD מכיוון שהוא מפיק תוצאות דומות אך זמן הריצה שלו גדול באופן משמעותי מכיוון שהוא מבצע את כל האיטרציות. לסיכום ראינו כי GA הינו האלגוריתם היעיל ביותר לפתרון בעיה זו מבין האלגוריתמים שניסונו לפי הדרך בה בחרנו להגדיר את הבעיה.

ניתוח תוצאות תורת המשחקים והשוואה אל מול Genetic Algorithms:

בחלק זה נתבונן בתוצאות הן של תורת המשחקים והן של GA עבור דוגמא קטנה הכוללת שלושה חוגים: מדעי המחשב עם 40 סטודנטים, מתמטיקה עם 30 והנדסת חשמל עם 20 סטודנטים.

בדוגמא החוגים ינסו לשבץ את המבחנים הבאים:

מתמטיקה: אנפי (7 נ"ז), ליניארית (6 נ"ז) ותורת הקבוצות (3 נ"ז)
 מדעי המחשב: אנפי (7 נ"ז), ליניארית (6 נ"ז) ומתמטיקה דיסקרטית (5 נ"ז)
 הנדסת חשמל: מכניקה (7 נ"ז), חשמל (6 נ"ז) ומתמטיקה דיסקרטית (5 נ"ז)

שיבוץ המבחנים של הבעיה בתורת המשחקים כאשר הקואליציות הן, מתמטיקה והנדסת תוכנה ביחד ומדעי המחשב:

		6		3		2		1		31		30
				חשמל		אלגברה ליניארית						
		14		13		10		9		8		7
		תורת הקבוצות		מתמטיקה דיסקרטית				מכניקה		חשבון אינפיניטסימלי		

נבחין כי ישנו הפרש של יומיים בין מכניקה לדיסקרטית שזהו ההפרש הקטן ביותר בלוח גם מבחינת ימים וגם מבחינת נקודות זכות, לעומת זאת ישנו הפרש של ארבעה ימים בין אלגברה ליניארית לאנפי שזהו ההפרש הגדול ביותר בלוח הן מבחינת ימים והן מבחינת נקודות זכות. שיבוץ זה הוא השיבוץ האידיאלי עבור החוג למתמטיקה לפי הגדרת המשחק.

שיבוץ המבחנים של הבעיה בתורת המשחקים כאשר הקואליציות הן, מתמטיקה ומדעי המחשב ביחד והנדסת תוכנה:

		6		3		2		1		31		30
				חשמל		אלגברה ליניארית						
		14		13		10		9		8		7
		מתמטיקה דיסקרטית		תורת הקבוצות				מכניקה		חשבון אינפיניטסימלי		

נבחין כי ישנו הפרש של שלושה ימים בין מכניקה לדיסקרטית שזהו ההפרש הקטן ביותר בלוח גם מבחינת ימים וגם מבחינת נקודות זכות, לעומת זאת ישנו הפרש של ארבעה ימים בין אלגברה ליניארית לאנפי שזהו ההפרש הגדול ביותר בלוח הן מבחינת ימים והן מבחינת נקודות זכות. שיבוץ זה הוא השיבוץ האידיאלי עבור החוגים מדעי המחשב והנדסת תוכנה לפי הגדרת המשחק.

ניתן לראות כי בדוגמא שנתנו לא ניתן היה להגיע לקואליציה בה כל חברי הקואליציה יהיו מרוצים שכן החוג למתמטיקה יוכל להשיג לוח טוב יותר אם יהיה בקואליציה עם הנדסת תוכנה וכי החוג להנדסת תוכנה דווקא יקבל ניקוד טוב יותר אם מתמטיקה ומדעי המחשב יקימו קואליציה.

הערות:

לצורך פשטות המשחק והדוגמא לא שיבצנו מבחנים בשישי ושבת. בנוסף לא החשבנו את שיש ושבת כימי לימוד ולכן ההפרש בין ליניארית לאנפי הוא עדיין 4 ימים ולא 6.

שיבוץ המבחנים בתקופת הזמן המוגדרת שהוציא האלגוריתם Genetic Algorithm:

יום ה' 3 Linear Algebra - A. אח. 1:30 ●	יום ד' 2	יום ג' 1 בפברואר Mechanics and Sp. אח. 1:30 ●	יום ב' 31	יום א' 30 Infinitesimal Calculus - 9 לפ. ●
10 Set Theory - A. אח. 1:30 ●	9	8 Discrete Mathematics. 9 לפ. ●	7	6
			14	13 Electricity and Maç. אח. 1:30 ●

ניתן לראות כי כאן ההפרש הקטן ביותר הן מבחינת הימים והן מבחינת נקודות הזכות הוא בין מתמטיקה דיסקרטית לחשמל, הפרש של 3 ימים. אחת הסיבות לכך היא שאלגוריתם הלמידה שלנו כן מחשיב את שישי ושבת כימי למידה ולכן מבחינתו יש 5 ימים ולא 3. נבחין כי השיבוץ של האלגוריתם משבץ מבחנים בימים הראשונים של תקופת הבחינות כך שאנפי למשל מתחיל ביום של תקופת המבחנים ולכן אין זמן רב ללמוד אליו.

השוואה בין תוצאות Genetic Algorithm לבין תוצאות תורת המשחקים:

ההשוואה כאן היא מעט קשה מכיוון שהבעיות מוגדרות באופן שונה. אבל מה שניתן לראות מהתוצאות הינו שGA מצליח לשמר בצורה טובה יותר את הפרש התאריכים הרצוי גם על תקופת זמן קצרה יחסית, בנוסף ניתן לקבל בצורה זו שיבוץ מהיר ויעיל יותר שכן הפתרון של תורת המשחקים פחות ישים עבור מספר קורסים גדול יותר ומספר גדול יותר של משבצות זמן. מנגד הפתרון של GA לא מייחס חשיבות לרצון של חוג אחד על פני חוג אחר דבר שניתן לראות שתורת המשחקים מקיימת על ידי הסתכלות בדוגמא של הקמת קואליציה בין מתמטיקה להנדסת חשמל. דבר נוסף שהפתרון של GA אינו מתייחס אליו הינו כמות הזמן הנדרשת ללמידה למבחן שאנו בחרנו לאמוד אותה לפי כמות הנ"ז של הקורס למשל קל לראות שמבחינה זו בשיבוץ של GA לו היינו מחליפים את מתמטיקה דיסקרטית ואת אלגברה ליניארית היינו מקבלים לוח טוב יותר עבור הסטודנטים. לעומתו בפתרון של תורת המשחקים יש התייחסות לנתון זה כחלק מפונקציית הערך של המשחק.

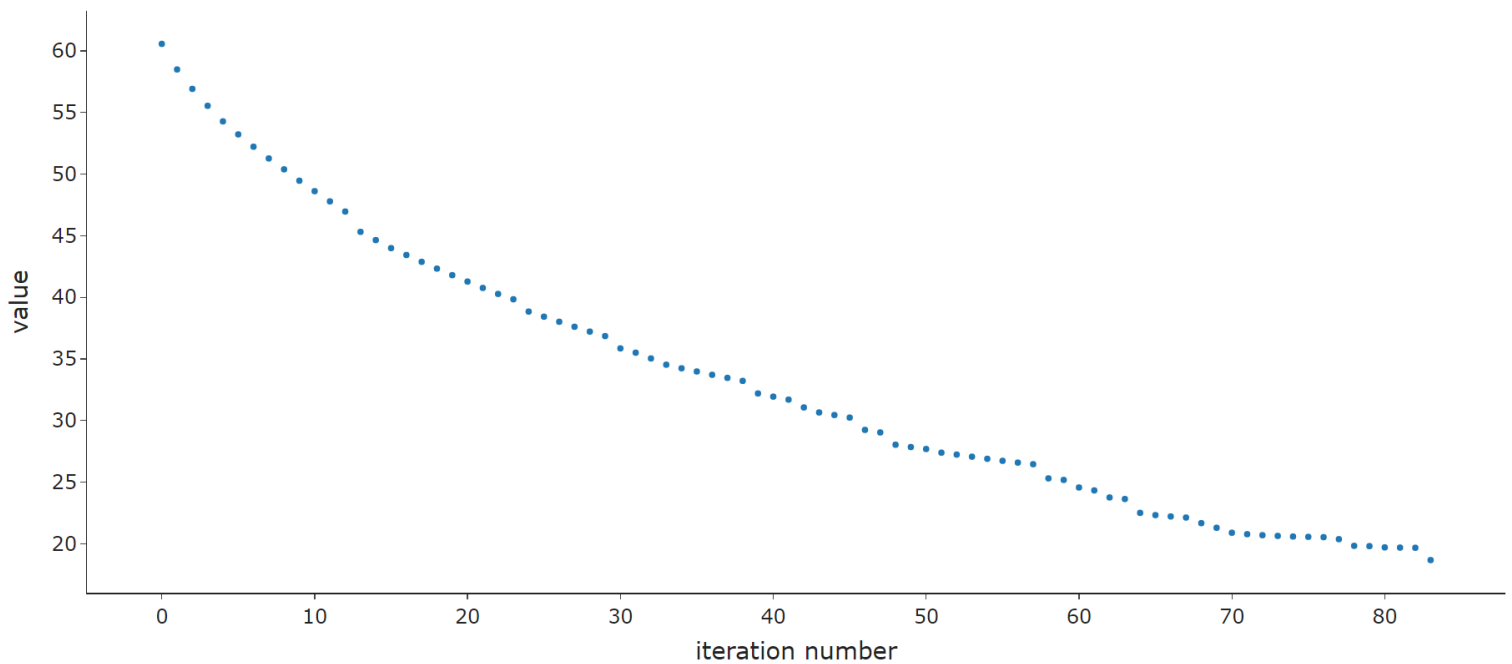
השוואה בין האלגוריתמים בעיית שיבוץ מבחנים לאולמות:

נציין כי הקלט של אלגוריתמים אלו שאנו רואים בתוצאות הוא שיבוץ לוח המבחינות התקין שאת הערכתו ניתן לראות בגרפים בעמודים הקודמים. מכאן שהקלט ל GA ו GD הינו קלט שונה ולכן יכול להוות הגורם להבדלים בין הגרפים השונים, אך כיוון שאנו מגדרים בעיה זו כהרחבה של הבעיה הקודמת אנו סבורים כי אין בעיה עם גישה זו וכי התוצאה של הפתרון הכולל היא זו שמעניינת בעינינו.

גרפים ותוצאות:

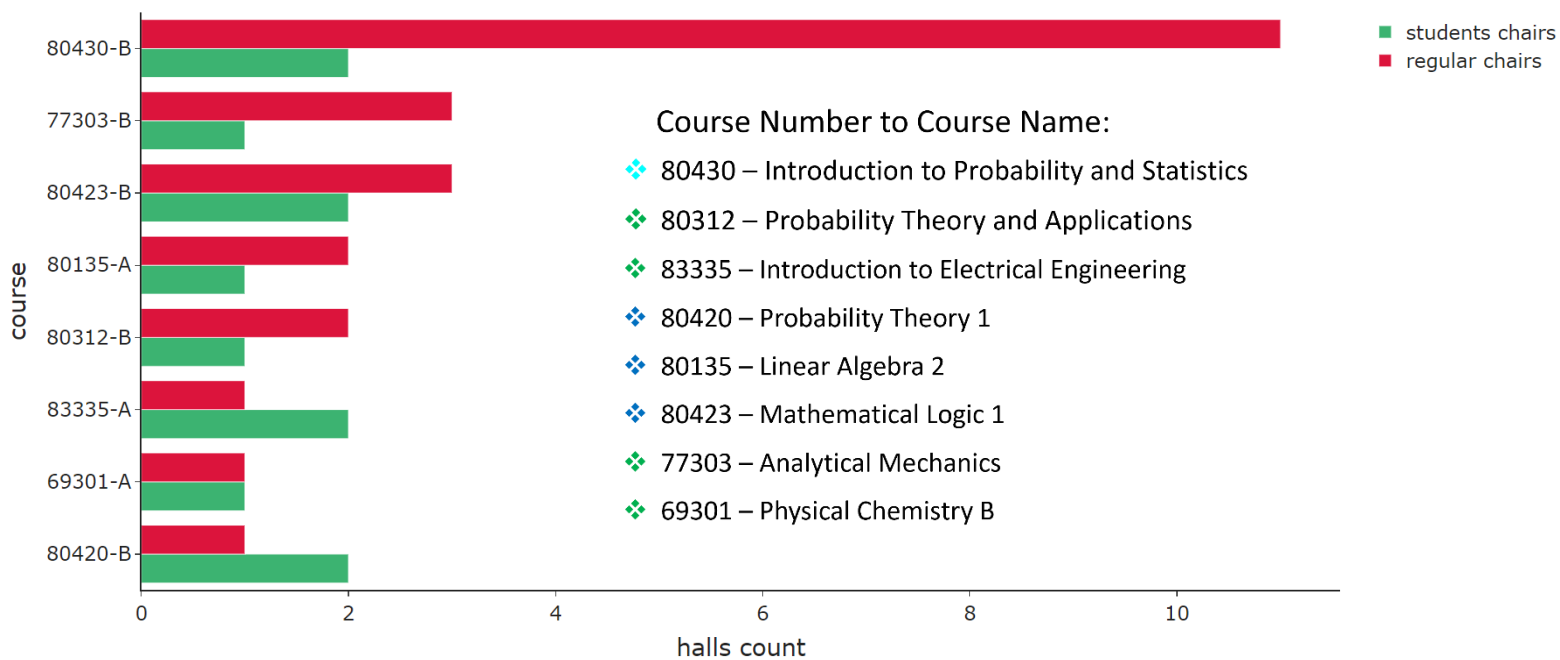
:Gradient Descent

GD Values as a Function of Iteration Number in Halls



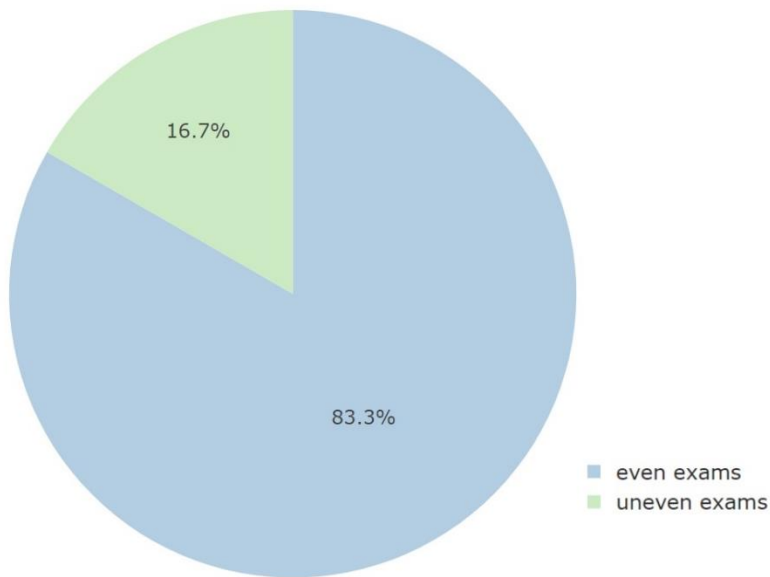
בגרף זה ניתן לראות את ערך פונקציית הערך שלנו (זו שמוגדרת בעמוד 13) כשאר הקבוע שאנו מכפילים במרחקים של האולמות הינו 1.5 והקבוע שאנו בודקים עבור הצפיפות הינו 1.25, כפונקציה של מספר האיטרציות. הגרף כפי שאנו מצפים הינו מונוטוני יורד, נבחין כי הגרף מסתיים לאחר 83 איטרציות לעומת 150 איטרציות שניתנו בבנאי ומכאן ניתן להסיק שהוא מגיע למינימום מקומי (שיכול להיות גם גלובאלי) עם ערך של 15.32.

Number of Halls with Students Chairs vs Number of Halls with Regular Chairs in Uneven Exams



בגרף זה ניתן לראות את כל המבחנים שלא כל האולמות שמשוברים אליהם הם בעלי סוג כיסאות זהה ואת היחס בין אולמות עם כיסאות רגילים לאולמות עם כיסאות סטודנט במבחנים אלו. ניתן לראות כי יש מספר קטן יחסית של מבחנים כאלו, 8 מתוך 48. נבחין כי בקורס כמיה פיזיקלית ניתן לראות שיש אולם 1 עם כיסאות רגילים ואולם אחד עם כיסאות סטודנט. מבחינתנו זהו היחס הגרוע ביותר מכון שאנו מחשבים זו חלוקה לא הוגנת אך מכון שאלו רק שני אולמות המשקל של חלוקה זו זניח באופן יחסי. בנוסף ניתן לראות כי בקורס מבוא להנדסת חשמל הייתה העדפה לשבץ את רוב האולמות בכיסאות סטודנט מאשר בכיסאות רגילים. תופעה זו מעניינת שכן נראה כי הפתרון העדיף שיבוץ שאנו רואים כהוגן מאשר שיבוץ שאנו מחשיבים כנוח.

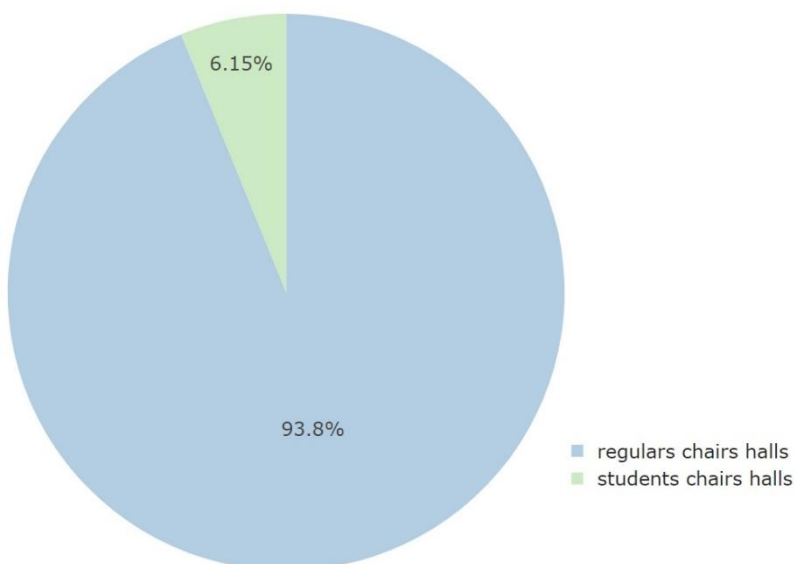
Number of Exams with both Students Chairs and Regular Chairs



בגרף זה ניתן לראות את אחוז המבחנים שבהם משובצים אולמות עם כיסאות שונים לעומת אחוז המבחנים שכל האולמות משמובצים עליהם בעלי אותם סוגי כיסאות. ניתן לראות כי קרוב ל-20% מהאולמות לא בעלי אותם סוגי כיסאות בשיבוץ זה.

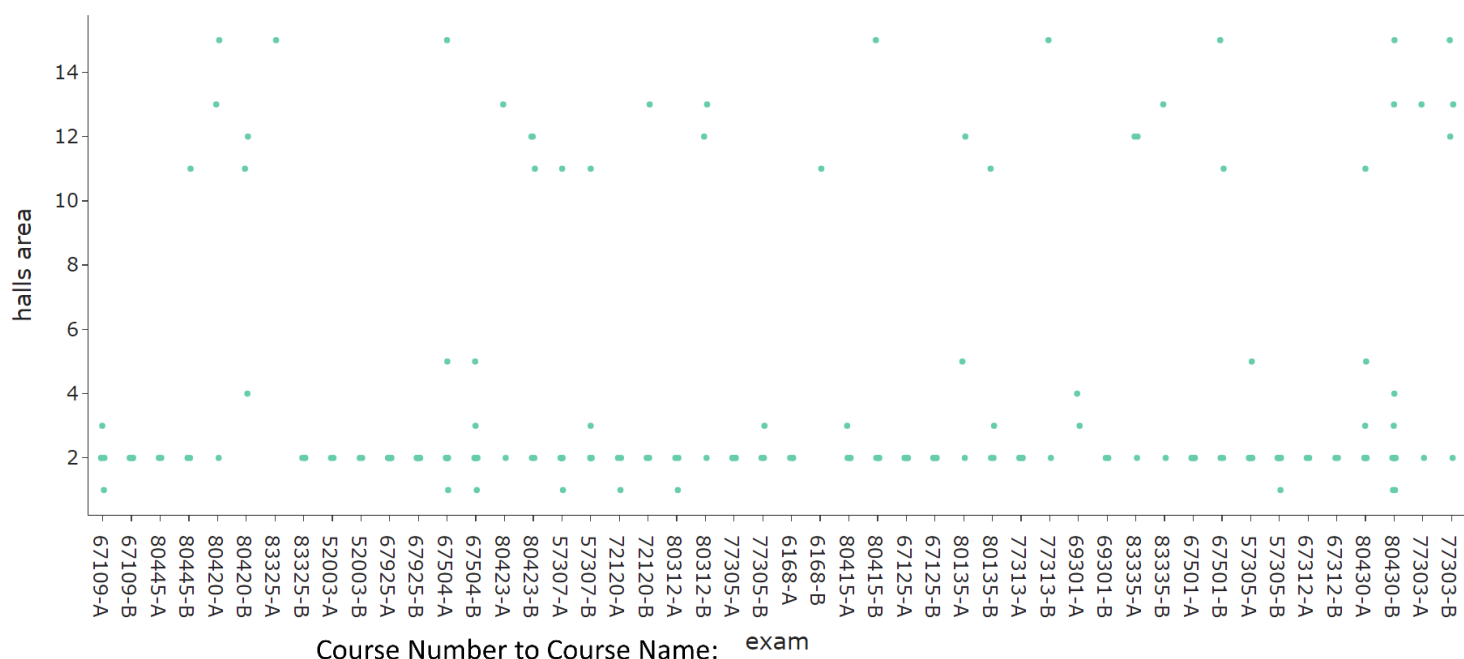
שני הגרפים בעמוד זה אמורים לשקף את הוגנות השיבוץ עבור הסטודנטים.

Number of Halls with Students Chairs vs Number of Halls with Regular Chairs



בגרף זה ניתן לראות את אחוז האולמות עם כיסאות רגילים לעומת אחוז האולמות עם כיסאות סטודנט בשיבוץ. גרף זה אמור לייצג את רמת הנוחות של הסטודנטים בעת ההיבחנות.

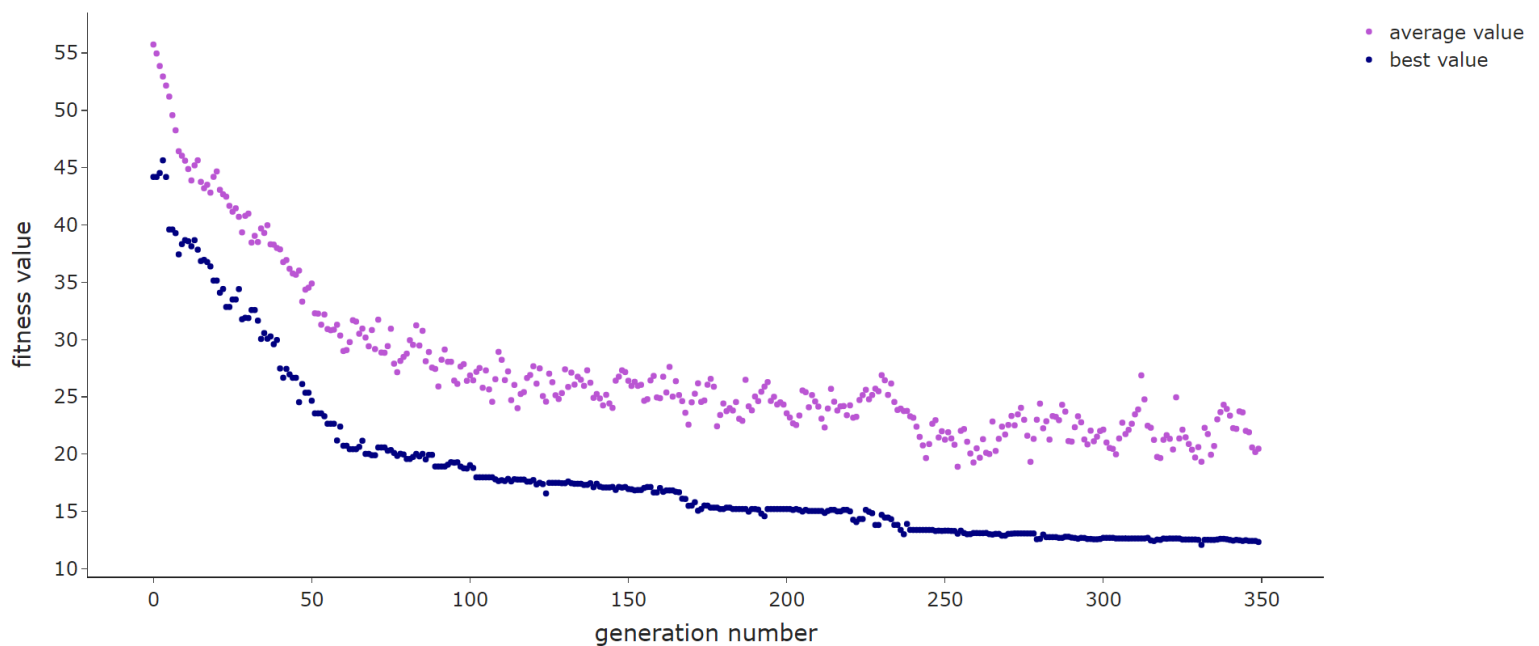
Distance between Halls that were Assigned to a Certain Exam



- | | | |
|---|--|---|
| <ul style="list-style-type: none"> 80430 – Introduction to Probability and Statistics 67501 – Introduction to Mathematical Logic for Programmers 57305 – Intermediate Macroeconomics 80312 – Probability Theory and Applications 72120 – Biochemistry of The Cell 52003 – Introduction to Statistics 67312 – Programming Workshop in C/C++ 6168 – Neurobiology of development and learning 83335 – Introduction to Electrical Engineering 83325 – Waves and Fundamentals of Modern Physics 1 77313 – Equations of Mathematical Physics 80415 – Infinitesimal Calculus 3 80420 – Probability Theory 1 77303 – Analytical Mechanics | <ul style="list-style-type: none"> 67504 – Algorithms 67125 – OOP 67925 – NAND 57307 – Price Theory 1 77305 – Waves and Optics 80135 – Linear Algebra 2 67109 – Data Structures 80445 – Algebraic Structures 1 80423 – Mathematical Logic 1 69301 – Physical Chemistry B | <p>Area to Hall</p> <ul style="list-style-type: none"> 1 – Canada 2 – School of Computer Science and Engineering 3 – Levin 4 – Mathematics 5 - Chemistry 7 11 – Feldman 12 – Sprinzak 13 – Levi 15 – Rotberg, Kaplan |
|---|--|---|

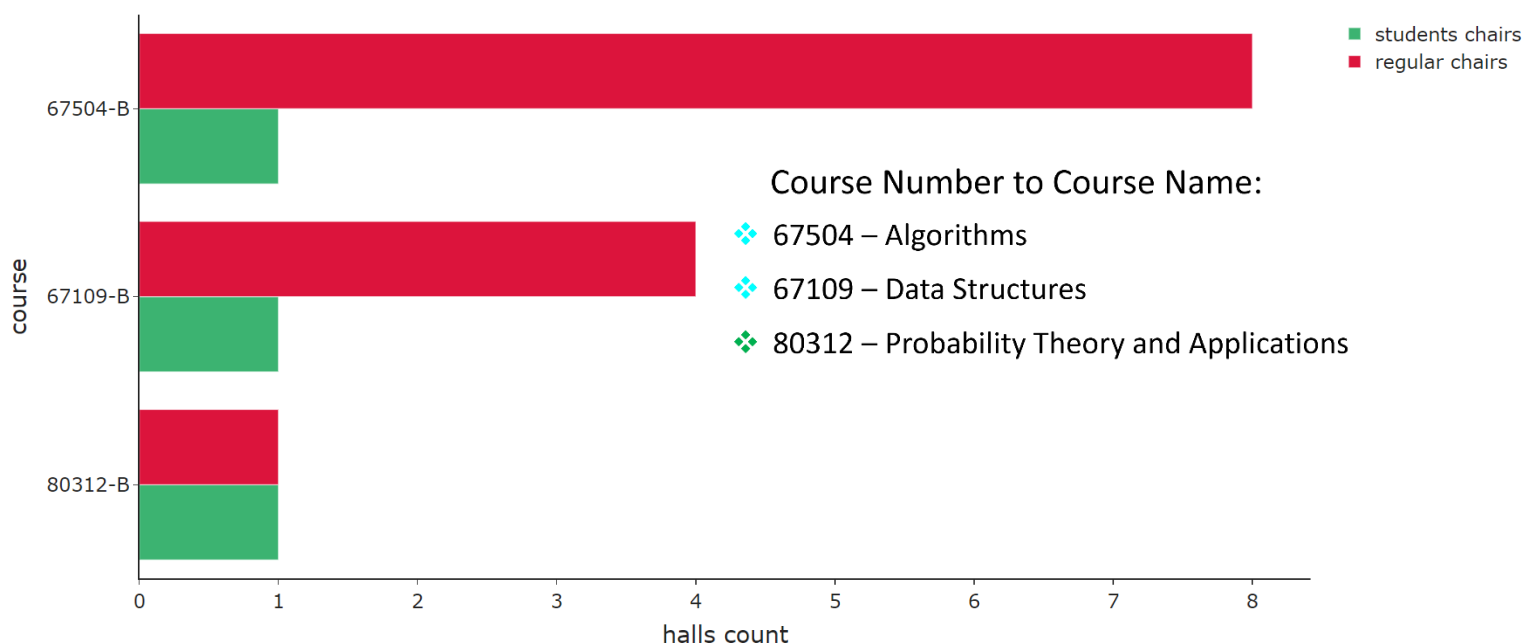
בגרף זה ניתן לראות את ההתפרשות של המבחנים על אזורים שונים באוניברסיטה כאשר ציר ה-א הינו כל המבחנים שאנו מוצאים להם שיבוץ בבעיה (מועדי א וב) ואילו ציר ה-ב הינו האזורים באוניברסיטה כאשר חילקנו את מספרי האזורים לפי קרבה פיזית בקמפוס (למשל לוין ומתמטיקה קרובים ולכן מספרי האזור שלהם קרובים) הנקודות הכחולות מציינות את האולמות ששובצו למבחן בטור שלהן. למשל עבור הקורס מבנה נתונים מועד א ניתן לראות כי האולמות שלו קרובים יחסית וכולם משובצים באזורים בין 2 ל 1 לעומת האולמות של מועד ב בהסתברות מפוזרים לאזורים רבים בקמפוס. גרף זה בא להראות כמה הפתרון מצליח לשמור על האילוץ הקל שגורף כי כל האולמות המשובצים למבחן מסוים צריכים להיות באותו אזור. ומטרתו היא בעיקר נוחות למתרגלים שלא יצטרכו להתרוצץ בין האולמות השונים במהלך המבחן.

GA Average and Best as a Function of Gen Number in Halls



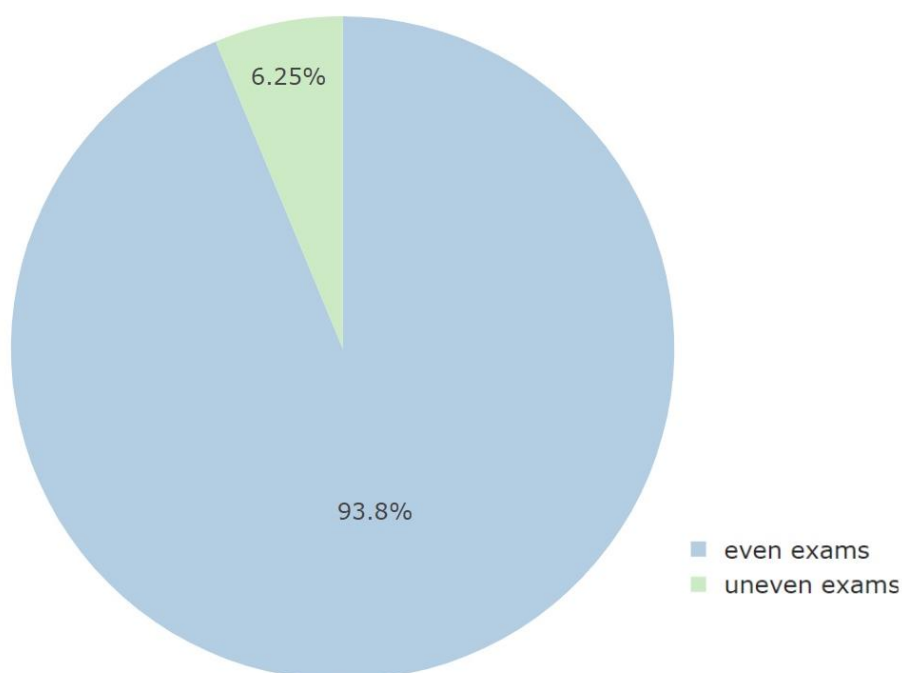
בגרף זה ניתן לראות את ערך ההתאמה של השיבוץ הכי טוב ושל ממוצע השיבוצים בדור כפונקציה של מספר הדור. ערך ההתאמה נקבע על ידי פונקציית הערך שלנו, כשאר הקבוע שאנו מכפילים במרחקים של האולמות הינו 1.5 והקבוע שאנו בודקים עבור הצפיפות הינו 1.25. בגרף זה כפי שהיינו מצפים ניתן לראות מגמת שיפור ככל שמספר הדורות גדל בערך הטוב ביותר, ואילו בערך הממוצע המגמה קצת פחות ברורה. ניתן לראות כי בערך עד 100 דורות ישנה ירידה חדה בערך הטוב ביותר ולאחר מכן היא מתמתנת. הסיבה לכך היא שככל הנראה לאחר 100 דורות ערך הפריט הטוב ביותר מגיע לערך "טוב" יחסית שמפריט זה כבר קשה ליצור פריטים עם ערכים טובים בהרבה ממנו. ערך הפריט הטוב ביותר בכול הדורות הינו 12.08 והוא מתקבל בערך בדור 330. בנוסף ניתן לראות כי עדיין אין אזורים שטוחים בגרף לאורך כל 350 הדורות ולכן יכול להיות שאם היינו מגדילים את כמות הדורות התוצאה הייתה משתפרת, אך אנו סבורים שתוצאה זו מספיקה ומשוקלים של זמן ריצה החלטנו להישאר על מספר איטרציות זה.

Number of Halls with Students Chairs vs Number of Halls with Regular Chairs in Uneven Exams



בגרף זה ניתן לראות את כל המבחנים שלא כל האולמות שמשובצים אליהם הם בעלי סוג כיסאות זהה ואת היחס בין אולמות עם כיסאות רגילים לאולמות עם כיסאות סטודנט במבחנים אלו. ניתן לראות כי בשיבוץ זה יש מעט מאוד מבחנים כאלו, 3 מתוך 48 מבחנים. נבחין כי בקורס תורת ההסתברות ושימושיה ישנה חלוקה של אולם אחד עם כיסאות סטודנט ואולם אחד עם כיסאות רגילים שזו מבחינתנו יחס החלוקה הגרוע ביותר, מכיוון שהוא מציג חלוקה לא הוגנת אך מכיוון שמספר האולמות קטן יחס זה זניח באופן יחסי לכלל השיבוץ.

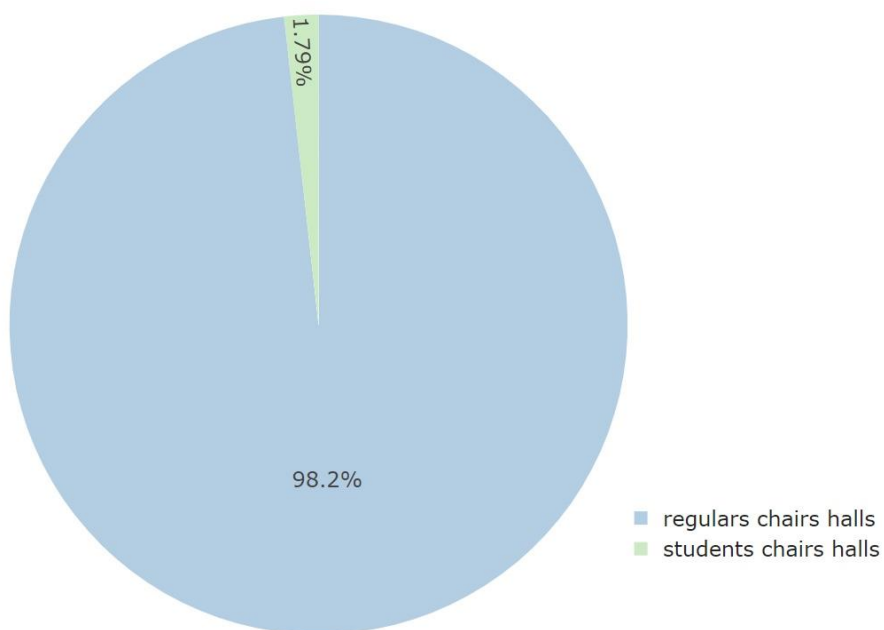
Number of Exams with both Students Chairs and Regular Chairs



בגרף זה ניתן לראות את אחוז המבחנים שבהם משובצים אולמות עם כיסאות שונים לעומת אחוז המבחנים שכל האולמות שמשובצים אליהם בעלי אותם סוגי כיסאות. ניתן לראות כי קרוב ל-10% מהאולמות לא בעלי אותם סוגי כיסאות בשיבוץ זה.

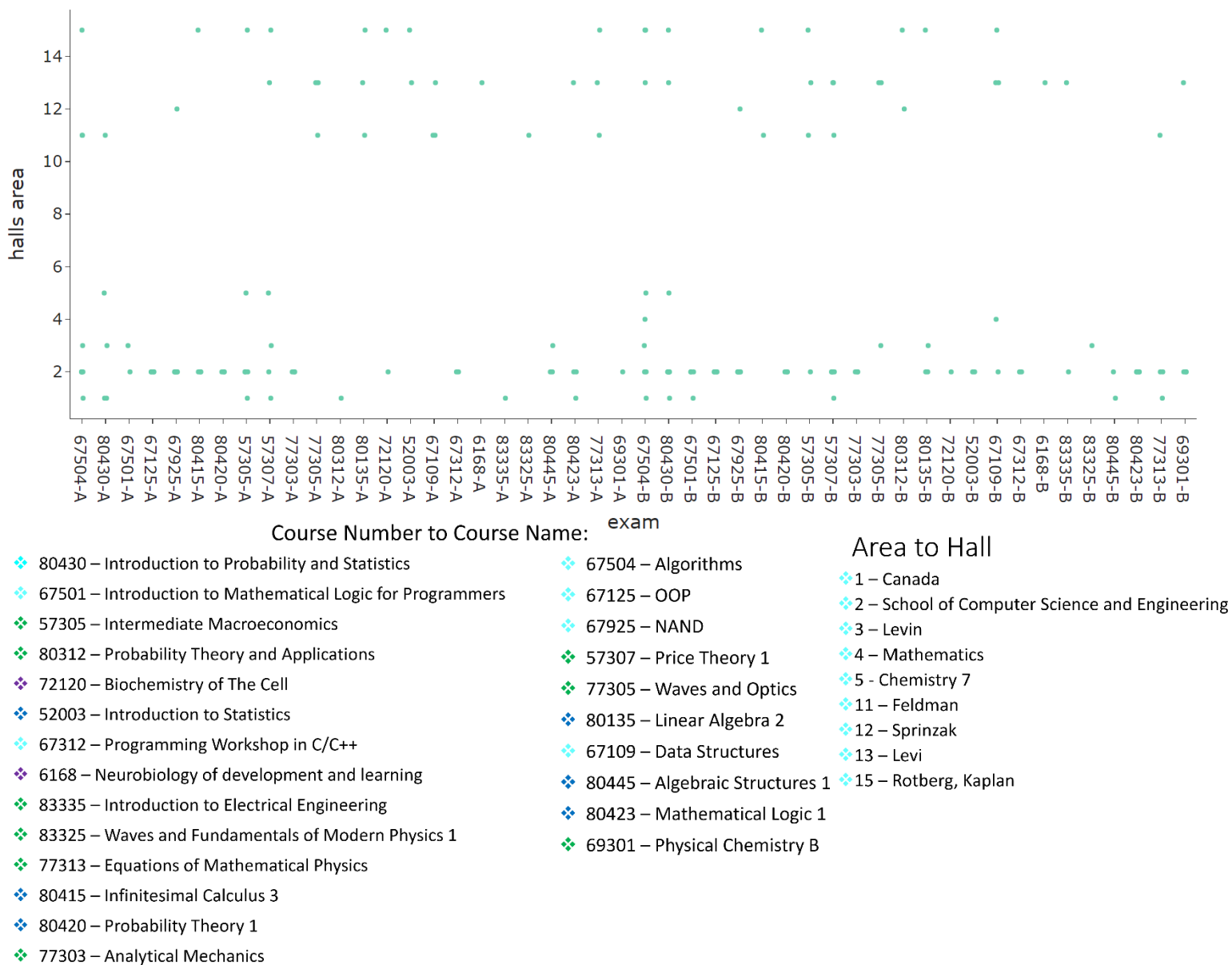
שני הגרפים בעמוד זה אמורים לשקף את הוגנות השיבוץ עבור הסטודנטים.

Number of Halls with Students Chairs vs Number of Halls with Regular Chairs



בגרף זה ניתן לראות את אחוז האולמות עם כיסאות רגילים לעומת אחוז האולמות עם כיסאות סטודנט בשיבוץ. גרף זה אמור לייצג את רמת הנוחות של הסטודנטים בעת ההיבחות.

Distance between Halls that were Assigned to a Certain Exam



בגרף זה ניתן לראות את ההתפרשות של המבחנים על אזורים שונים באוניברסיטה כאשר ציר ה-א הינו כל המבחנים שאנו מוצאים להם שיבוץ בבעיה (מועדי א וב) ואילו ציר ה-ב הינו האזורים באוניברסיטה כאשר חילקנו את מספרי האזורים לפי קרבה פיזית בקמפוס (למשל ליון ומתמטיקה קרובים ולכן מספרי האזור שלהם קרובים) הנקודות הכחולות מציינות את האולמות ששובצו למבחן בטור שלהן. למשל עבור הקורס לוגיקה למתמטיקאים מועד ב ניתן לראות כי האולמות שלו קרובים יחסית וכולם משובצים באזורים בין 2 ל 1 לעומת האולמות של מועד ב בהסתברות מפוזרים לאזורים רבים בקמפוס. גרף זה בא להראות כמה הפתרון מצליח לשמור על האילוץ הקל שגורף כי כל האולמות המשובצים למבחן מסוים צריכים להיות באותו אזור. ומטרתו היא בעיקר נוחות למתרגלים שלא יצטרכו להתרוצץ בין האולמות השונים במהלך המבחן.

השוואה בין Gradient Descent ל Genetic Algorithm בבעיית שיבוץ האולמות:

השוואה בין ערכי פונקציית הערך שקבענו (עמוד 13) באלגוריתמים השונים:
ראשית נבחין כי GD רץ בערך 80 איטרציות אל מול 150 שניתנו לו ומגיע לערך של 15.32 לעומתו GA ממשיך להראות מגמת ירידה גם לאחר 350 דורות ומגיע לערך נמוך יותר של 12.08.

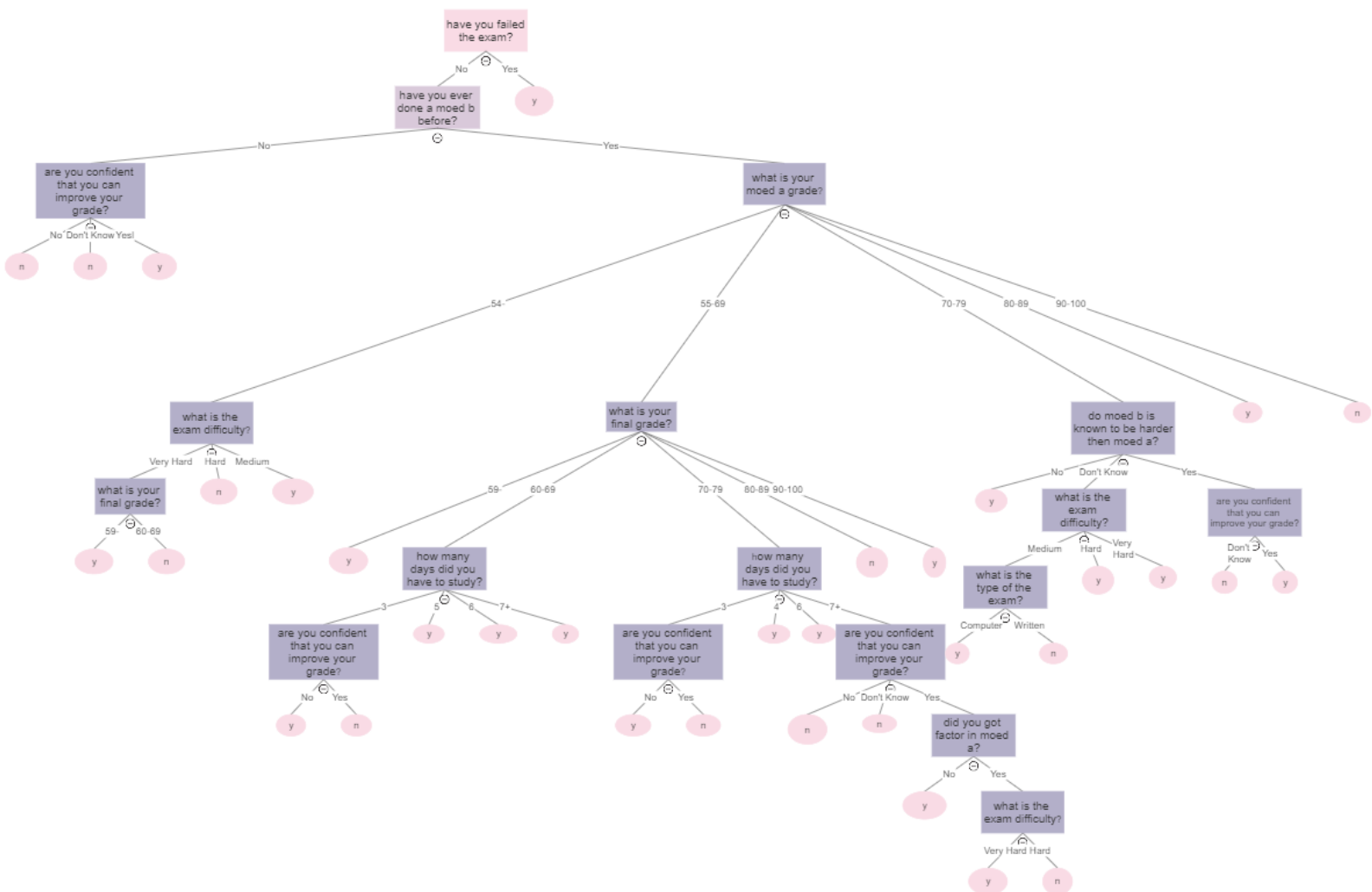
השוואה בין המבחנים שסוגי הכיסאות באולמות שלהם לא זהים בשיבוץ האופטימאלי:
ניתן לראות כי עבור GD מספר המבחנים בהם יש שיבוץ שך אולמות בעלי כיסאות שונים הינו 8 לעומת שלושה בשיבוץ של GA. בנוסף עבור GD במבחנים רבים יותר היחס מתקרב לכך שחצי מהאולמות עם כיסאות רגילים ומחציתם עם כיסאות סטודנט דבר המאפיין שיבוץ שאנו רואים כלא הוגן. מכאן נראה שבקריטריון זה ID של GA היא על העליונה.

השוואה בין אחוז האולמות בעלי כיסאות הסטודנט מכלל האולמות בשיבוץ :
ניתן לראות כי בGD בערך 6% מהאולמות הם אולמות עם כיסאות סטודנט ואילו בGA בערך 2% מהאולמות הם אולמות עם כיסאות סטודנט. שתי התוצאות טובות אך גם כאן GA מגיע לתוצאה טובה יותר.

השוואה בין שיבוץ מבחנים לאולמות קרובים:
גרף זה מעט קשה להשוואה אך נראה כי כאן דווקא GD מצליח יותר מGA, אך תוצאותיהן דומות סך הכול.

השוואה כללית בין האלגוריתמים:
כאן זמני הריצה של שני האלגוריתמים דומה והוא זניח לעומת הזמן שלוקח לבעיית שיבוץ המבחנים. ניתן לראות כי באופן כללי GA מוציא תוצאות טובות יותר מGD גם בבעיה זו, עם סייג שGD מצליח מעט יותר בשיבוץ של אולמות באזורים קרובים לאותו המבחן. חשוב לציין כי הקלט של שני האלגוריתמים שונה ומכיוון שGA מצליח יותר בבעיה הקודמת זה יכול להשפיע על ההצלחה שלו בבעיה זו, אך אנו רואים בבעיית שיבוץ האולמות הרחבה של בעיית שיבוץ המבחנים ולכן אנו מעוניינים בהשוואת הפתרונות הכוללים ולא בבדיקה הפרטנית עבור בעיה זו.

תוצאות היועץ ID3:



זהו העץ שנוצר על ידי אלגוריתם ID3 בהינתן המידע שאספנו מסטודנטים באוניברסיטה העברית. ניתן לראות כי חלק מהבחירות בעץ לא הגיוניות למשל הסוכן שלנו ימליץ לסטודנט שקיבל במבחן בין 55-69 ובציון הסופי בין 90-100 לגשת למועד ב אך אם אותם נתונים רק עם ציון סופי של 80-89 הוא ימליץ לו לא לגשת למועד ב. בנוסף באחד הענפים ניתן לראות כי הסוכן שלנו ממליץ לסטודנט שלא בטוח שהוא יכול לשפר לגשת למועד ב אך לסטודנט שבטוח שהוא יכול לשפר לא לגשת. כל הבחירות האלו שאנו לא רואים כהגיוניות כנראה הגיעו מכך שקבוצת המדגם של סטודנטים ששאלנו קטנה יחסית וכי חלק מהמידע לא מדויק או מהימן. ככל הנראה אם היה לנו זמן רב יותר לאיסוף המידע וניקוי שלו יכול להיות שטעויות מסוג זה היו פחות נפוצות.

נתבונן ב 3 דוגמאות הרצה של האלגוריתם:

החזרת תשובה חיובית:

```
have you failed the exam? (yes, no)
no
have you done moed b before? (yes, no)
yes
what is your exam grade? (54-, 55-69, 70-79, 80-89, 90-100)
55-69
what is your final grade? (59-, 60-69, 70-79, 80-89, 90-100)
90-100
you should take the moed b exam
```

החזרת תשובה שלילית:

```
have you failed the exam? (yes, no)
no
have you done moed b before? (yes, no)
no
are you confident that you can improve the exam score?(yes, no, don't know)
no
you shouldn't take the moed b exam
```

תשובה כאשר אין באפשרות האלגוריתם להכריע את הבעיה:

```
have you failed the exam? (yes, no)
no
have you done moed b before? (yes, no)
yes
what is your exam grade? (54-, 55-69, 70-79, 80-89, 90-100)
54-
what is the exam difficulty? (easy, medium, hard, very hard)
easy
looks like you are the first one to face this condition
we'll be happy if you'll help us expand our data and fill this form to let us know what you have chosen
https://docs.google.com/forms/d/e/1FAIpQLSfDaM7w2vvG9KHpyixrz2HfPhZx\_0n01T5RpRPNC-H2UQtFAg/viewform?usp=sf\_link
```

דיון בתוצאות והסקת מסקנות:

חלק ראשון בעיית שיבוץ המבחנים:

בעיית שיבוץ המבחנים למשבצות זמן:

ניתן לראות מהתוצאות כי האלגוריתם SA כמו שמימשנו אותה לא מתאים לבעיה מכיוון שהוא מוצא מינימום מקומי שאינו מתקרב למינימום הגלובאלי (לפחות רחוק בהרבה מהפתרונות של שאר האלגוריתמים). בנוסף ניתן לראות כי ניסיון השיפור של GD כך שבכול פעם שהוא ניתקע הוא ינסה להתחיל מנקודה שונה במרחב לא עזר עבור בעיה זו. הסבר אפשרי לכך הוא שרק נקודות מעטות במרחב הבעיה שהגדרנו מובילות למינימום הגלובאלי, ולכן כדי להגיע אליהן האלגוריתם יצטרך להגריל נקודות התחלה מספר רב של פעמים דבר שיהפוך את פתרון הבעיה ללא ישים עקב זמן ריצה ארוך מדי. בנוסף ניתן לראות כי GA מגיע לתוצאות קצת יותר טובות מGD מבחינה ערך ההחזרה קשה להגיד איזה מהאלגוריתמים טוב יותר מכיוון שהערכים מאוד קרובים. מבחינת ביצועים GD רץ כשעה ונעצר לפני שהוא מגיע למספר האיטרציות המקסימאלי שניתן לו, לעומתו GA רץ כחצי שעה ומגרף הערכים שלו ניתן להסיק כי לו היינו נותנים לו לרוץ למספר דורות גדול יותר היינו יכולים לקבל ערך טוב יותר ולכן אנו סבורים כי לפי המימושים שהצגנו והגדרת הבעיה שלנו GA הינו האלגוריתם הטוב יותר לפתרון הבעיה.

בנוסף מתוצאות הדוגמא של תורת המשחקים ניתן לראות כי מפתרון בצורה זו אנו משיגים אופציה של חוגים שונים לשבץ מבחן שיהיה טוב בהם, ובנוסף אנו מקבלים התחשבות בנקודות זכות של המבחנים כפקטור בבחירת ההפרש בין מבחנים שונים, דברים שאנו לא מקבלים בבעיית שיבוץ האולמות באמצעות אלגוריתמי החיפוש.

מהצד השני אנו באמצעות שיבוץ של אלגוריתמי החיפוש אנו מקבלים שיבוץ נטרלי יותר שלא נתון לשיקולים של חוגים מסוימים על פני אחרים ולא מושפע מגודל הקורס בבחירת שיבוץ אופטימאלי. יתרון נוסף של אלגוריתמי החיפוש הינו שבאמצעותם ניתן לפתור את בעיית שיבוץ האולמות גם עבור דוגמאות מציאותיות יותר שכן גם מספר המבחנים וגם הגדרת תקופת הבחינות משפיעים על מרחב החיפוש שלנו, ולכן עבור תקופה של חודש וחצי עם 48 מבחנים יהיה קשה מאוד להכריע פתרון בתורת המשחקים דבר שאלגוריתמי החיפוש מצליחים לעשות בחצי שעה עד שעה.

בעיית שיבוץ המבחנים לאולמות:

בבעיה זו בחרנו להתמקד בשני אלגוריתמים GA ו GD בבעיה זו קצת יותר קשה לקבוע מי האלגוריתם הטוב יותר מכיוון ששיבוץ האולמות תלוי בשיבוץ המבחנים שבו ראינו כי GA מצליח יותר. אך מכיוון שאין הבדל רב בין התוצאות המספריות בשיבוץ המבחנים אנו סבורים כי ניתן להסיק מהתוצאות שראינו כי GA באופן כללי מחזיר פתרון טוב יותר מGD גם בבעיה זו. זמן הריצה של שיבוץ האולמות בשני האלגוריתמים הוא זניח יחסית, אך כאן ההבדל בין התוצאות הוא מעט משמעותי יותר ובנוסף מתוך 150 איטרציות שניתנו לGD הוא נעצר אחרי 80 בערך בפתרון שהוא פחות טוב מהפתרון של GA שלפי הגרף שלו ניתן להסיק כי יש אפשרות לשיפור תוך כדי הגדלת כמות האיטרציות. לסיכום פסקה זו גם כאן ראינו כי GA הינו האלגוריתם הטוב יותר, זאת כנראה בשל הסוכסטייות שלו המאפשרת לו לברוח ממנימום מקומי.

מסקנות בעיית השיבוץ:

ניתן לראות כי GA היווה את הפתרון הטוב ביותר לשיבוץ באופן כללי, זאת כלל הנראה בשל האקראיות הרבה שלו, שכן לעומת RGD אנו מאפשרים לו הרבה צעדים אקראיים שיכולים לפגוע בטיב הפתרון, וזה ככל הנראה מאפשר לו להגיע למסלול שיוביל אותו בסופו של דבר למינימום הגלובאלי בעוד ש RGD, GD לא מצליחים "לברוח" מהמינימום המקומי. לגבי SA, הדרך שמימשנו אותו לא מאפשרת לו לקחת צעדים רעים ולכן אין לו את החופש של GA והוא לא מצליח לצאת מהמינימום המקומי שאליו הוא מגיע.

חלק שני יעוץ לסטודנטים על ההחלטה לגבי מועד ב:

כפי שניתן לראות בתוצאות בחלק זה ישנם פעמים בהן תשובת היועץ לא תאמה את מה שנראה לנו בהחלטה החכמה במצב הנתון. פער זה נובע ככל הנראה מכמות הדגימות הקטנה היחסית שאספנו ומכך שלא התעמקנו בניקוי המידע שכן זה לא היה הדגש העיקרי של הפרויקט. בכול זאת במספר לא זניח של המצבים הסוכן שלנו מצליח לענות תשובות שעולות בקנה מידה אחד עם ההיגיון שלנו.

בנוסף, מתוצאות העץ וסדר שאלת השאלות ניתן להסיק אילו מהנתונים שאספנו הם רלוונטיים יותר להחלטת הסטודנט האם לגשת למועד ב, כאשר השאלה המובילה כצפוי הינה האם ניגשת למועד ב, ולאחריה נראה שלנתון של האם ניגשת למועד ב בעבר יש גם השפעה רבה. כלל הנראה סטודנטים שניגשו למועד ב בעבר מרגישים יותר ביטחון בלגשת למועד ב בשנית. לעומת נתונים אלו שאלות כמו מה צורת ההיבחנות? ומהו סוג הבחינה? פחות רלוונטיות בהחלטה של הסטודנט על הסוגית מועד הב.

לסיכום, על אף שלא לגוריתם שלנו יש עוד הרבה מה ללמוד כדי שהוא יוכל לייעץ לסטודנט האם להיבחן במועד ב, (אבל אנחנו מאמינים שהוא יציל את הסטודנטים מהדילמה הזאת) ניתן ללמוד ממנו דברים כמו איזה קריטריונים חשובים בהחלטה האם לקחת לגשת למועד ב.

סיכום והצעות לשיפור:

סיכום:

שיבוץ מבחנים למשבצות זמן:

- Pure CSP/WCSP לא ישימים לפתרון הבעיה כמו שהגדרנו אותה
- Simulated Annealing כמו שמימשנו אותו לא מחזיר תוצאות מספיק טובות לבעיה.
- Random Gradient Descent משיג תוצאות דומות ל Gradient Descent כפי שמימשנו אותם וכפי שהגדרנו את הבעיה ולכן אין בו תועלת רבה בפרויקט זה.
- Gradient Descent כמו שמימשנו הינו שיטה טובה לפתרון הבעיה כפי שהגדרנו אותה, אך הוא עדיין נתקע במינימום מקומי וגם יקר בזמן ריצה.
- Genetic Algorithm הוא הפתרון היעיל ביותר מבין האלגוריתם שמימשנו לבעיה זו ככל הנראה בשל האקראיות הרבה שלו המאפשרת לו להתקרב למינימום הגלובאלי.
- תורת המשחקים מאפשרת לנו צורת הסתכלות הנותנת לחוגים לשבץ לוח מבחנים שטוב להם לעומת אלגוריתמי החיפוש שמציגים גישה נטרלת יותר.

שיבוץ מבחנים לאולמות:

- Gradient Descent כמו שמימשנו הינו שיטה טובה לפתרון הבעיה כפי שהגדרנו אותה, אך הוא עדיין נתקע במינימום מקומי.
- Genetic Algorithm הוא הפתרון היעיל ביותר מבין האלגוריתם שמימשנו לבעיה זו ככל הנראה בשל האקראיות הרבה שלו המאפשרת לו להתקרב למינימום הגלובאלי.

באופן כללי בבעיית השיבוץ תוך כדי שימוש באלגוריתמי חיפוש ויוריסטיקות הצלחנו להגיע לפתרונות טובים עבור בעיות שהן NP-Hard בזמן סביר, שזוהי תוצאה מרשימה בעינינו.

יעוץ לסטודנטים על ההחלטה לגבי מועד ב:

- עץ ההחלטה הסופי הושפע מטיב המידע שאספנו.
- באמצעות שימוש באלגוריתם ID3 ניתן היה לראות אילו מהנתונים לוקחים חלק גדול יותר בהחלטה של סטודנט האם לגשת למועד ב ואלו לא.

הצעות לשיפור:

שיבוץ מבחנים למשבצות זמן:

- ניתן לנסות יוריסטיקות נוספות בפתרון הבעיה של Pure CSP.
- ניתן לנסות פונקציית קירור שדועכת בקצב איטי יותר ב Simulated Annealing.
- ניתן לנסות לקבוע את ההפרשים לפי נקודות זכות ולא לפי חוגים.
- ניתן לנסות להגדיל את מספר הדורות ולשנות את hyper parameters האחרים של התוכנית ולראות אם מגיעים לתוצאות טובות יותר.
- בנוסף במסגרת הפרויקט לא הצלחנו להשיג מידע אמיתי על המבחנים והעדפות על השיבוצים על כן ניתן לנסות לעבוד עם מידע אמיתי ולראות אם עדיין מגיעים לפתרונות טובים.

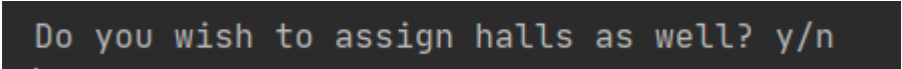
שיבוץ מבחנים לאולמות:

- ניתן להריץ את שני האלגוריתמים עם לוח מבחנים זהה כדי להשוות את התוצאות שלהם ללא תלות בשיבוץ המבחנים שלהם.
- ניתן לנסות הגבלות אחרות בפונקציית הערך כמו התאמה למגבלות קורונה
- ניתן לנסות להגדיל את מספר הדורות ולשנות את hyper parameters האחרים של התוכנית ולראות אם מגיעים לתוצאות טובות יותר.
- בנוסף במסגרת הפרויקט לא הצלחנו להשיג מידע אמיתי על המבחנים והעדפות על השיבוצים על כן ניתן לנסות לעבוד עם מידע אמיתי ולראות אם עדיין מגיעים לפתרונות טובים.

יעוץ לסטודנטים על ההחלטה לגבי מועד ב:

- ניתן לאסוף מידע רב יותר ולנקות אותו בצורה מקיפה יותר
- ניתן לשאוב השראה מתוך התוצאות אילו שאלות יותר רלוונטיות ומתוכן לייצר שאלון חדש

הוראות תפעול התוכנית

- נחלץ את הפרויקט מקובץ הzip.
- נתקין את כל החבילות הנדרשות מקובץ requirements.txt בעזרת פקוד הבאה:
pip install -r requirements.txt
- על מנת להריץ את האלגוריתמים **CSP** ו-**WCSP** נריץ את הקובץ `SolvePureCSP.py`. נריץ אותו עם הפרמטרים הבאים:
 1. סוג האלגוריתם
 2. סוג היוריסטיקה (עבור CSP בלבד)
 3. תאריך תחילת תקופת המבחנים
 4. תאריך סיום תקופת המבחניםהיוריסטיקות האפשריות הן:
`BACKTRACKING = backtracking`
`MINIMUM_REMAINING_VARS = mrv`
`DEGREE = d`
`LEAST_CONSTRAINING_VALUE = lcv`
`LEAST_CONSTRAINING_VALUE_AND_MINIMUM_REMAINING_VARS = combined`
לדוגמא, על מנת לפתור את הבעיה ע"י שימוש בCSP עם היוריסטיקה LCV נריץ משורש הפרויקט את הפקודה:
python SolvePureCSP.py csp lcv 2022/01/15 2022/03/08
לעומתו, על מנת לפתור את הבעיה ע"י שימוש בWCSP נריץ משורש הפרויקט את הפקודה:
python SolvePureCSP.py wcsp 2022/01/15 2022/03/08
- ניתן לראות את התוצאות של האלגוריתם
- על מנת להריץ את האלגוריתמים **GA, GD, RGD, SA** נריץ את הקובץ `ISASolver.py`. נריץ אותו עם אותם פרמטרים כפי שמפורט לעיל מלבד היוריסטיקה.
לדוגמא, על מנת לפתור את הבעיה ע"י שימוש בGA נריץ את הפקודה:
python ISASolver.py ga 2022/01/15 2022/03/08
באופן דומה, על מנת לפתור את הבעיה ע"י שימוש בGD נריץ משורש הפרויקט את הפקודה:
python ISASolver.py gd 2022/01/15 2022/03/08
באותו אופן מריצים את הפקודות עבור `sa`, `rgd`.
עבור הרצה של `gai gd` נוכל לבחור האם להרחיב את הבעיה או לא, לאחר סיום השיבוץ תופיע לנו ההודעה הבאה:

במידה ונרצה להרחיב ולשבץ גם אולמות את הבעיה נסמן y אחרת נסמן n.

- נוכל לראות מדדים להצלחת הפתרון של האלגוריתם בטרמינל, בנוסף ניתן לראות את שיבוץ המבחנים בGoogle Calendar בצורה מאוד נוחה ונגישה.
לאחר השיבוץ בGoogle Calendar תופיע ההודעה הזו:

Do you want to save the calendar? Please insert y/n

לחיצה על y תשמור את השינויים בלוח ולחיצה על n תמחק אותם. אנו ממליצים למחוק את השינויים לאחר ההתבוננות כדי שלא יהיה צורך לעשות זאת ידנית שכן זהו תהליך מתיש.
הגישה ליומן גוגל הינה בעזרת שם משתמש וסיסמה הבאים:

User name: aicalenderproject@gmail.com

Password: AIP12345!

כל שבועיים צריך לחדש את הרישיון ללוח, הגשנו את הקובץ המחודש, אך אם עברו יותר משבועיים ניתן להריץ את הקוד:

```
# scopes = ["https://www.googleapis.com/auth/calendar"]
# flow = InstalledAppFlow.from_client_secrets_file("../Utils/client_secret.json", scopes=scopes)
# credentials = flow.run_console()
# pickle.dump(credentials, open("../Utils/token.pkl", "wb"))
# credentials = pickle.load(open("../Utils/token.pkl", "rb"))
# service = build("calendar", "v3", credentials=credentials)
# result = service.calendarList().list().execute()
# export_to_calendar(courses, answer)
```

שנמצא מעל כל קריאה לcalendar בקובץ ISASolver.py בהערה.

- על מנת להריץ את הגרפים נריץ את הקובץ CreateGraphs.py נריץ אותו עם אותם הפרמטרים של הקובץ ISASolver.py
למשל כדי לייצר את הגרפים עבור GA נריץ את הפקודה:
python CreateGraphs.py gd 2022/01/15 2022/03/08

- על מנת להריץ את האלגוריתם ID3 נריץ את הקובץ ID3.py אשר נמצא בתיקיית ID3 בעזרת הפקודה:

python ID3/ID3.py

- נענה כן או לא על השאלות בהתאם. לדוגמא,

```
have you failed the exam? (yes, no)
yes
you should take the moed b exam
```

- בסוף נקבל עצה אם כדאי לגשת למועד ב או לא
הערה: כשאר מריצים מהטרמינל הוא משום מה לא מדפיס את הצבעים :),
על מנת לקבל את מלוא החוויה עם הצבעים יש להריץ דרך PyCharm 😊

רשימת קבצים של התוכנית:



תמונות ונספחים:

תמונה של לוח השנה ששיבצנו:

שבת 5	יום ד' 4	יום ה' 3	יום ד' 2 ● Introduction to Ele. אח. 1:30	יום ג' 1 בפברואר	יום ב' 31 ● Physical Chemistry B. לפ. 9 ● NAND - A. אח. 1:30	יום א' 30
12	11 ● Algorithms - A. לפ. 9	10 ● Equations of Mathem. לפ. 9	9	8	7 ● Waves and Fundan. אח. 1:30	6 ● Algebraic Structures 1 - לפ. 9 ● OOP - A. לפ. 9 ● Neurobiology of de. אח. 1:30
19	18 ● Neurobiology of devel. לפ. 9	17 ● Probabiloty Theory an. לפ. 9 ● Programming Worksh. לפ. 9 ● Biochemistry of Th. אח. 1:30 ● Intermediate Macr. אח. 1:30	16 ● Equations of Math. אח. 1:30	15	14	13 ● Mathematchal Logic - לפ. 9
26	25	24	23 ● Waves and Optics - B. לפ. 9 ● Introduction to Pro. אח. 1:30	22 ● Introduction to Statist. לפ. 9	21 ● Probabiloty Theory 1 - לפ. 9	20 ● Physical Chemistry B - לפ. 9 ● Price Theory 1 - B. אח. 1:30
5	4 ● Waves and Fundamen. לפ. 9	3 ● Analytical Mechanics. לפ. 9 ● Introduction to Mathei. לפ. 9 ● Linear Algebra 2 - B. לפ. 9 ● Infinitesimal Calcul. אח. 1:30	2	1 במרץ	28 ● Data Structures - B. לפ. 9	27 ● Algebraic Structures 1 - לפ. 9 ● OOP - B. לפ. 9

תמונה מקורבת של לוח שנה משובץ כך שפתחנו מבחן כדי להראות את שיבוץ האולמות בנוסף לשיבוץ המבחנים:

יום ה' 3 ● Intermediate Macr. אח. 1:30	יום ד' 2 ● Probabiloty Theory. אח. 1:30	יום ג' 1 בפברואר ● Analytical Mechanics. לפ. 9	יום ב' 31 ● Biochemistry of The C. לפ. 9	יום א' 30 ● Data Structures - A. אח. 1:30 ● Price Theory 1 - A. אח. 1:30
--	---	--	--	---

Data Structures - A

יום ראשון, 30 בינואר 2022 - 1:30 - 4:30 אח.

Feldman A, Feldman B, Kaplan

Exam

30 דקות קודם