מסרכת בינה מלאכותית לתכנון הפמנת ארוחה קבוצתית המשקלת את צרכיהם של כל חברי הקבוצה

אסף שול (207042714)
ברק רמני (204294417)
דוד שניידר (203380746)
ניצן ברזילי (315359265)
רועי שוסברגר (203381801)
קורס "מבוא לבינה מלאכותית"
האוניברסיטה העברית

תוכן עניינים

תקציר

בימינו, הזמנת אוכל משירותי משלוחים הפכה לתופעה נפוצה ויומיומית. תופעה זו הביאה איתה בעיה חדשה - כיוון שישנו שפע עצום של מאות מסעדות זמינות בכל רגע נתון, הנסיון לבחור מסעדה אחת שתתאים לצרכיהם של כל הסועדים בקבוצת חברים או עובדים הפך לאתגר גדול. מעבר לאילוצים של הסועדים (כגון כשרות, צמחונות והעדפות קולינריות), ישנם אילוצים גם מצד המסעדה (כגון שעות פתיחה ומינימום הזמנה). לכן, הבעיה אותה אנחנו מנסים לפתור היא תכנון הזמנת אוכל תוך שקלול העדפותיהם של חברי קבוצת הסועדים ועמידה בהגבלות המסעדה.

בפרויקט זה אנו מתייחסים לבעיה זו בתור בעית חיפוש במרחב חיפוש המכיל את כל הפתרונות האפשריים, ובוחנים שישה אלגוריתמי חיפוש אשר מוצאים פתרונות הולמים לבעיה במהירות מרשימה.

פרק א' – תיאור הבעיה

קלט הבעיה

הקלט מורכב משני מרכיבים:

- dataset המכיל מידע על מסעדות ועל המנות בתפריט של כל אחת מהן. לפרטים בנוגע לאופן בו הדאטא נאסף ועובד, פנו לנספח ב'.
- עבור כל אחד מהסועדים בקבוצה, רשימת העדפות המתקבלת כקלט מוגדרת באופן הבא:

	אילוץ	**ערכים אפשריים**
אילוצים על המסעדה	כשרות	כשר / לא משנה
	מינימום דירוג	ערך מספרי בין 1 ל-10
	מידת רעב	גבוהה / נמוכה
	סגנונות	בחירה מרובה מבין 65 סגנונות אפשריים
אילוצים על המנה	צמחונות	צמחוני / לא משנה
	נטול גלוטן	נטול גלוטן / לא משנה
	ללא אלכוהול	נטול אלכוהול / לא משנה
	חריפות	חריף / לא חריף / לא משנה
	מגבלת מחיר	מספר בשקלים המייצג את המחיר המקסימלי למנה כולל משלוח

כל אחד מהעדפות הסועדים הומרה לפרמטר אחד או יותר המייצגים אילוץ אשר הפתרון נדרש לקיים או לתעדף. חלק מהאילוצים הנם אילוצים קשים (אשר חייבים להתקיים, כלומר נפסול פתרונות שלא מקיימים אותם) וחלקם הם אילוצים גמישים (אשר לא חייבים להתקיים, אך נעדיף פתרונות שמקיימים אותם). לפירוט בנוגע להמרת העדפות הסועדים לאילוצים, ראו נספח ב' – עיבוד ואיסוף הדאטא.

הנחות מקדימות

בכדי להגדיר מרחב חיפוש אשר הנו קטן מספיק כדי להיות פתיר בזמן ריאלי (באמצעות אלגוריתם חיפוש חכם) וגדול מספיק כדי להיות מאתגר לפתרון, ביצענו את ההנחות הבאות:

- **הגבלת מספר הסועדים בצוות:** כיוון שגודלו של מרחב החיפוש מושפע באופן ישיר ומשמעותי ממספר הסועדים בקבוצה, החלטנו לצמצם את הבעיה כך שהקלט יכיל אילוצים של שלושה סועדים בדיוק. לפירוט בנוגע לבחירה במספר הסועדים ראו נספח ג' – תיאור מרחב הפתרונות.

- **צמצום גאוגרפי של מסעדות:** החלטנו לצמצם את הבעיה כך שהקלט יכיל מסעדות מאזור ת״א בלבד (לפירוט ראו נספח ב׳ – איסוף ועיבוד הדאטא).
- **חלוקת דמי המשלוח:** הנחנו כי דמי המשלוח שהמסעדה גובה מתחלקים באופן שוויוני בין הסועדים בקבוצה.
- **הגדרת מנה עיקרית :**כיוון שמטרת הבעיה היא למצוא מנה עיקרית עבור כל אחד מהסועדים בקבוצה, הצטרכנו להגדיר מהי מנה עיקרית. כיוון שהדאטא לא מכיל חלוקה מובנית בין מנות עיקריות לסוגים אחרים של מנות ((לפירוט ראו נספח ב׳ – איסוף ועיבוד הדאטא), הגדרנו מנה עיקרית בתור מנה שמחירה 30 ש״ח לפחות.

פלט הבעיה

רשימה המכילה שם של מסעדה r, ושלוש מנות m_1, n_2, m_3 מתוך התפריט שהמסעדה מציעה, אשר עונות באופן מיטבי על צרכי הסועדים (לפירוט על האופן בו הוגדר מענה מיטבי, פנו לפרק ב׳). ישנה חשיבות לסדר בו מופיעות המנות בפתרון, שכן כל מנה מיועדת להתאים להעדפותיו של סועד מסוים. כמו כן, נציין כי כיוון שבעולם האמיתי מדובר במצב נפוץ, בחרנו לאפשר לשני סועדים שונים בקבוצה להזמין את אותה המנה (כלומר לאפשר כפילויות של מנות בפתרון המוחזר).

פרק ב' – אלגוריתמים לפתרון הבעיה

<u>מדידת איכות הפתרון</u>

כאמור, מטרתם של האלגוריתמים אשר יוצגו בפרק זה היא לבחור שילוב של מסעדה ושלוש מנות מתוך התפריט שלה אשר עונות בצורה מיטבית על העדפות הסועדים. על מנת שנוכל להעריך את איכות הפתרונות של האלגוריתמים השונים, הזדקקנו לדרך שתאפשר לכמת את מידת ההתאמה של פתרון נתון – כלומר לתת לו ציון מספרי שמייצג עד כמה הוא עונה על צרכי שלושת הסועדים.

לשם כך, הגדרנו פרמטרים המייצגים את העמידה של המסעדה (7 פרמטרים) והמנות (18 פרמטרים = 3 סועדים * 6 פרמטרים לסועד). חלק מהפרמטרים מייצגים אילוצים קשים (מסומנים באדום) וחלקם מייצגים אילוצים גמישים (מסומנים בירוק). לפירוט על האופן בו הפרמטרים נוצרים והדאטא עליו הם מבוססים, ראו נספח ב' – איסוף ועיבוד הדאטא.

פרמטרים ברמת המסעדה:

- **O (פתוח)** – פרמטר בינארי המקבל ערך 1 אם המסעדה פתוחה ביום ההזמנה, ו-0 אחרת
- **M (מינימום הזמנה)** – פרמטר בינארי המקבל ערך 1 אם סכום שלושת המנות בפתרון גדול או שווה למינימום ההזמנה של המסעדה בפתרון, ו-0 אחרת.
- **K (כשר)** – פרמטר בינארי המקבל ערך 1 אם מידת הכשרות של המסעדה (כשרה / לא כשרה) תואמת את העדפות הסועדים, ו-0 אחרת.
- **D (עמידה בהעדפות המשלוח)** – פרמטר בינארי המקבל ערך 1 אם זמן המשלוח הצפוי מהמסעדה עומד בהעדפות הסועדים, ו-0 אחרת.
- **DT (זמן המשלוח)** – פרמטר רציף המכיל את זמן המשלוח בדקות (מקבל ערכים בטווח בין 10 ל-120).
- **RD (הפרש הדירוג)** – פרמטר רציף המייצג את ההפרש בין דירוג המסעדה לממוצע הדירוגים של הסועדים (מקבל ערכים בטווח שבין 9- ל9).
- **C (עמידה בהעדפות סגנונות המזון)** - פרמטר קטגורי המייצג את העמידה בהעדפות הסועדים בנוגע לסגנונות מזון (מקבל ערך מספרי שלם בין 0 ל-2 כולל).

פרמטרים ברמת המנה המיועדת לסועד מסוים – לכל סועד $i \in [1,2,3]$:

- **V_i (צמחוני)** – פרמטר בינארי המקבל ערך 1 אם המנה עומדת בהעדפות הצמחונות של הסועד, ו-0 אחרת.
- **G_i (נטול גלוטן)** - פרמטר בינארי המקבל ערך 1 אם המנה עומדת בהעדפות הגלוטן של הסועד, ו-0 אחרת.
- **A_i (נטול אלכוהול)** - פרמטר בינארי המקבל ערך 1 אם המנה עומדת בהעדפות האלכוהול של הסועד, ו-0 אחרת.
- **S_i (מידת חריפות)** - פרמטר בינארי המקבל ערך 1 אם המנה עומדת בהעדפות החריפות של הסועד, ו-0 אחרת.

- PH_i **(עמידה בהגבלת מחיר)** - פרמטר בינארי המקבל ערך 1 אם המנה עומדת בהגבלת המחיר של הסועד, ו-0 אחרת.
- PS_i **(הפרש מחיר)** – פרמטר רציף המכיל ערך מספרי אי שלילי המייצג את ההפרש בשקלים בין המחיר המקסימלי שהסועד היה מוכן לשלם לבין מחיר המנה שהותאמה לו בפתרון.

השתמשנו בפרמטרים הנ"ל כדי לייצר את פונקצית ה-Gain המוגדרת באופן הבא:

<u>Gain Function</u>

If $O * M * K * V_1 * V_2 * V_3 * G_1 * G_2 * G_3 * A_1 * A_2 * A_3 * PH_1 * PH_2 * PH_3 == 0$:
    Gain = 0
Else:
    Gain = $\frac{(D*DT)}{60} + RD + \frac{(PS_1+PS_2+PS_3)}{180} + \frac{C}{3} + \frac{(S_1+S_2+S_3)}{3}$
Return Gain

נפרט על השיקולים שהובילו לבניית פונקצית ה-Gain באופן הבא:

- ראשית, הפונקציה בודקת את כל האילוצים הקשים. כיוון שמדובר בפרמטרים בינאריים, מכפלתם תניב ערך 1 רק אם כולם מתקיימים, ותניב 0 במקרה בו לפחות אחד מהם אינו מתקיים. במקרה השני מדובר בפתרון שאינו תקין, ולכן פונקצית ה-Gain מחזירה ערך 0.
- במידה וכל האילוצים הקשים מתקיימים, הפונקציה מחזירה סכום של ערכיהם המנורמלים של כל הפרמטרים המייצגים אילוצים רכים, אשר נבחרו באופן הבא:
    - העמידה בהעדפות זמן המשלוח מיוצגת ע"י $\frac{(D*DT)}{60}$. המונה מתבסס על כפילת זמן המשלוח בדקות (DT) בפרמטר הבינארי D, המייצג את העדפות הסועדים בנוגע לזמן המשלוח (הנגזרות ממידת הרעב שלהם). את המכפלה הנ"ל חילקנו ב-60 על מנת לנרמל את הרכיב כולו לערכים בין 0 ל-1 (כיוון שהערך המקסימלי ש-DT יכול לקבל הוא 60 דקות).
    - העמידה בהעדפות זמן המשלוח מיוצגת ע"י RD, שמכיל את ההפרש בין דירוג המסעדה לדירוג הממוצע שדרשו הסועדים. כלומר, פרמטר זה מתגמל בערך חיובי מסעדות ש"עלו על הציפיות" ומעניש בערך שלילי מסעדות שלא עומדות ברף שהציבו הסועדים.
    - הרצון לתעדף מנות זולות מגולם ע"י הרכיב $\frac{(PS_1+PS_2+PS_3)}{180}$. החלוקה ב-180 מייצגת חלוקה ב-3 על מנת למצע את הפרשי המחירים, ואז חלוקה ב-60 כדי לנרמל את התוצאה לערכים בין 0 ל-1 ולמשקל את רכיב זה כראוי (ניסינו מספר משקולים שונים, אך כל משקול גבוה יותר של רכיב זה גרם לתעדוף יתר של מרכיב המחיר על חשבון פרמטרים אחרים).
    - העמידה בהעדפות סגנונות המזון מגולמת ברכיב $\frac{C}{3}$ 'העמדיה בהעדפות רמת החריפות של המנות מגולמת ברכיב $\frac{(S_1+S_2+S_3)}{3}$, ששניהם מוצעו ע"י חלוקה במספר הסועדים.

אלגוריתם נאיבי – יצירת Baseline

על מנת לייצר baseline, כלומר למצוא את הפתרון האופטימלי (המשיג את ציון ה-gain הגבוה ביותר מבין כל הפתרונות במרחב החיפוש), יצרנו אלגוריתם נאיבי. האלגוריתם הנאיבי עובר על כל הפתרונות

האפשריים במרחב החיפוש (קומבינציות של מסעדה ו-3 מנות מהתפריט שלה) ומחזיר את הפתרון בעל ציון ה-gain המקסימלי שזוהה לאורך ריצתו.

כדי שנוכל להעריך כמה פתרון מסוים (של אלגוריתם אחר) הוא טוב, רצינו להיות מסוגלים להשוות אותו לפתרון האופטימלי לא רק באופן מספרי (כלומר ע"י בחינת ההפרש בין ציוני ה-gain שלהם), אלא גם באופן יחסי. כלומר, רצינו לדעת עבור כל פתרון שהתקבל כמה פתרונות אפשריים מתוך מרחב הפתרונות מפרידים בינו לבין הפתרון האופטימלי. לשם כך, התאמנו את האלגוריתם הנאיבי כך שהוא שומר בקובץ יעודי כל פתרון אשר קיבל ציון gain חיובי ממש (כלומר גדול מאפס). כלומר, לאחר סיום ריצת האלגוריתם הנאיבי מתקבל לא רק הפתרון האופטימלי, אלא מיפוי של כל הפתרונות במרחב הפתרונות לציון ה-gain שלהם. לכן, באמצעות מיון הקובץ לפי ציון ה-gain ניתן לחשב עבור כל פתרון לאיזה אחוזון מתוך הציונים של מרחב הפתרונות הוא משתייך.

אלגוריתמי חיפוש

כיוון שהבעיה שבחרנו עוסקת במציאת פתרון מיטבי בתוך מרחב חיפוש עצום בגודלו, החלטנו למדל את הבעיה בתור בעיית חיפוש, ולבחון אלגוריתמים שונים שיאפשרו לחפש במרחב בצורה חכמה ויעילה.

<u>מידול הבעיה כבעיית חיפוש</u>

כדי למדל את הבעיה כבעיית חיפוש, הגדרנו את הדברים הבאים:

- <u>מצב</u>: קומבינציה של מסעדה ושל 3 מנות עיקריות מבין המנות שבתפריט שלה + האפשרות למנה ריקה.
- <u>מצב מטרה</u>: מצב אשר מכיל קומבינציה של מסעדה ושל 3 מנות עיקריות מהתפריט שלה, אשר מקיימות את כל האילוצים שהוגדרו כקשיחים (ראו תחילת הפרק).
- <u>גרף החיפוש</u>: גרף החיפוש הוא עץ בעומק 4, כאשר השכבה הראשונה בעץ מייצגת בחירה של מסעדה (כלומר המצבים בה מכילים מסעדה ו-3 מנות ריקות), ו-3 השכבות האחרונות מייצגות כל אחת בחירה של מנה עבור אחד מהסועדים (כלומר בכל שלב מוחלפת מנה ריקה במנה מתפריט המסעדה).
- <u>מעבר בין מצבים בגרף החיפוש</u>: בכל מעבר בין מצבים (ירידה של שכבה אחת במורד העץ מכיוון השורש לכיוון העלים), מתווסף אלמנט יחיד. כלומר, אם המעבר הוא ממצב השורש (מסעדה ריקה ו-3 מנות ריקות) לשכבה הראשונה $l = 0 \rightarrow l = 1$, המעבר יגרור החלפה של המסעדה הריקה בשם של מסעדה מהמאגר. אם המעבר הוא מהשכבה ה-$l \in [1,2,3]$ לשכבה ה-$l + 1$, הוא יכלול החלפה של המנה של הסועד ה-l ממנה ריקה למנה מתפריט המסעדה שנבחרה בשכבה ה-1.

על מנת לפתור את הבעיה, בחנו סוגים שונים של אלגוריתמי חיפוש:

<u>אלגוריתמי Uninformed Search</u>

ראשית, בחנו שני סוגי אלגוריתמי חיפוש לא-מונחה:

- **אלגוריתם DFS:** אלגוריתם אשר מחפש בעומק העץ עד להגעה למצב המטרה הראשון שהוא נתקל בו. כיוון שמרחב החיפוש שלנו הוא סופי, האלגוריתם בהכרח יסיים את ריצתו בזמן סופי, אך זמן הריצה שידרש לפתרון וכן ציון הפתרון שימצא תלוים בסדר המעבר על העץ. לכן, כדי להמנע ממצב

שבו אנחנו מחזירים תמיד פתרונות שמאותו אזור בעץ (האזור עליו עוברים ראשון אם סדר המעבר הוא קבוע), החלטנו לבחור בכל שלב מעבר באופן אקראי.

- **אלגוריתם UCS – Uniform Cost Search:** אלגוריתם המשתמש בפונקצית העלות בתור משקולות, ומוצא את מצב המטרה אשר הדרך אליו (כלומר סכום העלויות על בחירת המסעדה ובחירת 3 המנות) היה הנמוך ביותר.
  - <u>פונקצית עלות (cost)</u>: פונקצית העלות מוגדרת כך שהיא נותנת ציון למעבר מסוים (הוספת מסעדה או הוספת מנה). עבור כל סוגי המעברים, הפונקציה מוסיפה לציון מספר גדול עבור כל אילוץ קשיח שאינו מתקיים. אם המעבר מייצג בחירת מסעדה, הפונקציה מוסיפה בנוסף שקלול של האילוצים הרכים הקשורים למסעדה (סגנון המנות ודירוג המסעדה). אם המעבר מייצג בחירת מנה, הפונקציה מוסיפה בנוסף שקלול של האילוצים הרכים הקשורים למנה (הפרש המחירים ורמת החריפות).

<u>אלגוריתם Informed Search</u>

לאחר מכן, בחנו אלגוריתם נוסף המבצע חיפוש מונחה (יוריסטי):

- **אלגוריתם A*:** אלגוריתם המשתמש בפונקציית העלות בנוסף לשימוש בפונקצית יוריסטיקה, שמטרתה למנוע ממנו לבחור מסלולים העתידים להסתיים במבוי סתום (עלה בעץ שאינו מצב מטרה – לדוגמא, אם אחד האילוצים שלנו הוא מנה צמחונית, ובחרנו מסעדה שאין בתפריט שלה מנות צמחוניות). השתשמנו בפונקציית העלות שהוגדרה מעלה ובפונקציית יוריסטיקה שממקסמת את ערכי הפרמטרים המייצגים את האילוצים הקשים והאילוצים הרכים של הסועדים. לפירוט בנוגע לפונקציה זו ראו את הפונקציה rest_heuristic בקובץ localSearchAlgorithms.py.

<u>אלגוריתמי Local Search</u>

לבסוף, פנינו לאלגוריתמים לחיפוש לוקאלי. עבור שני האלגוריתמים הראשונים (Hill climbing, Simulated Annealing) הגדרנו פונקצית ערך (value) המקבלת מצב בעץ ומחזירה ציון מספרי המייצג את מידת העמידה שלו בהעדפות הסועדים (הן האילוצים הקשים והן האילוצים הרכים).

- <u>פונקציית הערך (value)</u>: פונקציה זו היא התאמה של פונקצית ה-gain לצרכי שני האלגוריתמים הנ"ל, תוך ביצוע שינוי ההכרחי להצלחתם. השינוי שבוצע היה התאמת הפונקציה כך שתהיה מונוטונית עולה ממש (כדי להמנע ממצב בו האלגוריתם יתקע בנקודת מקסימום לוקלית של פונקצית הערך). פונקצית הערך מוגדרך באופן הבא:

<u>Value Function</u>

Gain = $O + M + K + V_1 + V_2 + V_3 + G_1 + G_2 + G_3 + A_1 + A_2 + A_3 + PH_1 + PH_2 + PH_3$
If transition is restaurant choice:
    Gain += 10 * ($O + M + K$)
If transition is 1st meal choice:
    Gain += 10 * ($V_1 + G_1 + A_1 + PH_1$) + 1
If transition is 1st meal choice:
    Gain += 10 * ($V_2 + G_2 + A_2 + PH_2$) + 2
If transition is 1st meal choice:
    Gain += 10 * ($V_3 + G_3 + A_3 + PH_3$) + 3
Return Gain

בחנו שלושה אלגוריתמי חיפוש לוקאלי:

- **אלגוריתם Hill climbing:** אלגוריתם חיפוש לוקלי המחפש נקודת מקסימום מקומי של פונקצית הערך תוך שימוש בתור קדימויות.
- **אלגוריתם Simulated Annealing:** אלגוריתם חיפוש לוקלי בעל אלמנט רנדומי, המתבטא בבחירה אקראית של מצב ההתחלה של האלגוריתם, או בבחירה אקראית של מעבר בין מצבים בעץ החיפוש. האלגוריתם מחזיר תשובה כאשר הוא משיג ערך value מקסימלי (כלומר הגעה לנקודת מקסימום של פונקצית ה-value, העשויה להיות לוקלית) העומדת באילוצים הקשיחים.
- **אלגוריתם גנטי:** העיקרון שבבסיסו של אלגוריתם גנטי הוא חיקוי תהליך האבולוציה כפי שמתבצע בטבע - "החזק שורד". לשם הגדרת בעיית החיפוש שאנו מנסים לפתור באופן שניתן לפתרון ע"י אלגוריתם גנטי, תרגמנו את הבעיה באופן הבא:
  - <u>פרט באוכלוסייה</u> מוגדר ע"י רשימה של גנים, שבה כל גן מייצג מנה.
  - <u>גן</u> יהיה מספר שלם (int) שהוא האינדקס של השורה בה נמצאת המנה בדאטה פריים של המסעדה. מספר הגנים יהיה כמספר הסועדים, כלומר פרט באוכלוסייה הוא בעצם רשימת אינדקסים של מנות.
  - <u>מוטציה</u> תהיה החלפה של מנה אחת במנה אקראית אחרת מהמנות האפשריות של המסעדה, למוטציה יש הסתברות p מסוימת לקרות.
  - <u>זיווג</u> בין שתי פרטים באוכלוסייה יתבצע ע"י בחירה אקראית של מקור עבור כל אחד מהגנים מבין 2 ההורים.
  - <u>פונקציית ההתאמה (fitness)</u>: מקבלת פרט באוכלוסיה (פתרון אפשרי) ומחזירה ציון המייצג את מידת העמידה של הפתרון באילוצים. מוגדרת באופן כמעט זהה לפונקצית ה-gain למעט התאמות מינוריות הנדרשו לאופן פעולת האלגוריתם (למשל, פונקצית ה-gain מחזירה ערך 0 עבור אי עמידה באילוצים קשים, בעוד פונקצית ה-fitness מחזירה במקרה זה ערך חיובי קטן המשקלל את האילוצים הרכים).

כלומר, באלגוריתם זה אנו מייצרים אוכלוסייה של 3 פתרונות ובוחרים מתוכם את המתאימים ביותר לפתרון הבעיה (לפי פונקצית ה-fitness), ואז יוצרים באמצעות זיווג שלהם (עם הסתברות של 0.2 לקיום מוטציה) דור חדש של 5 פתרונות חדשים. נחזור על התהליך במשך 100 פעמים (דורות) ובסוף נגיע לפתרון הטוב ביותר. כדי להימנע מפתרונות שבהם בשלב הזיווג נוצרים צאצאים המשלבים מנות ממספר מסעדות שונות, החלטנו להריץ את האלגוריתם עבור כל מסעדה בנפרד (כלומר לבחור עבור כל מסעדה שילוב של 3 מנות מהתפריט שלה), ואז לבחור מבין הפתרונות שנמצאו את הפתרון (מסעדה ושלוש מנות) בעל ציון ה-fitness הגבוה ביותר.

מימוש האלגוריתמים

האלגוריתם הנאיבי מומש על ידינו. כל שאר אלגוריתמי החיפוש מלבד האלגוריתם הגנטי מומשו באמצעות חבילת simpleai, והאלגוריתם הנאיבי מומש באמצעות חבילת pygad.

פרק ג' – תוצאות ומסקנות

תיאור הבעיות שנבחנו

כדי לבחון את האלגוריתמים,בדקנו את הפתרונות שהם החזירו על מרחב החיפוש המלא (30,676,897 פתרונות) באמצעות ההעדפות של שלוש קבוצות סועדים שונות (המייצגות את העדפותיהם של שלושה צוותים עובדים אמיתיים – להרחבה ראו נספח ב'). להלן ההעדפות של שלוש קבוצות הסועדים שנבחנו:

ההעדפות של קבוצת סועדים מספר 1 (השמורות בקובץ input_constraints_1.txt):

	סועד 1	**סועד 2**	**סועד 3**
כשרות	לא משנה	לא משנה	לא משנה
צמחונות	צמחוני	לא משנה	לא משנה
נטול גלוטן	לא משנה	לא משנה	לא משנה
ללא אלכוהול	לא משנה	לא משנה	לא משנה
חריפות	לא חריף	חריף	חריף
מגבלת מחיר	75 ש"ח	80 ש"ח	75 ש"ח
מינימום דירוג	8	8	6
מידת רעב	רעב	לא רעב	לא רעב
סגנונות	נודלס, פסטה, דגים, קארי, אסייתי, אוכל ביתי, מזרח תיכוני, מקסיקני, הודי, יווני	פסטה, עוף	דגים, פוקי, סושי, אסייתי, המבורגר

ההעדפות של קבוצת סועדים מספר 2 (השמורות בקובץ input_constraints_2.txt):

	סועד 1	**סועד 2**	**סועד 3**
כשרות	כשר	כשר	לא משנה
צמחונות	לא משנה	צמחוני	לא משנה
נטול גלוטן	לא משנה	לא משנה	לא משנה
ללא אלכוהול	לא משנה	ללא אלכוהול	לא משנה
חריפות	לא משנה	לא משנה	חריף
מגבלת מחיר	90 ש"ח	80 ש"ח	100 ש"ח
מינימום דירוג	7	9	7
מידת רעב	לא רעב	רעב	רעב
סגנונות	בשר, המבורגר	בייגל, תאילנדי	סלט, תאילנדי

ההעדפות של קבוצת סועדים מספר 3 (השמורות בקובץ input_constraints_3.txt):

	סועד 1	סועד 2	סועד 3
כשרות	לא משנה	לא משנה	לא משנה
צמחונות	צמחוני	לא משנה	לא משנה
נטול גלוטן	לא משנה	ללא גלוטן	לא משנה
ללא אלכוהול	לא משנה	ללא אלכוהול	לא משנה
חריפות	לא חריף	חריף	חריף
מגבלת מחיר	75 ש״ח	100 ש״ח	65 ש״ח
מינימום דירוג	7	8	1
מידת רעב	רעב	רעב	לא רעב
סגנונות	טרי, מזרח תיכוני, בריא, סלט	אסייתי, יפני, תאילנדי	טורטיה, מקסיקני

הפתרונות שהופקו

עבור קבוצת סועדים מספר 1, הושגו הפתרונות הבאים:

	מסעדה	מנה לסועד 1	מנה לסועד 2	מנה לסועד 3
אלגוריתם נאיבי	Hakosem | Shlomo Hamelech	סלט ירקות	המרדומה של ״גילה״	המרדומה של ״גילה״
DFS	Poke Buddha Bowl	Tibet's Vegan הטבעוני מטיבט	Gyoza | Dim Sum- גיוזה | דים 5 Pcs סאם - 5 יח'	Rocher Ball (Vegen) כדור רושה (טבעוני)
UCS	Port 19 | Herzl	פורט טבעוני	נקניקייה חריפה אש	נקניקייה חריפה אש
A*	Zuk Farm Deli	מרק עגבניות (צמחוני)	כנפיים מהחווה - רוטב בפאלו חריף	כנפיים מהחווה - רוטב בפאלו חריף
hill climbing	Otello	Valencia Almonds	רושה שוויצרי	פקאן לבן
simulated annealing	Asia-T	פאד בונג טופו	גיוזה	1000 צ׳ילי
אלגוריתם גנטי	Chapo Burger	ארוחת צ׳יזבורגר טבעונית	ארוחת אל צ׳אפו	ארוחת אל צ׳אפו

עבור קבוצת סועדים מספר 2, הושגו הפתרונות הבאים:

	מסעדה	מנה לסועד 1	מנה לסועד 2	מנה לסועד 3
אלגוריתם נאיבי	Rak Marak	מרק עגבניות	מיקס מרק עדשים + כתומים	מיקס מרק עדשים + כתומים
DFS	Pizza Rondo | TLV	מלאווח פיצה	פיצה טבעונית זוגית	פיצה שום משפחתית
UCS	Metro Burger	מטרו חזה עוף	המבורגר טבעוני	ביג מטרו מרגז פיקנטי
A*	Metro Burger	ביג מטרו כבש	המבורגר טבעוני	ביג מטרו מרגז פיקנטי
hill climbing	Rak Marak	מיקס מרק כתומים + עגבניות	מיקס מרק כתומים + עגבניות	מיקס מרק כתומים + עגבניות
simulated annealing	Go Noodles | Ibn Gabirol	פאד בונג ביף	פאד בונג טופו	סאטה
אלגוריתם גנטי	Bechor and Shoshi | Tel Aviv	קציצות דגים - 5 יחידות	תבשיל ירקות מבושלים טבעוני	שקשוקה ברוטב עגבניות חריף

עבור קבוצת סועדים מספר 3, הושגו הפתרונות הבאים:

	מסעדה	מנה לסועד 1	מנה לסועד 2	מנה לסועד 3
אלגוריתם נאיבי	Hakosem | Shlomo Hamelech	סלט ירקות	המרדומה של "גילה"	המרדומה של "גילה"
DFS	Hakosem | Shlomo Hamelech	מחמצי הקוסם	אורז הקוסם	נתחי בשר לחומוס
UCS	Wok Republic | Allenby	ספייסי נודלס רידיפיין	מיקס רייס ללא גלוטן	בומביי צ'יקן
A*	Wok Republic | Ben Yehuda	קארי נודלס רידיפיין	ספייסי נודלס ללא גלוטן	מיקס רייס וואג'י
hill climbing	Rak Marak	מיקס מרק כתומים + עגבניות	מיקס מרק עדשים + כתומים	מרק כתומים אסיאתי
simulated annealing	Rak Marak	סלט סלקים ועלים	סלט כרובית	מרק עדשים
אלגוריתם גנטי	Wok Republic | Allenby	גרין נודלס רידיפיין	קונג פאו נודלס ללא גלוטן	ספייסי וואג'י נודלס

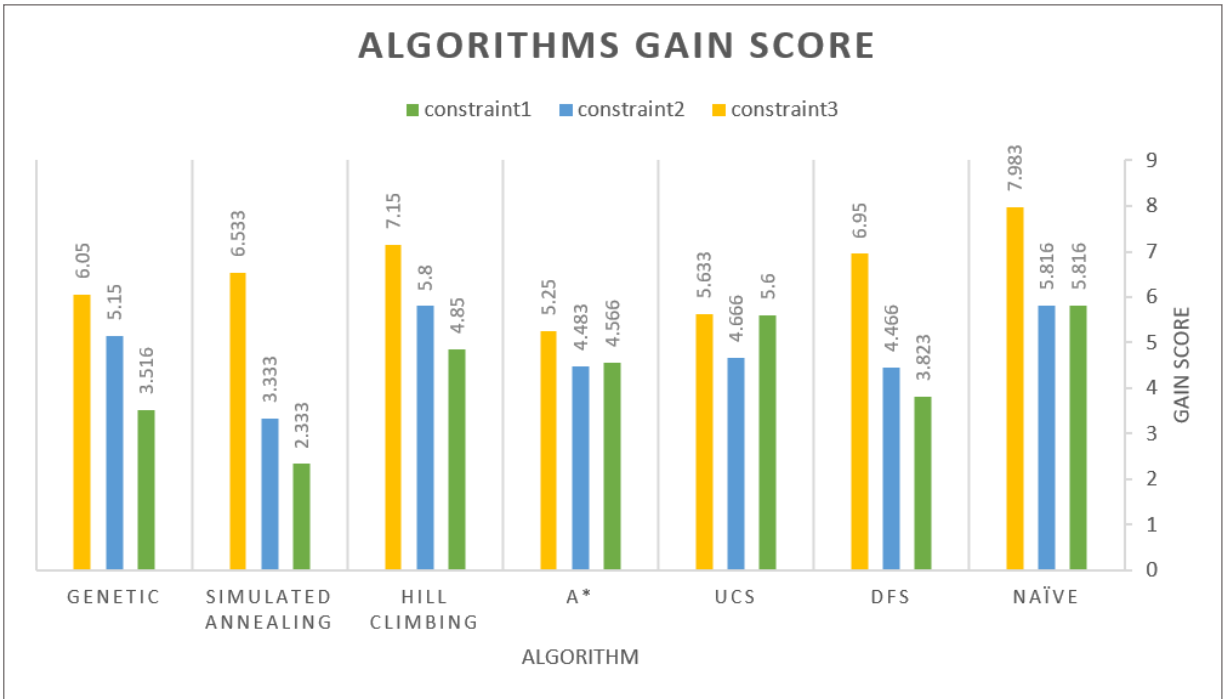
ניתוח התוצאות

ראשית, נזהה כי כיוון שכל אחד מהאלגוריתם פועל באופן שונה ומתבסס על פונקציה שונה (למשל פונקצית cost, פונקצית value, פונקצית יוריסטיקה, פונקצית fitness), כל אחד מהם החזיר פתרון שונה. למעט מקרים בודדים, בחינה אנושית של המנות שנבחרו על ידי האלגוריתמים השונים מעידה כי אכן מדובר במנות עיקריות אשר סביר כי הסועדים ישמחו לקבל לארוחת צהריים. למעשה, אחד הסועדים (סועד 2 מקבוצת סועדים 1) אף העיד כי המנה האופטימלית שנבחרה עבורו ע"י האלגוריתם הנאיבי היא אחת המנות האהובות עליו, שהוא מזמין לארוחת צהריים באופן תדיר.

בנוסף, עבור שלושת הקלטים והפתרונות הנ"ל, נציג השוואה בין תוצאות האלגוריתמים לפי שלושת הפרמטרים הבאים:

- ציון ה-Gain (גרף 1א וגרף 1ב)
- אחוז ההעדפות שסופק עבור כל אחד מקבוצות הסועדים (גרף 2א, גרף 2ב וגרף 2ג)
- זמן הריצה (גרף 3א וגרף 3ב)

<u>ציון ה-Gain</u>



גרף 1א – השוואת ציון ה-Gain של הפתרונות שהתקבלו ע"י האלגוריתמים השונים

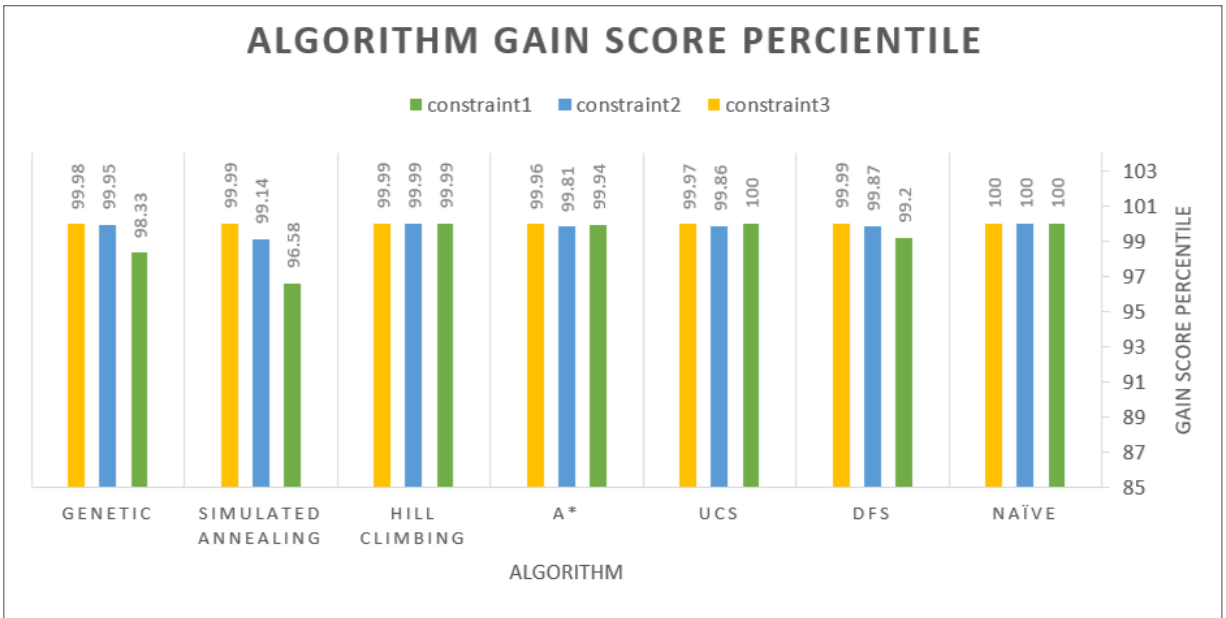
ראשית, נביט בגרף 1א, המציג השוואה בין ציון ה-Gain של הפתרונות שהתקבלו ע"י האלגוריתמים השונים, ונשתמש בו כדי להשוות בין איכות הביצועים של האלגוריתמים השונים. נתון זה משקף את מידת ההצלחה של כל אלגוריתם, שכן הצלחה הוגדרה ע"י השגת ציון Gain (המשקלל את העדפות הסועדים) מקסימלי. כיוון שהאלגוריתמים פועלים בדרכים שונות וכל אחד מהם מנסה למזער או למקסם פונקציה אחרת (אשר

דומה לפונקצית ה-Gain אך לא זהה לה), חישוב ציון ה-Gain על התוצאה הסופית של כל אלגוריתם יוצר מטריקה המאפשרת להשוות ביניהם.

ראשית, ניתן להבחין כי עבור כל אחד מהקלטים, האלגוריתם הנאיבי (המוצא את הפתרון האופטימלי בכל מרחב הפתרונות) משיג כצפוי את הפתרון בעל ציון ה-Gain הגבוה ביותר.

נשים לב כי בקווים כלליים, נראה שהאלגוריתמים מסכימים ביניהם על מידת הקושי במציאת פתרון עבור כל אחת מקבוצות ההעדפות שהוזנו להם כקלט –כל האלגוריתמים הצליחו למצוא עבור קבוצת האילוצים 3 ציון גבוה יחסית (3.357 בממוצע), וכן רובם (5 מתוך 7 אלגוריתמים) מצאו עבור קבוצת האילוצים השניה ציון זהה או גבוה יותר בהשוואה לקבוצת האילוצים הראשונה.

מבין האלגוריתמים שאינם האלגוריתם הנאיבי, האלגוריתם שהשיג את הציון המקסימלי עבור קבוצת ההעדפות 1 הוא אלגוריתם UCS, והאלגוריתם שהשיג את הציונים המקסימליים עבור קבוצות ההעדפות 2 ו-3 הוא אלגוריתם hill climbing.

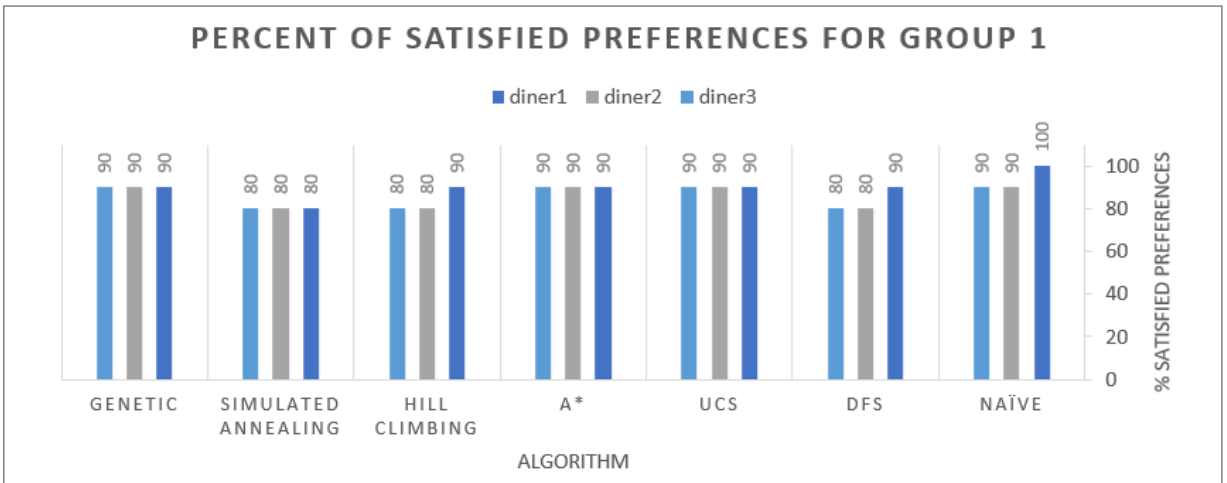


גרף 1ב– השוואת אחוזון ציון ה-Gain של הפתרונות שהתקבלו ע"י האלגוריתמים השונים

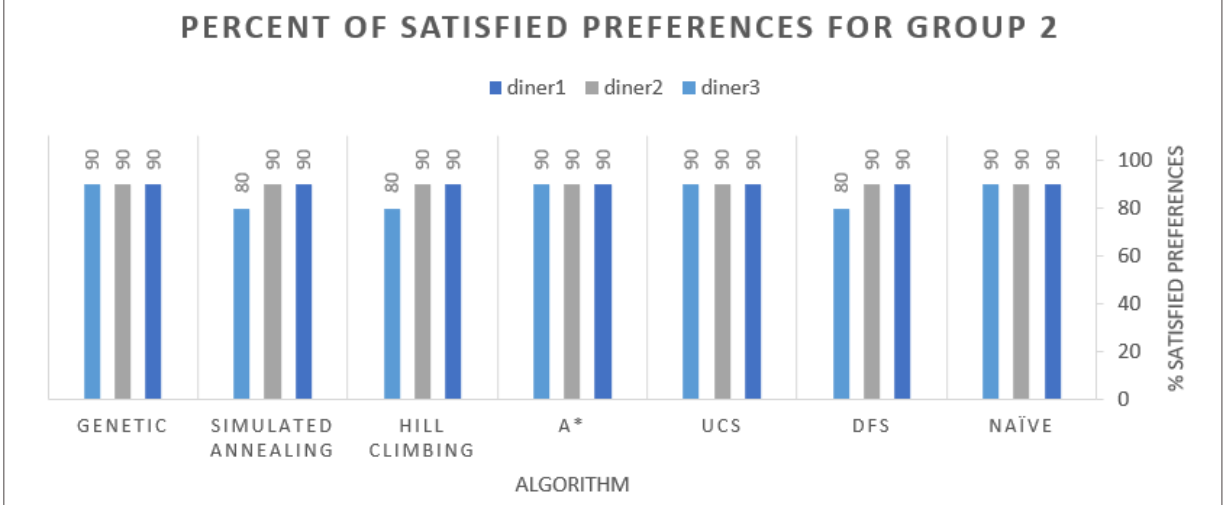
נביט בגרף 1ב, המשווה בין אחוזון ציון ה-Gain (מתוך מרחב החיפוש במלואו) של הפתרונות שהושגו ע"י האלגוריתמים, ונשתמש בנתון זה כדי לבחון את איכות הביצועים של כל אחד מהאלגוריתמים. כלומר, לכל אחת מקבוצות ההעדפות דירגנו את כל הפתרונות במרחב הפתרונות לפי ציון ה-Gain שהוענק לה ע"י האלגוריתם הנאיבי, ואז עבור כל אחד מהפתרונות שהחזירו האלגוריתמים השונים בחנו את אחוזון ציון ה-Gain אליו הוא משתייך.

ניתן לראות שלמעט שני מקרים, כל האלגוריתמים החזירו ציון הנמצא באחוזון העליון מתוך הציונים של הפתרונות במרחב החיפוש כולו (כלומר החזירו פתרון בעל ציון טוב יותר מ-99% מהפתרונות מבין מעל ל-30 מיליון פתרונות אפשריים). אלגוריתם UCS אף הצליח עבור קבוצת ההעדפות 1 להשיג פתרון בעל ציון הכמעט זהה לציון הפתרון האופטימלי, ודורג במקום ה-11 מבין ציוני כל הפתרונות האפשריים.

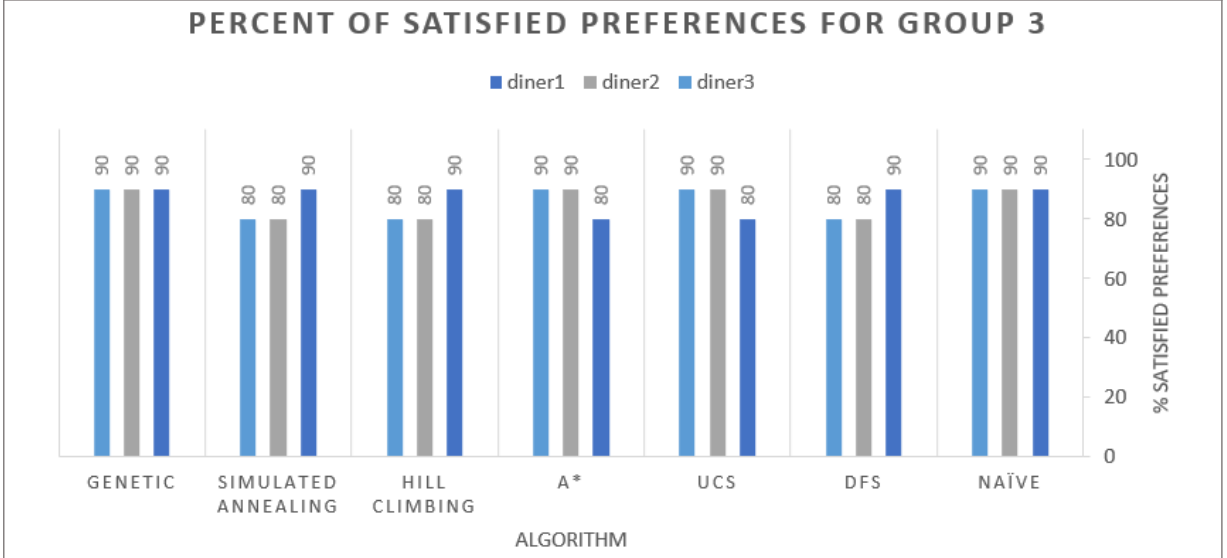
אחוז ההעדפות שסופק עבור כל אחד מקבוצות הסועדים



גרף 2א– השוואת ציון אחוז ההעדפות שסופקו ע"י הפתרונות שהתקבלו ע"י האלגוריתמים השונים עבור קבוצת הסועדים 1



גרף 2ב– השוואת ציון אחוז ההעדפות שסופקו ע"י הפתרונות שהתקבלו ע"י האלגוריתמים השונים עבור קבוצת הסועדים 2



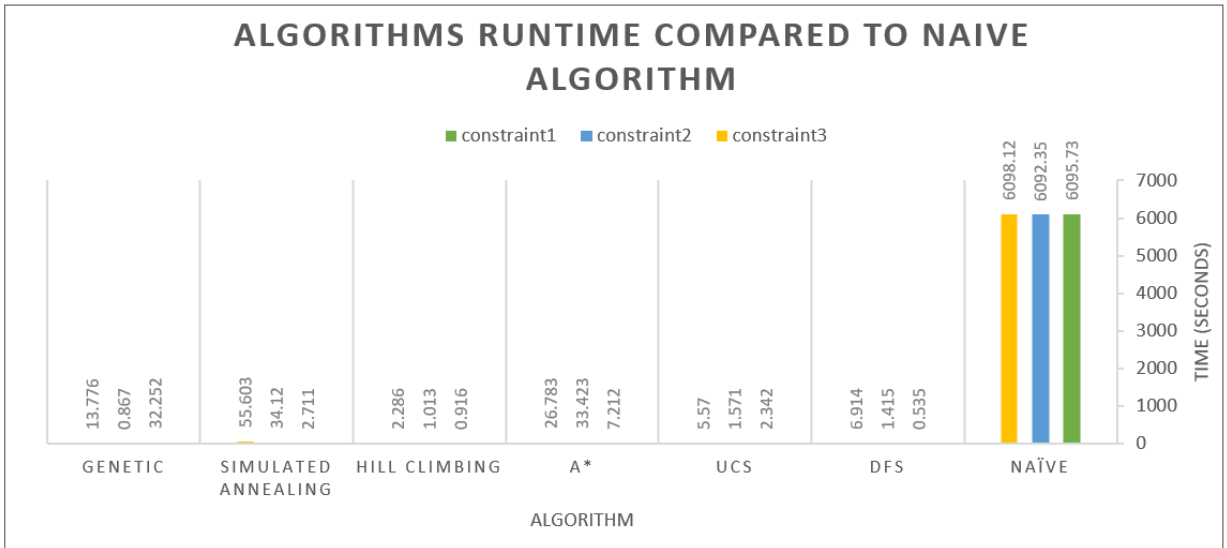
גרף 2ג– השוואת ציון אחוז ההעדפות שסופקו ע"י הפתרונות שהתקבלו ע"י האלגוריתמים השונים עבור קבוצת הסועדים 3

נביט בגרפים 2א, 2ב ו2ג, המציגים את אחוז ההעדפות שסופקו עבור כל אחד מהסועדים עבור קבוצת הסועדים הראשונה, השניה והשלישית בהתאמה. חשוב לבחון את הנתון הנ"ל כיוון שהוא מאפשר להעריך את איכותה של פונקצית ה-Gain שהגדרנו, כלומר לאשש כי מציאת פתרון עם ערך Gain מקסימלי אכן מביא לסיפוק האילוצים כנדרש.

ראשית, נשים לב שכל האלגוריתמים מצליחים לשמור על איזון בין צרכיהם של חברי הקבוצה השונים – עבור כל קבוצות הסועדים וכל האלגוריתמים שנבחנו, ההבדל הגדול ביותר חבר הקבוצה עם אחוז ההעדפות המסופקות הגבוה ביותר לבין חבר הקבוצה עם אחוז ההעדפות המסופקות הנמוך ביותר, הוא לכל היותר 10%. נציין כי כיוון שכל סועד מספק בדיוק עשר העדפות, הנ"ל שקול להבדל בסיפוק של העדפה אחת בדיוק. כלומר, לא נוצר מצב בו יש סועד "מקופח" שצרכיו נרמסים במטרה לענות על צרכיהם של חברי קבוצה אחרים.

כמו כן, ניתן לראות כי עבור אף אחת מקבוצות הסועדים, לא קיים במרחב החיפוש פתרון אשר מספק 100% מהעדפותיהם של כל הסועדים בקבוצה. מבט ברשימת האילוצים שלא התקיימו חושף כי האילוץ שהכי נפוץ שלא מתקיים הוא אילוץ סגנונות המזון. סיבה ראשונה העשויה להסביר את התופעה היא סתירה מובנית בין רשימות האילוצים שהוזנו כקלט. כיוון שכל אחד מהסועדים בצוות מציין את העדפותיו באופן בלתי תלוי בשאר חברי הצוות, יתכן מצב שבו לא קיים פתרון במרחב החיפוש אשר עונה על הצרכים של כל חברי הצוות במקביל. למשל, בשלוש הדוגמאות שנבחנו, אין שום חפיפה בין סגנונות המזון שמעדיפים שלושת חברי הצוות. מעבר לכך, בהרבה מהמקרים כל אחד מהסועדים בחר סגנונות שונים מאוד מחבריו לצוות, כך שלמרות שמסעדות רבות מציעות מגוון סגנונות מזון, לא קיימת ב-dataset אפילו מסעדה אחת שמציעה סגנונות מזון שיענו על צרכיהם של שלושת הסועדים.

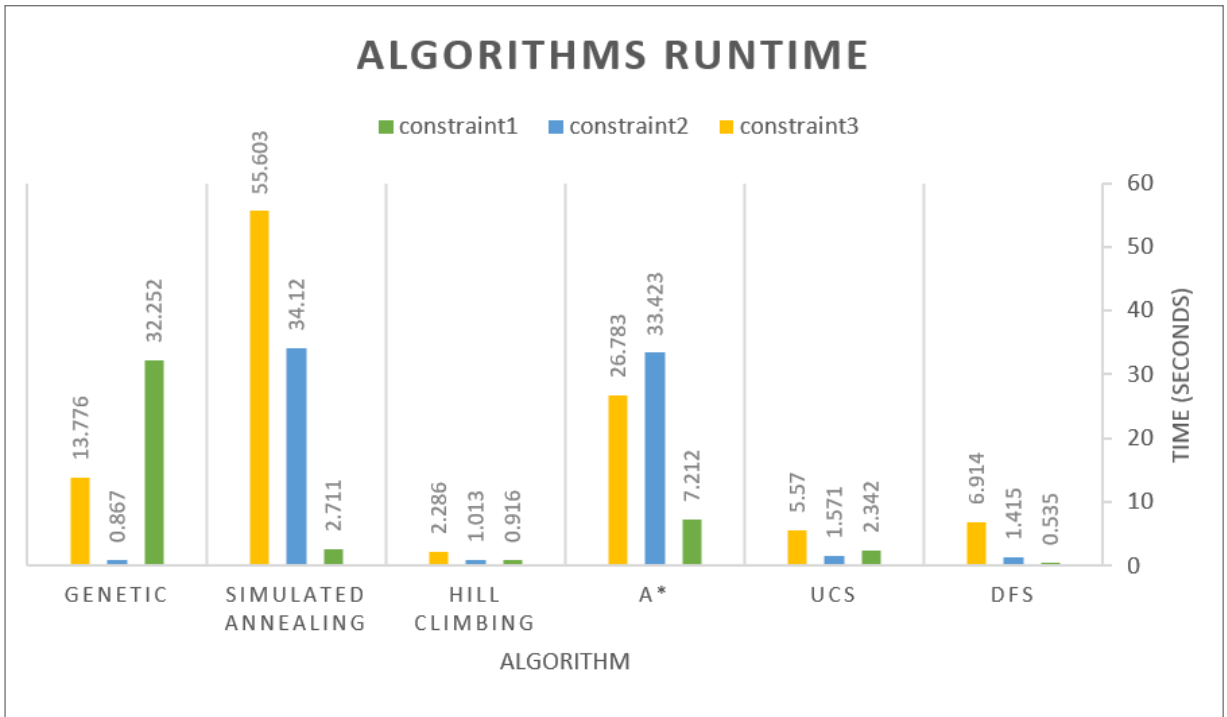
<u>זמן ריצה</u>



גרף 3א – השוואת זמן הריצה האלגוריתמים השונים לזמן הריצה של האלגוריתם הנאיבי

נביט בגרף 3א, המציג את ההבדל בזמני הריצה בין האלגוריתם הנאיבי לבין שאר האלגוריתמים שנבחנו. ניתן לראות כי כל האלגוריתמים הציגו שיפור עצום בזמן הריצה. למעשה, האלגוריתם בעל זמן הריצה האיטי ביותר (Simulated annealing על קבוצת ההעדפות השלישית) הצליח להחזיר פתרון בזמן קצר פי

110~ בהשוואה לאלגוריתם הנאיבי, והאלגוריתם בעל זמן הריצה המהיר ביותר (DFS על קבוצת ההעדפות הראשונה) החזיר פתרון בזמן קצר פי ~11,400 בהשוואה לאלגוריתם הנאיבי.



גרף 3ב – השוואת זמן הריצה האלגוריתמים השונים

כעת נביט בגרף 3ב, המציג השוואה בין זמני הריצה של האלגוריתמים השונים. ניתן לראות שהאלגוריתמים המהירים ביותר הם DFS, UCS ו-Hill climbing. שאר האלגוריתמים דורשים באופן עקבי יחסית זמנים ארוכים יותר לפתרון. חשוב לציין כי עבור אלגוריתמים בעלי אלמנט רנדומי (למשל DFS, Simulated Annealing והאלגוריתם הגנטי), זמני הריצה משתנים מאוד בין ריצות שונות של האלגוריתם.

פרק ד' – סיכום והצעות להמשך

לסיום, נדון במספר תובנות משמעותיות שעלו מתוצאות הפרויקט.

התבוננות בתוצאות הפרויקט מראה כי ששת האלגוריתמים שנבדקו הניבו תוצאות מרשימות הן במדד ציון ה-Gain (בו כל האלגוריתמים הצטיינו, והצליחו להחזיר כמעט תמיד פתרון באחוזון העליון מבין כלל מרחב החיפוש) והן במדד שביעות רצון הסועדים, כלומר אחוז ההעדפות המסופקות של כל אחד מהסועדים. בנוסף, כל ששת האלגוריתמים שנבחנו דרשו באופן עקבי זמני ריצה קצרים במספר סדרי גודל בהשוואה לזמן הריצה שדרש האלגוריתם הנאיבי שבדק את כל האפשרויות במרחב החיפוש. כיוון שהאלגוריתמים מחזירים בד"כ תוצאות שונות על אותה רשימת העדפות, בחרנו שלא לבחור ביניהם ולאפשר למשתמשים במערכת להריץ כל אחד מהם בנפרד כדי לקבל מגוון אפשרויות העשויות לקלוע לטעמם. עם זאת, בחרנו באלגוריתם Hill climbing בתור אלגוריתם ברירת המחדל, כיוון שהוא מציג איזון בין תוצאות עקביות (לפי ציון ה-Gain) לבין זמן ריצה קצר.

כמו כן, אמנם רוב המנות שהוחזרו ע"י האלגוריתמים אכן מסתמנות בבדיקה אנושית כמנות עיקריות, אך נראה שישנם מקרים בהם התוצאות כוללות מצרכי מזון או קינוחים. כפרויקט המשך ניתן לבחון את האפשרות לשימוש בכלי NLP על מנת לזהות בצורה מיטבית את המנות העיקריות מכל מסעדה.

הרחבה מתבקשת נוספת היא התאמת המודל כך שיוכל לפעול על כל מספר סועדים מבוקש. אמנם לצורך הפרויקט בחרנו לבחון את האלגוריתמים על שלושה סועדים בלבד (במטרה להיות מסוגלים להשוות את תוצאותיהם לתוצאות האלגוריתם הנאיבי), אך כעת לאחר שאיששנו כי פונקצית ה-Gain שהגדרנו אכן משיגה את מטרתה וכי האלגוריתמים אכן מחזירים תוצאות הולמות ומרשימות, ניתן לבצע את ההתאמות הנדרשות באלגוריתמים (שאינם האלגוריתם הנאיבי) כך שיוכלו לתמוך בקבוצות סועדים בגדלים שונים.

בנוסף, ניתן לבחון בהמשך להיות שימוש בדאטא המתקבל מה-API של Wolt בזמן אמת, והנגשת מודלים נבחרים לציבור הרחב, כך שכל צוות שמעוניין בכך יוכל להשתמש בהם כדי לתכנן את הזמנת ארוחת הצהריים שלו על בסיס דאטא עדכני שרלוונטי לזמן ההזמנה.

לסיכום, בפרויקט זה מידלנו את בעיית תכנון הזמנת מזון קבוצתית בתור בעיית חיפוש, ובחנו שישה אלגוריתמי חיפוש שונים אשר הצליחו למצוא במהירות מרשימה תוצאות הולמות עבור כל הסועדים בכל אחת משלוש קבוצות הסועדים שנבחנו.

נספח א' – הוראות להרצת התוכנית

יצירת קובץ קלט המייצג העדפות סועדים

כדי להגדיר את הפלט לאלגוריתם המייצג את העדפותיהם של שלושת הסועדים בצוות, יש לספק קובץ txt בפורמט הבא (נביט לדוגמא בקובץ input_constraints_3.txt, המכיל את אילוצי קבוצת הסועדים שהוצגה לאורך המסמך):

```
DINER 1 PREFERENCES
kosher - 0
vegetarian - 0
gluten_free - 0
alcohol_free - 0
spiciness - 0
max_price - 80
min_rating - 7
hungry - 0
cuisines - ['burger']
weekday - sunday


DINER 2 PREFERENCES
kosher - 0
vegetarian - 1
gluten_free - 0
alcohol_free - 0
spiciness - 0
max_price - 80
min_rating - 5
hungry - 0
cuisines - ['burger', 'hummus']
weekday - sunday


DINER 3 PREFERENCES
kosher - 0
vegetarian - 0
gluten_free - 0
alcohol_free - 0
spiciness - 1
max_price - 80
min_rating - 6
hungry - 0
cuisines - ['salad']
weekday - sunday
```

כלומר, יש למלא את ההעדפות של כל אחד משלושת הסועדים בצוות לפי המקרא הבא:

- kosher - יש למלא 1 עבור "חייב להיות כשר" או 0 עבור "לא משנה"
- vegetarian - יש למלא 1 עבור "חייב להיות צמחוני" או 0 עבור "לא משנה"
- gluten_free - יש למלא 1 עבור "חייב להיות נטול גלוטן" או 0 עבור "לא משנה"
- alcohol_free - יש למלא 1 עבור "חייב להיות נטול אלכוהול" או 0 עבור "לא משנה"
- spiciness - יש למלא 2 עבור "חריף", 1 עבור "לא חריף" או 0 עבור "לא משנה"
- max_price - יש למלא מספר שלם המייצג את מחיר המנה המקסימלי (כולל משלוח) בשקלים

- min_rating - יש למלא מספר שלם בין 1 ל-10 המייצג את הדירוג המינימלי של המסעדה שהסועד מוכן להזמין ממנה (עבור "לא משנה" יש למלא 1)
- hungry - יש למלא 1 עבור "רמת רעב גבוהה" (המשלוח צריך להגיע תוך 45 דקות) או 0 עבור "לא משנה"
- cuisines - יש למלא סוגריים מרובעים ובתוכם רשימת סגנונות מזון מוקפים במירכאות ומופרדים בפסיקים. רשימת הסגנונות האפשריים (מבחר וניסוחי הסגנונות השונים הנם כפי שהם מופיעים במקור ב-Wolt):

poke	meat	grill	coffe	american
salad	mediterranean	hamburger	cookies	Arabic
sandwich	mexican	healthy	curry	asian
schnitzel	middle eastern	home cooking	dessert	bagel
smoothie	nacho	hummus	dinner	bakery
soup	noodles	ice cream	ethnic	bowl
steak	pasta	indian	fish	bread
street food	pastries	italian	french	breakfast
sushi	pastry	japanese	fresh	burger
sweets	pita	kebab	georgian	café
thai	pizza	lunch	greek	chicken

- weekday - יש למלא את היום בשבוע בו מתבצעת ההזמנה, באנגלית ובאותיות קטנות.

מלבד החלפת הערכים המייצגים את האילוצים, אין לשנות את הקובץ (ובפרט, אין לשנות את הריווח בין השורות).

הרצת התכנית

לאחר יצירת קובץ העדפות הסועדים יש ליצור סביבה וירטואלית העומדת בדרישות התוכנתיות (כמפורט בקובץ requirements.txt) באופן הבא (הפקודות ותוכן חבילת הדרישות מותאמים למחשבי האקוריום):

```
virtualenv ./venv --python python3 --system-site-packages

source ./venv/bin/activate.csh

pip install -r ./requirements.txt
```

כעת ניתן להריץ את התכנית. לנוחיותכם מצורפים לפרויקט חמישה קבצי העדפות סועדים לדוגמא, אשר ניתן להריץ עליהם את התכנית.

על מנת להריץ את התכנית, יש לכתוב ב-command line את אחת משורות ההרצה הבאות:

```
python3 main.py <preference_file_path> <output_file_path>

python3 main.py <preference_file_path> <output_file_path> <algorithm>
```

כאשר preference_file_path הנו הניתוב לקובץ העדפות הסועדים הרצוי, output_file_path הוא הניתוב לקובץ בו ישמר הפתרון שהאלגוריתם יחזיר, ו-algorithm הוא האלגוריתם הרצוי מתוך הרשימה הבאה:

naive	dfs	ucs
hill_climbing	simulated_annealing	genetic

שימו לב שריצת האלגוריתם הגנטי לוקחת בממוצע יותר מ-10 שעות. במידה ותבחרו להריץ את התכנית מבלי לבחור אלגוריתם באופן מפורש, האלגוריתם שיורץ הוא אלגוריתם ברירת המחדל שהוא אלגוריתם hill climbing.

פלט התכנית

לאחר סיום ריצת התכנית (המלווה בהדפסות למסך על התקדמות ריצת האלגוריתם וסיומה) יודפס למסך וכן ישמר בקובץ הפלט שהגדרתם טקסט המתאר את הפתרון שהאלגוריתם החזיר. פלט לדוגמא (שנוצר כתוצאת מריצת האלגוריתם הגנטי על סט האילוצים השלישי):

```
genetic algorithm
----------------- CHOSEN SOLUTION -----------------
restaurant: Wok Republic | Allenby
    open: ✅
    kosher: ✅
    rating: ✅
    delivery matches hunger level: ✅
    cuisines: ❌

Meal for 1st diner: גרין נודלס ריידיפיין 🌿
    price: ✅, 13.0 ILS cheaper than max meal price
    vegetarian: ✅
    gluten: ✅
    alcohol: ✅
    spicy: ✅

Meal for 2nd diner: 🌶🌶 🌿 קונג פאו נודלס ללא גלוטן
    price: ✅, 43.0 ILS cheaper than max meal price
    vegetarian: ✅
    gluten: ✅
    alcohol: ✅
    spicy: ✅

Meal for 3rd diner: ספייסי וואג'י נודלס 🌶🌶
    price: ✅, 13.0 ILS cheaper than max meal price
    vegetarian: ✅
    gluten: ✅
    alcohol: ✅
    spicy: ✅


----------------- RESULTS -----------------
Gain score: 6.049999999999999
Total price: 171.0
Runtime: 13.77659010887146
```

נספח ב' – איסוף ועיבוד הדאטא

איסוף הדאטא

איסוף מידע על מסעדות ומנות

כדי לאסוף מידע על מסעדות והמנות אותן הן מציעות, השתמשנו ב-API של חברת Wolt. ה-API מספק גישה למידע בפורמט JSON העוסק במסעדות עצמן (ראו קובץ לדוגמא) או בתפריטים של מסעדות (ראו קובץ לדוגמא).

השתמשנו ב-API כדי לאסוף מידע על כל המסעדות הפעילות בת"א (313 מסעדות). בחרנו בעיר ת"א כיוון שהיא העיר בעלת מספר ומגוון המסעדות הזמינות ב-Wolt הגדול ביותר בישראל. כדי למנוע כפילויות במרחב החיפוש, במידה ומסעדה היא חלק מרשת של מסעדות המציעות את אותו התפריט, בחרנו באופן שרירותי את הראשונה מביניהן.

למרות שה-API מאפשר טכנית שליפה של המסעדות הפתוחות בכל רגע נתון, לטובת האחידות בתכנון ובהרצת האלגוריתמים השונים, בחרנו לאסוף באמצעות ה-API את הדאטא הרלוונטי, לשמור אותו offline כדי שנוכל להמשיך להשתמש באותו הדאטא באופן עקבי לאורך כל תהליך העבודה.

הדאטא שנאסף ועליו התבסס תהליך העבודה שמור בשני קבצי CSV (אחד המכיל מידע על מסעדות, ואחד המכיל מידע על מנות) בתיקיית data.

איסוף מידע על אילוצים של סועדים

כיוון שרצינו לפתור בעיות של אנשים אמיתיים, החלטנו לאסוף רשימות העדפות מצוותים אמיתיים אשר נתקלים באתגר הזמנת המזון המשותפת באופן יומיומי. לכן, פרסמנו בקבוצת "צרות בהייטק" בפייסבוק שאלון האוסף מצוותים אמיתיים את ההעדפות שלהם לפי הפורמט של רשימת הקלט שהגדרנו.

קיבלנו עשרות תשובות לשאלון, ומתוכן בחרנו שלושה צוותים שרשימות ההעדפות שלהם יצרו שילובים שתפסנו כמאתגרים לפתרון (למשל מנה נטולת גלוטן וגם חריפה, שילובי סגנונות מזון לא שגרתיים וכו'). רשימות ההעדפות של שלושת הצוותים הנ"ל שמורה בתיקיית example_preferences.

עיבוד הדאטא הגולמי ויצירת פרמטרים

כאמור בפרק 1, הקלט לבעיה בה אנו עוסקים מורכב מתיאור העדפות הסועדים (כאשר כל סועד מספק עשר העדפות). בהנתן ההעדפות של סועד מסוים ובהנתן חלק רלוונטי מפתרון (כלומר שם של מסעדה ושם של מנה מהתפריט שלה המיועדת לאותו סועד), ניתן לקבוע עד כמה הפתרון תואם להעדפותיו של הסועד. לכן, השתמשנו בהעדפות הסועדים ובדאטא שהתקבל מה-API אודות המסעדות והמנות כדי לייצר עבור כל פתרון אפשרי קבוצת פרמטרים (שחלקם בינאריים וחלקם מכילים ערך רציף) המייצגת את מידת העמידה של הפתרון בהעדפות הסועדים.

הגדרת מסעדה

לאחר בחינה של הדאטא המתקבל מה-API של Wolt, הבחנו כי הוא מכיל לא רק מסעדות אלא גם סופרמרקטים ומעדניות, אשר אינם מציעים מנות אלא מוצרים לרכישה. לכן, כדי לוודא כי מרחב החיפוש שלנו מכיל רק מסעדות, פסלנו כל עסק אשר לא צוין לגביו כי סוג העסק הוא מסעדה. בנוסף, כיוון שמטרת

האלגוריתמים היא להתאים לסועדים מנות אשר ניתן להזמין אותן במשלוח, פסלנו כל מסעדה או מנה אשר לא זמינות במשלוח.

הגדרת מנה עיקרית

הבעיה בה אנו עוסקים מתייחסת למציאת מנה עיקרית עבור כל אחד מהסועדים, לפיכך רצינו לצמצם את מרחב החיפוש שלנו כך שהוא יכיל רק מנות עיקריות. לצערנו, המידע המתקבל מה-API אינו מכיל סיווג אחיד של מנות ממנו אפשר להסיק האם מנה היא מנה עיקרית (כל מסעדה מחלקת את המנות שלה לקטגוריות לפי בחירתה החופשית, כלומר אין פרמטר אחיד המאפשר להסיק האם מנה היא מנה עיקרית). לכן, במטרה לפסול פריטים לא רלוונטיים מהתפריט (דוגמת שתיה, רטבים בתשלום, הערות להכנה אשר לעתים מופיעות כפריט בתפריט וכו') החלטנו להגדיר מנה עיקרית בתור מנה שמחירה הוא 30 ש"ח או יותר (מספר זה נבחר לאחר מספר נסיונות עם סכומים נמוכים יותר אשר בחינה ידנית של המנות שנותרו לאחר הסינון הראתה שהסינון לא היה מספק ונותר אחוז גדול של מנות שאינן עיקריות).

נציין שכיוון שהדרך היחידה המשמשת לזיהוי מנה עיקרית היא מחיר המנה, הזיהוי אינו הרמטי, ולכן מרחב הפתרונות מכיל גם אחוז מסוים של מנות ראשונות, תוספות וקינוחים.

נתאר כעת את הפרמטרים אשר יצרנו לכל פתרון אפשרי, ואת האופן בו השתמשנו בדאטא הגולמי ובהעדפות הסועדים כדי לייצר אותם (לפירוט טכני על המימוש ראו קובץ gainFunction.py).

פרמטרים העוסקים במסעדה

- **O (פתוח)** – פרמטר בינארי המקבל ערך 1 אם המסעדה פתוחה ביום ההזמנה, ו-0 אחרת. כיוון שרצינו לאפשר את קיום אילוץ זה (הזמנה ממסעדות פתוחות בלבד) למרות שאנחנו ניגשים לדאטא ב-offline, הסתמכנו על מידע המסופק ע"י ה-API בנוגע לכל מסעדה, המכיל רשימה של ימים בהם המסעדה פתוחה, וביקשנו מהסועדים לספק כחלק מהעדפותיהם את היום בו ההזמנה שלהם מתרחשת. כלומר, הפרמטר מקבל ערך 1 אם היום אותו ציינו הסועדים מופיע ברשימת הימים בהם המסעדה פתוחה.
- **M (מינימום הזמנה)** – פרמטר בינארי המקבל ערך 1 אם סכום שלושת המנות בפתרון גדול או שווה למינימום ההזמנה של המסעדה בפתרון, ו-0 אחרת.
- **K (כשר)** – פרמטר בינארי המקבל ערך 1 אם מידת הכשרות של המסעדה (כשרה / לא כשרה) תואמת את העדפות הסועדים, ו-0 אחרת. הסתמכנו על שיוך המסעדה לסוג המזון "כשר" או "כשר למהדרין" כדי לזהות מסעדות כשרות, וסימנו כל מסעדה שלא משויכת לאחת מהקטגוריות הנ"ל כלא כשרה. כעת, הפרמטר K יקבל ערך 0 במקרה בו לפחות אחד מהסועדים סימן שהוא מעוניין באוכל כשר אך המסעדה אינה כשרה, וערך 1 בכל מקרה אחר.
- **D (עמידה בהעדפות המשלוח)** – פרמטר בינארי המקבל ערך 1 אם זמן המשלוח הצפוי מהמסעדה עומד בהעדפות הסועדים, ו-0 אחרת. זמן המשלוח חושב ע"י סכימת זמן הכנת המנה הממוצע וזמן משלוח המנה הממוצע, כפי שסופקו ע"י ה-API. ההעדפה שמתקבלת מהסועדים בנוגע לזמן המשלוח מגולמת בציון רמת הרעב שלהם – אם הם ציינו שרמת הרעב שלהם גבוהה, הפרמטר D יקבל ערך 1 רק אם זמן המשלוח קטן או שווה ל-45 דקות.
- **DT (זמן המשלוח)** – פרמטר רציף המכיל את זמן המשלוח בדקות (מקבל ערכים בטווח בין 10 ל-120). זמן המשלוח חושב ע"י סכימת זמן הכנת המנה הממוצע וזמן משלוח המנה הממוצע, כפי שסופקו ע"י ה-API.

- **RD (הפרש הדירוג)** – פרמטר רציף המייצג את ההפרש בין דירוג המסעדה לממוצע הדירוגים של הסועדים (מקבל ערכים בטווח שבין 9- ל9). כלומר, ככל שהפרמטר מקבל ערך גבוה יותר, המשמעות היא שהמסעדה היא בעלת דירוג גבוה יותר בהשוואה לדירוג הממוצע של הסועדים. אם הפרמטר מקבל ערך שלילי, המשמעות היא שהמסעדה לא עוברת את דרישת הדירוג הממוצעת של הסועדים.
- **C (עמידה בהעדפות סגנונות המזון)** - פרמטר קטגורי המייצג את העמידה בהעדפות הסועדים בנוגע לסגנונות מזון (מקבל ערך מספרי שלם בין 0 ל-2 כולל). לצערנו, ה-API לא מספק מידע על הסגנון אליו משתייכת כל מנה, אלא רק רשימה של סגנונות מזון שהמסעדה מציעה ככלל. עבור כל סועד, יצרנו פרמטר בינארי זמני C_i המקבל ערך 1 אם לפחות אחד מהסגנונות ברשימת ההעדפות שלו מוצע ע"י המסעדה. כלומר, הפרמטר C מוגדר ע"י $C = C_1 + C_2 + C_3$, והוא מקבל ערך מספרי בין 0 ל-2 כולל המייצג את כמות הסועדים שהעדפת הסגנונות שלהם נענית ע"י המסעדה.

<u>פרמטרים העוסקים במנה</u>

כיוון שרצינו לשקף באמצעות הפרמטרים את ההתאמה של פתרון כלשהו (מסעדה ושלוש מנות) להעדפותיהם של כל שלושת הסועדים, עבור העדפות העוסקות במנה המיועדת לסועד ספציפי, יצרנו מכל העדפה שלושה פרמטרים – אחד עבור כל סועד. אלו הפרמטרים שייצרנו עבור כל $i \in [1,2,3]$:

- **V_i (צמחוני)** – פרמטר בינארי המקבל ערך 1 אם המנה עומדת בהעדפות הצמחונות של הסועד, ו-0 אחרת. כיוון שה-API לא מספק מידע מפורש על האם מנה היא צמחונית, הגדרנו מנה צמחונית בתור מנה המכילה את המילה "צמחוני" או "טבעוני" על הטיותיה השונות בעברית ובאנגלית, או מכילה את האימוג'י " 🌱 " אשר משמש מסעדות רבות לסימון מנות צמחוניות. חשוב לציין כי למרות ששיטה זו לסימון מנות צמחוניות מאפשרת ודאות גדולה יחסית במנות שהיא מזהה (מנה שתסומן כצמחונית היא כמעט בודאות צמחונית), ההשלכה לבחירה בשיטה זו היא שהיא מפספסת מנות רבות (כלומר יש מנות צמחוניות רבות כמו סלט או פסטה, שכל עוד המסעדה לא מציינת מפורשות שהן צמחוניות, יסומנו כלא צמחוניות בפרמטר זה). כלומר, הפרמטר מקבל ערך 0 אם הסועד צמחוני אך המנה אינה צמחונית, וערך 1 בכל מקרה אחר.
- **G_i (נטול גלוטן)** - פרמטר בינארי המקבל ערך 1 אם המנה עומדת בהעדפות הגלוטן של הסועד, ו-0 אחרת. כיוון שה-API לא מספק מידע מפורש על האם מנה היא נטולת גלוטן, הגדרנו מנה נטולת גלוטן בתור מנה המכילה את המילה "נטול גלוטן" על הטיותיה השונות בעברית ובאנגלית, או מכילה את האימוג'י " 🌾 " אשר משמש מסעדות רבות לסימון מנות נטולות גלוטן. באופן דומה למוסבר בנוגע למנות צמחוניות, גם במקרה זה יתכן כי נפספס מנות רבות שהנן נטולות גלוטן אם הדבר לא צוין מפורשות ע"י המסעדה. כלומר, הפרמטר מקבל ערך 0 אם הסועד נמנע מגלוטן אך המנה אינה נטולת גלוטן, וערך 1 בכל מקרה אחר.
- **A_i (נטול אלכוהול)** - פרמטר בינארי המקבל ערך 1 אם המנה עומדת בהעדפות האלכוהול של הסועד, ו-0 אחרת. המידע המתקבל מה-API מציין באופן מפורש עבור כל מנה מה אחוז האלכוהול שהיא מכילה, ולכן הגדרנו מנה כנטולת אלכוהול אם אחוז האלכוהול שהיא מכילה הוא 0. כלומר, הפרמטר מקבל ערך 0 אם הסועד נמנע מאלכוהול אך המנה מכילה אלכוהול, וערך 1 בכל מקרה אחר.

- S_i **(מידת חריפות)** - פרמטר בינארי המקבל ערך 1 אם המנה עומדת בהעדפות החריפות של הסועד, ו-0 אחרת. כיוון שה-API לא מספק מידע מפורש על האם מנה היא נטולת גלוטן, הגדרנו מנה חריפה בתור מנה המכילה את המילה "חריף" על הטיותיה השונות בעברית ובאנגלית, או מכילה את האימוג'י "🌶" אשר משמש מסעדות רבות לסימון מנות חריפות. כלומר, הפרמטר מקבל ערך 0 אם הסועד מעדיף מנה לא חריפה אך המנה חריפה או אם הסועד מעריף מנה חריפה והמנה לא חריפה, וערך 1 בכל מקרה אחר.
- PH_i **(עמידה בהגבלת מחיר)** - פרמטר בינארי המקבל ערך 1 אם המנה עומדת בהגבלת המחיר של הסועד, ו-0 אחרת. כלומר, הפרמטר מקבל ערך 1 אם מחיר המנה בשקלים קטן או שווה למחיר המקסימלי שהסועד ציין, וערך 0 בכל מקרה אחר.
- PS_i **(הפרש מחיר)** – פרמטר רציף המכיל ערך מספרי אי שלילי המייצג את ההפרש בשקלים בין המחיר המקסימלי שהסועד היה מוכן לשלם לבין מחיר המנה שהותאמה לו בפתרון. אם מחיר המנה בפתרון גבוה מהמחיר המקסימלי של הסועד, החלפנו את הערך המספרי השלילי שהתקבל בהפרש בערך 0.

נספח ג' – תיאור מרחב הפתרונות

מרחב הפתרונות

מרחב הפתרונות מוגדר ע"י הפרמטרים הבאים:

- k - מספר הסועדים בקלט הבעיה
- m - מספר המסעדות הזמינות ב-Wolt בת"א: 311
- לכל $i \in [0, m-1]$, n_i - מספר מספר המנות העיקריות (שמחירן יותר מ-30 ש"ח) בתפריט של המסעדה ה-i (נע בין 1 ל-130, כאשר הערך הממוצע הוא 25).

אנו מגבילים את מרחב החיפוש למנות רק מאותה מסעדה, הבעייה שלנו מתירה בחירה עם חשיבות לסדר, ולכן לכל $i \in [0, m-1]$, מספר הקומבינציות האפשריות למנות הוא n_i^k.

כלומר, מרחב הפתרונות מכיל בסה"כ $\sum_{i=0}^{m-1} n_i^k$ פתרונות.

עבור k=3 (בעיה עם שלושה סועדים), נקבל מניתוח כמות המנות העיקריות בכל אחת מהמסעדות בדאטא שנאסף כי מרחב הפתרונות מכיל 30,676,897 פתרונות אפשריים.

גודל מרחב הפתרונות כתלות בכמות הסועדים

כיוון שגודל מרחב הפתרונות מושפע באופן ישיר ומשמעותי מכמות הסועדים בקלט הבעיה, נתאר כעת את גודל מרחב הפתרונות שנוצר עבור בחירות שונות של כמות סועדים.

חיפוש נאיבי במרחב זה, כפי שתואר בספר הפרויקט, לוקח כ-14 שעות – נשתמש בנתון זה על מנת להעריך את הזמן שידרש ליצירת פתרון נאיבי עבור אפשרויות שונות של מספר סועדים:

- **עבור 4 סועדים** – נקבל מרחב חיפוש בגודל 31,137,832,773 (גדול פי ~100 בהשוואה לשלושה סועדים), כלומר חיפוש נאיבי בו יקח בקירוב חודשיים.
- **עבור 5 סועדים** נקבל מרחב חיפוש בגודל 366,888,674,329 (גדול פי ~12,000 בהשוואה לשלושה סועדים) כלומר חיפוש נאיבי בו יקח בקירוב 19~ שנים.
- **עבור 6 סועדים** - נקבל מרחב חיפוש בגודל 46,257,666,349,037 (גדול פי ~1,500,000 בהשוואה לשלושה סועדים) כלומר חיפוש בו יקח בקירוב 2400~ שנים.

כדי ליצור איזון ולבחור בעיה שתהיה מאתגרת מספיק אך תשאר אפשרית לפתרון בזמן הגיוני, בחרנו להגדיר את הבעיה שלנו עבור שלושה סועדים.