מבוא לבינה מלאכותית - 236501
תרגיל בית 1

מגישים:
יקיר חלץ 305028441
גל פלייסיג 302912985

פרק ראשון – משלוחי פיצה

חלק א׳ – מבוא והנחיות

1)

k value	Without fueling	With fueling (l = 5)
1	1	1
2	2	10
3	6	150
4	24	3000
5	120	75000
6	720	2250000
7	5040	78750000
8	40320	3150000000
9	362880	141750000000
10	3628800	7087500000000

חלק ב׳ – הגדרת מרחב החיפוש במפה

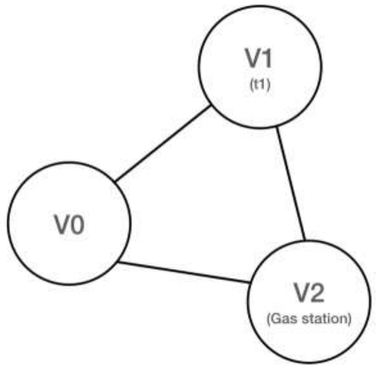
חלק ג׳ – הגדרת מרחב החיפוש של מסלולי נסיעת הטוסטוס

2)

a. הערך המקסימלי של מקדם הסיעוף במרחב החיפוש הוא במצב בו כלל הצמתים בגרף (צמתים המתאימים להזמנות וצמתי תחנות הדלק) מהווים קליקה. במצב כזה, כל צומת מחובר לכל הצמתים האחרים.מקדם הסיעוף של כל צומת שהוא צומת הזמנה או צומת תחנת דלק יהיה $k + l - 1$ אבל הצומת v0 מחובר לכל הצמתים ולכן מקדם הסיעוף שלו יהיה מקסימלי וגדול ב-1 משל שאר הצמתים, דהיינו $l + k$. נזכיר כי אין אופרטור מעבר מצומת שאינו v0 לצומת v0, לכן מקדם הסיעוף המקסימלי יהיה עבור v0.

b. הערך המינימלי של מקדם הסיעוף במרחב החיפוש הוא 1, בהנחה שהגרף המתאר את מרחב המצבים הינו קשיר. זאת מכיוון שיתכן צומת v (צומת המתאים להזמנה או צומת תחנת דלק) שניתן להגיע אליו מצומת אחד אחר בלבד. ואז מקדם הסיעוף של v הוא 1. בהנחה שהגרף לא בהכרח קשיר, מקדם הסיעוף יכול להיות גם 0 עבור צומת מבודד.

3) יתכנו מעגלים במרחב המצבים שלנו. לדוגמה עבור הגרף הבא כאשר v_1 הוא צומת המתאים להזמנה t_1, ו- v_2 הוא צומת תחנת דלק. נניח ש- $d_0 = 5$, ו- $d_{refuel} = 10$, ו- $dist(v_0, v_2) = 5$. אזי המצב ההתחלתי יהיה:



a. $S_0 = (v_0, 5, \{v_1\}, \varnothing)$
b. מכאן נעבור לתחנת הדלק ונגיע למצב $S_1 = (v_2, 10, \{v_1\}, \varnothing)$.
c. מכאן נחזור למצב ההתחלתי ונגיע למצב $S_2 = (v_0, 5, \{v_1\}, \varnothing)$.
d. קל לראות ש-S0=S2, כלומר ישנו מעגל במרחב המצבים.

4) נפריד לשני מקרים:

אם d שלם אזי:

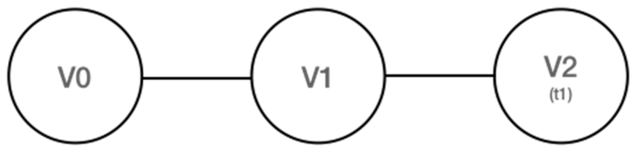
a. למיקום הנוכחי יש |V| אפשרויות (כאשר V כולל את כל צמתי ההזמנה, תחנות הדלק ואת v_0)
b. לכמות הדלק שנותרה יש d_{refuel}, אפשרויות (מספר שלם)
c. לקבוצת ההזמנות המחכות יש 2^k אפשרויות שכן כל הזמנה יכולה להיות במצב של נמסרה ובמצב של טרם נמסרה.
d. קבוצת ההזמנות הגמורות תלויה ב-T.

לסיכום, מספר המצבים במרחב זה הוא $|V| * d_{refuel} * 2^k$ מצבים.

עבור d שאינו שלם – ישנן אינסוף אפשרויות למצבים (מצפיפות המספרים האי-רציונליים), שכן בכל מצב אנו יכולים להיות עם d השונה מכל מצב אחר.

לא כל המצבים ישיגים. למשל, יתכן צומת v_1, שהוא צומת המתאר הזמנה, אשר ישיג רק דרך צומת v_2 שהינו צומת תחנת דלק, כאשר $dist(v_1, v_2) = d_{refuel} + 1$. אזי גם אם יעצור השליח לתדלק לפני המעבר למסור את ההזמנה ב- v_1 – הוא לעולם לא יוכל להגיע ל- v_1. ולכן כל מצב המכיל את v_1 כצומת נוכחי – אינו ישיג.

5) יתכנו בורות ישיגים מהמצב ההתחלתי שאינם מצבי מטרה. לדוגמה נתבונן בגרף הבא:



נניח ש- $d_0 = d_{refuel} = 5$. כמו-כן נניח ש- $dist(v_0, v_1) = dist(v_1, v_2) = d_{refuel} = 5$.
אזי המצב ההתחלתי הינו:
$S_0 = (v_0, d_{refuel} = 5, \{v_2\}, \varnothing)$.

כעת נעבור ל- v_1 ונקבל את המצב: $S_1 = (v_1, d_{refuel} = 5, \{v_2\}, \varnothing)$.

מצב זה הינו בור ישיג. נשים לב כי לא ניתן להמשיך ממנו לאף מצב אחר, ובנוסף זהו אינו מצב מטרה (v_1 אינו צומת המתאים להזמנה).

6) פונקצית העוקב מוגדרת כדלהלן.

$$Succ\left((v_1, d_1, T_1, F_1)\right)$$
$$= \left\{(v_2, d_{refuel}, T_2, F_2) \mid v_2 \in V, v_2 \in GasStations, (v_1, v_2) \in E, dist(v_1, v_2) \le d_1, T_2 = T_1, F_2 = F_1\right\}$$
$$\cup \left\{(v_2, d_2, T_2, F_2) \mid v_2 \in V, v_2 \in Ord, (v_1, v_2) \in E, dist(v_1, v_2) \le d_1, T_2 = T_1 \setminus \{v_2\}, F_2 = F_1 \cup \{v_2\}\right\}$$

הסבר: v_2 הוא צומת ב-V, יש דרך המקשרת בין הצמתים v_1, v_2, הדלק הנותר במיכל לפני תחילת הנסיעה מספיק כדי לנסוע מ- v_1 ל- v_2 (מבחינת מרחק – קילומטרים), ובנוסף אחת משתי אפשרויות לגבי T ו-F:
הצומת v_2 הוא תחנת דלק ואז $T_2 = T_1$ ו- $F_2 = F_1$, כלומר הם נותרים ללא שינוי, ובנוסף ק"מ הדלק שנשאר מתעדכן ל- d_{refuel} (הקבוצה הראשונה באיחוד) **או** שהצומת v_2 הוא צומת המתאים להזמנה, ואז F_2 הוא F_1 בתוספת הצומת v_2, ו- T_2 הוא T_1 ללא הצומת v_2 (כלומר ההזמנה v_2 הושלמה – הקבוצה השניה באיחוד).

7) בהנחה שאין שתי הזמנות במיקומים זהים, חסם תחתון לעומק המינימאלי של מצב מטרה כלשהו במרחב החיפוש הוא 2, וזאת מכיוון שיתכן מצב שבו מצומת ההתחלה יש דרך ישירה לצומת כלשהו (נניח תחנת דלק) ואז מתחנת דלק זו יש דרך ישירה לכל אחד מצמתי ההזמנות, ואז כדי להשלים את כל ההזמנות נצא מצומת תחנת הדלק ונגיע אל צומת ההזמנה הראשון בעומק 2, ואז נשוב לצומת תחנת הדלק ומשם נצא לצומת ההזמנה הבא בעומק 2 שוב, ונחזור לצומת תחנת הדלק, וכן הלאה לגבי כל צמתי ההזמנה. ואז נגיע למצב: $\left(v_t, d, \varnothing, Ord\right)$ כאשר v_t הוא צומת הזמנה כלשהו, d הוא מרחק כלשהו שניתן להשלים עם הדלק שנותר (נותרה כמות כלשהי של דלק), לא נותרו עוד הזמנות למסור ובהתאם כמובן כל ההזמנות נמסרו. והמצב שתיארנו שייך ל- G_d – כלומר הוא מצב מטרה – לפי ההגדרה. עומק 2 הוא העומק המינימלי מכיוון שלא ניתן לחזור לצומת v_0 ולכן יש לבצע לפחות שני צעדים מ v_0 (גם בהינתן שהצומת הראשון מ v_0 הוא צומת הזמנה).

<u>חלק ד׳ – מתחילים לתכנת</u>

8)

<u>להלן פלט הריצה לפני התיקון:</u>

Map(src: 54 dst: 549) UniformCost time: 0.01 #dev: 212 total_cost: 12.00000 |path|: 13 path: [54, 55, 56, 57, 58, 59, 60, 28893, 14580, 14590, 14591, 14592, 14593]

<u>להלן פלט הריצה המתוקנת:</u>

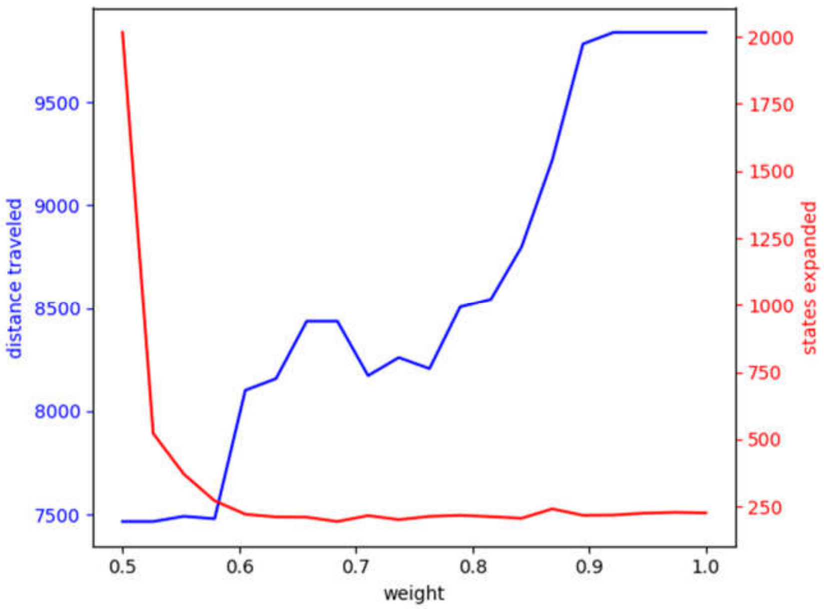
Map(src: 54 dst: 549) UniformCost time: 1.05 #dev: 17355 total_cost: 7465.52560 |path|: 137 path: [54, 55, 56, 57, 58, 59, 60, 28893, 14580, 14590, 14591, 14592, 14593, 81892, 25814, 81, 26236, 26234, 1188, 33068, 33069, 33070, 15474, 33071, 5020, 21699, 33072, 33073, 33074, 16203, 9847, 9848, 9849, 9850, 9851, 335, 9852, 82906, 82907, 82908, 82909, 95454, 96539, 72369, 94627, 38553, 72367, 29007, 94632, 96540, 9269, 82890, 29049, 29026, 82682, 71897, 83380, 96541, 82904, 96542, 96543, 96544, 96545, 96546, 96547, 82911, 82928, 24841, 24842, 24843, 5215, 24844, 9274, 24845, 24846, 24847, 24848, 24849, 24850, 24851, 24852, 24853, 24854, 24855, 24856, 24857, 24858, 24859, 24860, 24861, 24862, 24863, 24864, 24865, 24866, 82208, 82209, 82210, 21518, 21431, 21432, 21433, 21434, 21435, 21436, 21437, 21438, 21439, 21440, 21441, 21442, 21443, 21444, 21445, 21446, 21447, 21448, 21449, 21450, 21451, 621, 21452, 21453, 21454, 21495, 21496, 539, 540, 541, 542, 543, 544, 545, 546, 547, 548, 549]

חלק ה – אלגוריתם *A

9) מימוש בקוד.
10) מימוש בקוד.
11) מימוש בקוד. להלן פלט הריצה:

Map(src: 54 dst: 549) A* (h=AirDist, w=0.500) time: 0.14 #dev: 2016 total_cost: 7465.52560 |path|: 137 path: [54, 55, 56, 57, 58, 59, 60, 28893, 14580, 14590, 14591, 14592, 14593, 81892, 25814, 81, 26236, 26234, 1188, 33068, 33069, 33070, 15474, 33071, 5020, 21699, 33072, 33073, 33074, 16203, 9847, 9848, 9849, 9850, 9851, 335, 9852, 82906, 82907, 82908, 82909, 95454, 96539, 72369, 94627, 38553, 72367, 29007, 94632, 96540, 9269, 82890, 29049, 29026, 82682, 71897, 83380, 96541, 82904, 96542, 96543, 96544, 96545, 96546, 96547, 82911, 82928, 24841, 24842, 24843, 5215, 24844, 9274, 24845, 24846, 24847, 24848, 24849, 24850, 24851, 24852, 24853, 24854, 24855, 24856, 24857, 24858, 24859, 24860, 24861, 24862, 24863, 24864, 24865, 24866, 82208, 82209, 82210, 21518, 21431, 21432, 21433, 21434, 21435, 21436, 21437, 21438, 21439, 21440, 21441, 21442, 21443, 21444, 21445, 21446, 21447, 21448, 21449, 21450, 21451, 621, 21452, 21453, 21454, 21495, 21496, 539, 540, 541, 542, 543, 544, 545, 546, 547, 548, 549]

12) להלן הגרף שנוצר:



הסבר הגרף שהתקבל: עבור הגרף האדום שמתאר את מס׳ המצבים שפותחו כנגד המשקל (מהירות האלגוריתם), ניתן להבחין כי כאשר w = 0.5 מספר המצבים שפותחו הוא מקסימלי – והאלגוריתם הכי איטי, וככל ש-w גדל עד הגעה ל-w = 1, מספר המצבים שפותחו קטן, כאשר עבור w = 1 אנו מקבלים את האלגוריתם Greedy Best First אשר מסתמך בצורה בלעדית על היוריסטיקה ועקב כך מפתח הכי מעט צמתים.

לעומת זאת, עבור הגרף הכחול שמתאר את מסלול הפתרון שמצא האלגוריתם, ניתן להבחין כי כאשר w = 0.5 המרחק שעבר האלגוריתם הוא מינימלי – כלומר איכות הפתרון יותר טובה, וככל ש-w גדל עד הגעה ל-w = 1, מסלול הפתרון שמצא האלגוריתם גדל, כאשר עבור w = 1, שוב, אנו מקבלים את האלגוריתם Greedy Best First.

חלק ו – בעית המשלוחים המופשטת

13) מימוש בקוד.

14) נסביר את היוריסטיקה במילים: בהינתן מצב s, כל עוד קבוצת צמתי ההזמנות אינה ריקה (דהיינו ישנן עוד הזמנות למסור), אז מסתכלים על כל צמתי ההזמנות שנותרו למסירה ועל המרחק האווירי בין כל אחד מהם לצומת הנוכחי v, וערך היוריסטיקה הוא הערך המקסימלי מבין ערכים אלה. אם קבוצת צמתי ההזמנות ריקה, ערך היוריסטיקה הוא 0.

כפי שהוגדר בהרצאה, צריך להתקיים $0 <= h(s) <= h^*(s)$, כלומר ערך היוריסטיקה במצב כלשהו s קטן או שווה לערך הפתרון האופטימלי במצב זה, וגדול או שווה ל-0.

יוריסטיקה זו מקיימת את התנאי:

ראשית, עבור 0 <= h(s) זה תמיד מתקיים מכיוון ש-h(s) מודד מרחק אווירי בין צמתים, שהוא תמיד גדול או שווה ל-0, ובמקרה בו לא נותרו צמתי הזמנה לבקר בהם – ערך היוריסטיקה הוא 0.

כעת עבור h(s) <= h*(s):

נניח בשלילה שקיים s עבורו h(s) > h*(s), ונותרו עוד n צמתי הזמנה לחלק (n >= 1), אחרת אם לא נותרו צמתי הזמנה אזי h(s)=h*(s)=0). אזי הערך של h*(s) מתאר אורך מסלול שבו עוברים על כל n הצמתים שנותרו לחלוקה ועושים זאת בדרך קצרה ממש מהמרחק בין v לצומת ההזמנה המרוחק ממנו ביותר במרחק אווירי, שנסמנו u. מצב כזה יתכן אם ורק אם כל המסלול שעובר h*(s) נמצא על קו ישר בין v ל-u, שכן אחרת המסלול היה מ-v לצומת אחד לפחות שאינו על הקו הישר בין v ל-u, ומשם ל-u, ומסלול זה גדול ממש מהמרחק האווירי בין v ל-u בניגוד להנחה בשלילה (ניתן להסתכל על מסלול זה כעל משולש ואנו יודעים כי סכום אורכי כל זוג צלעות במשולש גדול ממש מאורך הצלע השלישית).

אם כן, המסלול שעובר h*(s) נמצא על קו ישר בין v ל-u. ואז המסלול האופטימלי של h*(s) הוא בדיוק על הקו הישר בין v ל-u, וערך מסלול זה שווה בדיוק למרחק האווירי בין v ל-u, שהוא h(s), כלומר מתקיים ש-h*(s) == h(s) בסתירה להנחה בשלילה.

בנוסף לאמור לעיל, יוריסטיקה קבילה צריכה לקיים בפרט שלכל מצב במצבי המטרה, ערך היוריסטיקה הוא 0, ואכן ניתן לראות כי כל מצב במצב המטרה מקיים ש- $s.T = \varnothing$ שכן חלוקת כלל ההזמנות נסתיימה, ולפי הגדרת היוריסטיקה בסעיף ערכה הוא 0.

לסיכום, היוריסטיקה MaxAirDist קבילה.

15) מימוש בקוד.

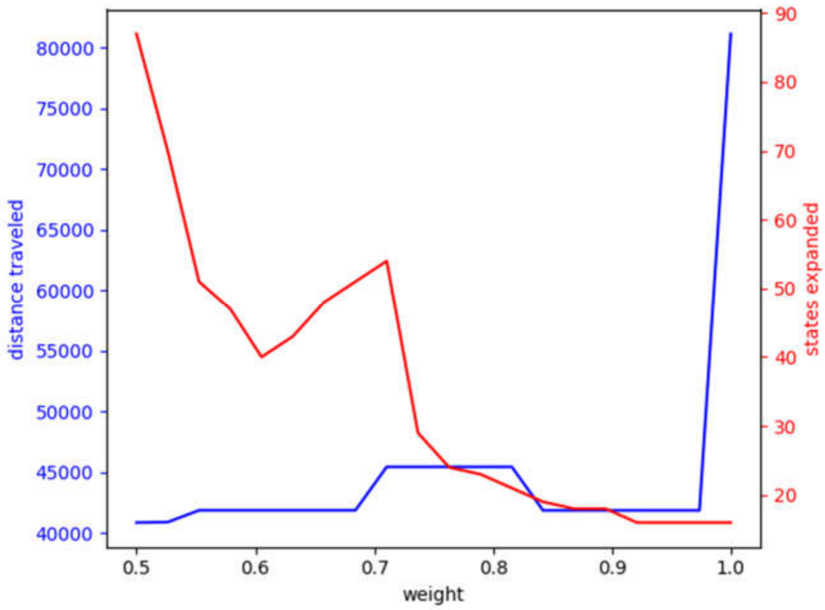
16) מימוש בקוד. להלן פלט הריצה:

Solve the relaxed deliveries problem.
RelaxedDeliveries(big_delivery)   A* (h=MaxAirDist, w=0.500)   time: 5.96   #dev: 3908   total_cost: 40844.21165 |path|: 11   path: [33919, 18409, 77726, 26690, 31221, 63050, 84034, 60664, 70557, 94941, 31008]   gas-stations: [31221, 70557]

17) מימוש בקוד. להלן פלט הריצה:

RelaxedDeliveries(big_delivery)   A* (h=MSTAirDist, w=0.500)   time: 2.53   #dev: 87   total_cost: 40844.21165 |path|: 11   path: [33919, 18409, 77726, 26690, 31221, 63050, 84034, 60664, 70557, 94941, 31008]   gas-stations: [31221, 70557]

18) מימוש בקוד. להלן הגרף שהתקבל:



חלק ז – אלגוריתם חיפוש חמדני-סטוכאסטי

19) שינוי הסקאלה לא משנה את ההתפלגות לוקטור x נתון. הוכחה:
מהגדרה עבור x,T נתונים, ההתפלגות מוגדרת באופן הבא:

$$\forall x_i \in x^t, \Pr(x_i) = \frac{\left(\frac{x_i}{\alpha}\right)^{-\frac{1}{T}}}{\sum_{pnt_h \in best\ N\ points} \left(\frac{x_h}{\alpha}\right)^{-\frac{1}{T}}}$$

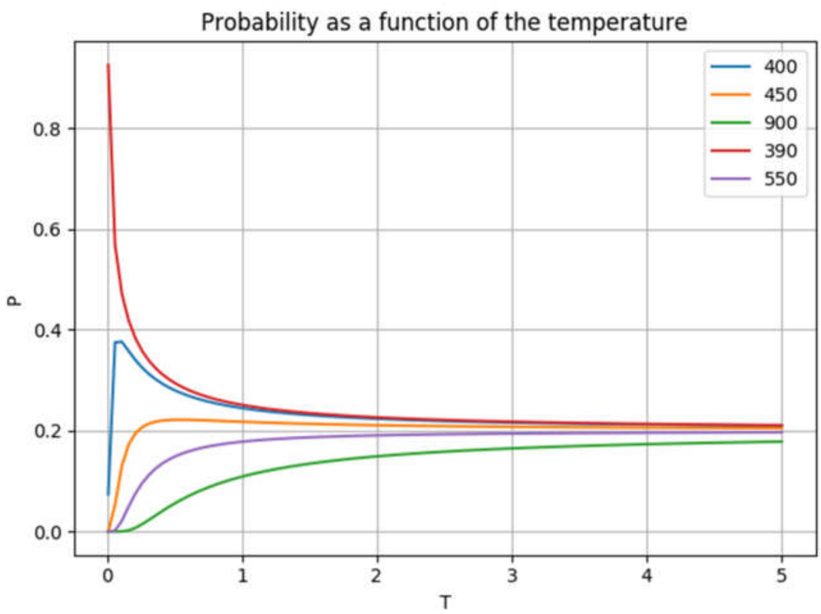
מכיוון שx נתון, אזי הוא קבוע ולכן מתקיים גם ש $\alpha = \min_j(x_j)$ קבוע. לכן מתקיים:

$$\forall x_i \in x^t, \Pr(x_i) = \frac{\left(\frac{x_i}{\alpha}\right)^{-\frac{1}{T}}}{\sum_{pnt_h \in best\ N\ points} \left(\frac{x_h}{\alpha}\right)^{-\frac{1}{T}}}$$

$$= \frac{(\alpha)^{-\frac{1}{T}} \cdot (x_i)^{-\frac{1}{T}}}{\sum_{pnt_h \in best\ N\ points} (\alpha)^{-\frac{1}{T}} \cdot (x_h)^{-\frac{1}{T}}} = \frac{(\alpha)^{-\frac{1}{T}} \cdot (x_i)^{-\frac{1}{T}}}{(\alpha)^{-\frac{1}{T}} \cdot \sum_{pnt_h \in best\ N\ points} (x_h)^{-\frac{1}{T}}} = \frac{(x_i)^{-\frac{1}{T}}}{\sum_{pnt_h \in best\ N\ points} (x_h)^{-\frac{1}{T}}} = \Pr(x_i)$$

כלומר, קיבלנו ששינוי הסקאלה משאיר את התפלגות ערכי הוקטור כפי שהיא.

20) מימוש בקוד. להלן הגרף שהתקבל:

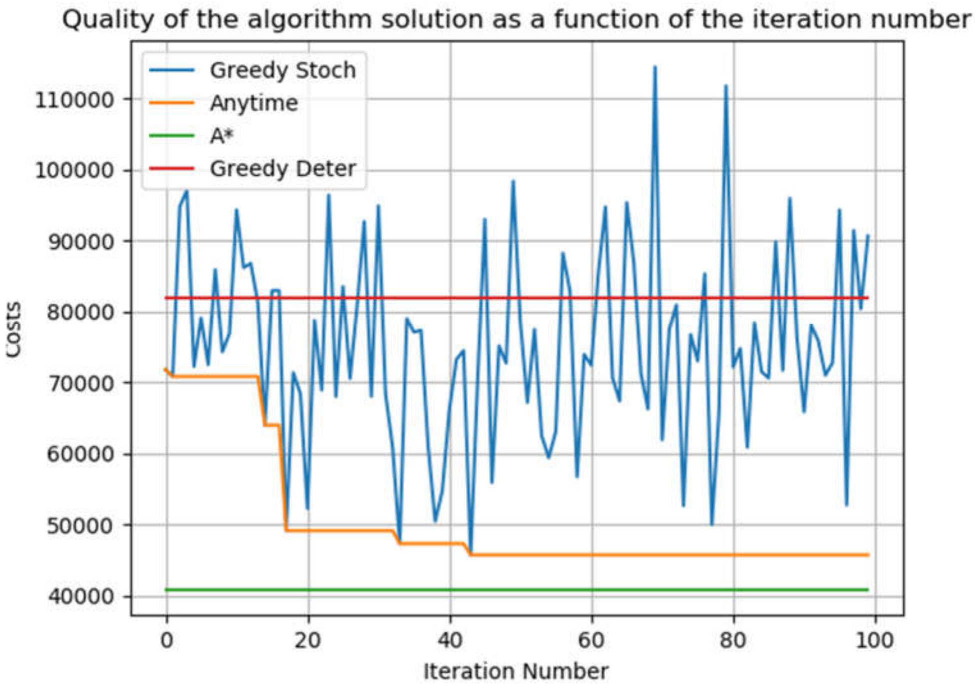


21) בגבול T -> 0 ניתן לראות כי ההסתברות של ערכי x שאינם המינימום שואפים ל-0, בעוד שההסתברות של ערך ה-x המינימלי שואף ל-1. כפי שהוסבר בסעיף על הטמפרטורה, ככל ש-T קטן (כלומר שואף ל-0), נעדיף בחירות "בטוחות" יותר, כלומר אלה עם ערך היוריסטיקה הנמוך ביותר (כלומר הקרובות ביותר למצב

המטרה), וכאן הבחירה ה"בטוחה" ביותר היא זו עם ערך ה-x המינימלי (390) – שמקבלת עדיפות בהסתברות.

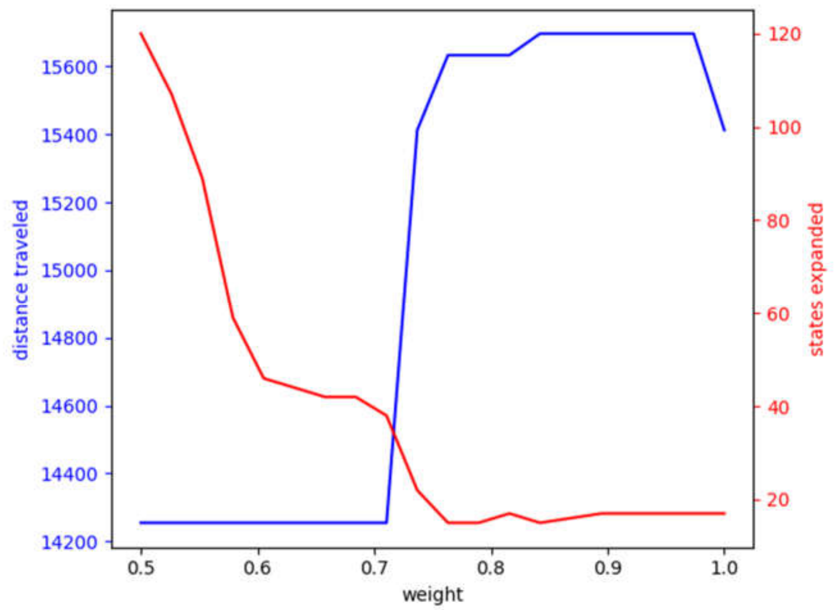
22) בגבול T -> infinity ניתן לראות כי ההסתברות של כלל ערכי x מתפלגת יוניפורמית. מכיוון שכאן יש 5 ערכים – ההסתברות של כל אחד מהם שואפת ל-0.2 = 1/5. כפי שהוסבר בסעיף על הטמפרטורה, ככל ש-T גדל (כלומר שואף ל-infinity), אנחנו מאפשרים בחירות "הרפתקניות" יותר, כלומר לא נעדיף בהכרח את הבחירה ה"בטוחה" ביותר וכל הבחירות יקבלו הסתברות שווה.

23) מימוש בקוד.
24) מימוש בקוד. להלן הגרף שהתקבל:



חלק ח – אלגוריתם מבוסס *A

25) מימוש בקוד.
26) מימוש בקוד. להלן פלט הריצה:



27) עבור בעית ה-StrictDeliveries נשתמש ביוריסטיקה שנקראת RelaxedDeliveriesHeuristic, שנסמנה h. **אופן** פעולת h הוא, בהינתן לה צומת נוכחי, תריץ את האלגוריתם *A עם היוריסטיקה MSTAirDistHeuristic. אם לא התקבל פתרון, היא תחזיר את הערך אינסוף, אחרת היא תחזיר את העלות של המסלול האופטימלי שמצאה *A בדרך אל התוצאה. יוריסטיקה זו קבילה, מכיוון שהיא מחזירה את התוצאה של אלג׳ *A עם היוריסטיקה MSTAirDistHeuristic, שהיא קבילה (מבוססת על מרחקים אויריים) ולכן *A קבילה. ולכן היוריסטיקה RelaxedDeliveriesHuristic שמכילה את שניהם קבילה.

28) מימוש בקוד. להלן פלט הריצה:

StrictDeliveries(small_delivery) A* (h=RelaxedProb, w=0.500) time: 8.56 #dev: 80 total_cost: 14254.81688 |path|: 8 path: [43516, 67260, 17719, 43454, 43217, 32863, 7873, 42607] gas-stations: [17719, 32863]

<u>הסבר:</u>
מספר המצבים שפותחו בהרצה בסעיף 28 הוא 80. לעומת זאת בריצה בסעיף 26 עם המשקלים המשתנים (בין משקל 0.5 למשקל 1) פותחו מספר מצבים בסדר גודל שנע בין כ-20 לכ-120. בפרט עבור משקל 0.5 בסעיף 26 פותחו כ-120 מצבים. עבור משקל של כ-0.565 לערך בסעיף 26 מספר הפיתוחים היה לראשונה נמוך יותר מאשר מספר הפיתוחים בפתרון מסעיף 28. העלות הכוללת עבור הפתרון עם משקל זה היא מחיר כולל של כ-14,254, ואילו עבור הפתרון מסעיף 28 המחיר הכולל הוא כ-14,254, כלומר ה-total_cost לא נפגע כלל (הם הגיעו לאותו מסלול בתוצאה). זאת מכיוון ששתי היוריסטיקות קבילות, ולכן יגיעו לאותו פתרון אופטימלי. זמן הריצה של סעיף 28, שהוא עם משקל דיפולטי w = 0.5, הוא 8.56 שניות, והוא נהיה **פחות טוב** ביחס לפתרון עבור משקל w = 0.565 בסעיף 26, שהוא כ-0.26 שניות, מכיוון שהיוריסטיקה בסעיף 28 כבדה יותר חישובית – אנחנו מריצים *A בתוך *A (בכל פעם שקוראים לפונ׳ היוריסטית) ולא רק *A פעם אחת כמו בסעיף 26, ובנוסף בפונ׳ estimate שמימשנו עבור RelaxedDeliveriesHeuristic אנחנו יוצרים מחדש את הקלט לבעיה Relaxed ובנוסף את הבעיה עצמה ואם זהו לא צומת מטרה גם פותרים אותה – כל אלה אלמנטים אשר מאריכים בצורה משמעותית את זמן הריצה כפי שאנו רואים.

<u>הערה</u>: להלן הפלט עבור סעיף 26 עם משקל w = 0.565:
StrictDeliveries(small_delivery) A* (h=MSTAirDist, w=0.565) time: 0.26 #dev: 77 total_cost: 14254.81688 |path|: 8 path: [43516, 67260, 17719, 43454, 43217, 32863, 7873, 42607] gas-stations: [17719, 32863]

פרק שני – שאלה תאורטית

א) אם h קבילה => לכל צומת s, $0 <= h(s) <= h^*(s)$. כעת נפריד את ההוכחה לשני מקרים:

a. אם הצומת s מוגדר ב-h, אזי $h0(h,s) = h(s)$ וגם $h0^*(h,s) = h^*(s)$ מכיוון שהן מתארות את אותה פונ׳ יוריסטית. בנוסף מתקיים כי $h0(h,s) >= 0$ לכל s שכן הפונ׳ מקבלת את הערך 0 (שעומד בתנאי) או h(s) שגם עומד בתנאי כי הוא תמיד אי-שלילי. ולכן מתקיים:
$0 <= h0(h,s) = h(s) <= h^*(s) = h0^*(h,s)$, כלומר בפרט:
$0 <= h0(h,s) <= h0^*(h,s)$, ולפי הגדרה היוריסטיקה קבילה.

b. אם הצומת s אינו מוגדר ב-h, אזי $h0(h,s) = 0$. כמו-כן, הערך המינימלי של h0 (h,s) שהוא $h0^*(h,s)$ הוא תמיד גדול או שווה ל-0, כלומר בפרט:
$0 <= h0(h,s) <= h0^*(h,s)$, ולפי הגדרה היוריסטיקה קבילה.

ב) נגדיר את היוריסטיקה להיות h1. אזי צריך להתקיים: לכל צומת n בגרף, $0 <= h1(n) <= h1^*(n)$. בנוסף, צריך להתקיים: לכל צומת n בגרף, $h1(n) >= h0(h,n)$.

<u>תיאור האלגוריתם:</u>

1. לכל צומת n בגרף, הרץ את Applicable_h(n). אם התקבל True – אזי $h1(n) = h(n)$. אחרת:
2. הרץ את IsGoal(n). אם התקבל True: $h1(n) = 0$. אחרת:
3. הרץ את Succ(n) וקבל את כל הצמתים שניתן להגיע אליהם מ-n. אם אין צמתים כנ"ל, החזר 0. אחרת, לכל צומת i כנ״ל, הרץ את Applicable_h(i) והחזר את הערך <u>המינימלי</u> מבין הערכים הנ״ל על כל הצמתים (אם כל הצמתים החזירו False: החזר 0).

<u>עסיבוכיות זמן:</u>
נפריד למקרים:
אם h מוגדרת על n – $h1(n) = h(n)$ וזה חישוב שמתבצע ב-O(1) כפי שנתון.
אם h אינה מוגדרת על n – אזי נעבור לשלב 2 של האלגוריתם. אם זהו צומת מטרה: $h1(n) = 0$ וזה חישוב שמתבצע ב-O(1).
אחרת, נגיע לשלב 3 של האלגוריתם, ונפעיל את הפונ׳ Succ(n) וזה גם כן חישוב שמתבצע ב-O(1). אולם, לכל אחד מהצמתים שניתן להגיע אליהם מ-n, שהם לכל היותר b צמתים, אנחנו מפעילים את הפונ׳ Applicable_h(i) והסיבוכיות שלה היא לכל היותר O(1), ולכן בסה״כ בשלב 3 של האלגוריתם החישוב יתבצע ב-O(b).

בסה״כ האלגוריתם רץ בסיבוכיות זמן O(b) לכל היותר.

<u>הוכחה ש-h1 קבילה:</u>
צ״ל: לכל צומת n בגרף, $0 <= h1(n) <= h1^*(n)$.

- הפונ׳ h1 מקבלת את הערך של הפונ׳ h (בשלב 1 של האלגוריתם) או שהיא מקבלת 0 (בשלב 2 של האלגוריתם. בשלב 3 של האלגוריתם יתכנו שני המקרים). ולכן אי השוויון הבא מתקיים: $0 <= h1(n)$ (נתון ש-h אי שלילית).
- כעת נותר להוכיח ש $h_1(n) \le h_1^*(n)$: אם h מוגדרת על n האלגוריתם יחזיר תשובה בשלב 1 והיא זהה לפונ׳ h, ואנו יודעים שעבור h מתקיים $h(n) <= h^*(n)$. אחרת, h לא מוגדרת על n. אזי: אם n הוא צומת מטרה, אזי האלגוריתם יחזיר תשובה בשלב 2 והתשובה היא 0, ומתקיים $h1(n) = 0 <= h1^*(n)$ תמיד. אחרת, אם הגענו לשלב 3 של האלגוריתם אזי:
  - אם כל השכנים של הצומת n אינם מוגדרים ב-h אזי $h1(n) = 0$ ומתקיים בהכרח $h1(n) <= h1^*(n)$. אחרת:
  - יש ל-n לפחות שכן אחד שכן מוגדר. אנחנו יודעים ש-n אינו צומת מטרה, אחרת האלגוריתם היה מפסיק בשלב 2. מכאן שבהכרח יש לו שכן אחד לפחות שהוא קרוב יותר למטרה ממנו (כלומר בעל ערך h קטן יותר), נסמן את השכן בעל ערך ה-h המינימלי בj. אזי נקבל $h1(n) = h1(j) = h(j)$ כי h מוגדר על j. כעת מקבילות h מתקיים שלכל צומת i, $h(i) <= h^*(i)$. ומכאן ש-$h1(n) <= h^*(j)$. כמו-כן $h1^*(j) = h^*(j)$ כי h מוגדרת על j. אבל בהכרח $h1^*(j) <= h1^*(n)$ שכן הצומת j קרוב יותר למטרה מהצומת n. ולכן $h1(n) <= h1^*(j) <= h1^*(n)$, כלומר $h1(n) <= h1^*(n)$.

ומכאן שהיוריסטיקה אכן קבילה.

הוכחה ש-h1 מיודעת יותר מאשר h0:

צ״ל: לכל צומת n בגרף, h1(n) >= h0(h,n).

- אם h מוגדרת על n אזי h1(n) = h(n) = h0(h, n) והתנאי מתקיים.
- אחרת, n אינו מוגדר, ואז h0(h,n) = 0 תמיד. ובנוסף h1(n) יקבל את הערך 0 אם הוא צומת מטרה או אם אף אחד מהשכנים שלו אינו מוגדר ב-h, או ערך גדול/שווה מכך אם קיים לו לפחות שכן אחד שמוגדר ב-h. בכל מקרה, $h1(n) \geq 0 = h0(h,n)$.

ג) כעת, לא נתון דבר על טופולוגיית מרחב המצבים. הפתרון שכתבנו בסעיף ב' מתאים באופן כללי לגרף ללא מגבלות (שאינו בהכרח עץ), וכאשר לא נתון דבר על טופולוגיית מרחב המצבים, ניתן לתארה כגרף כזה. מכאן שנוכל להשתמש באותה יוריסטיקה מסעיף ב', והוכחנו שהיא אכן קבילה ומיודעת יותר מאשר h_0.

ד) <u>קיים</u> אלגוריתם קביל המשתמש ביוריסטיקה h0(h’,s) וזמן ריצתו חסום מלעיל על-ידי A* המשתמש באותה יוריסטיקה h0(h’,s). אלגוריתם זה הוא <u>IDA*</u>.

- <u>האלגוריתם קביל:</u>
  - האלגוריתם קביל אם פונ׳ המחיר חסומה מלמטה וחיובית, והיוריסטיקה שבה הוא משתמש היא קבילה:
  - פונ׳ המחיר חסומה מלמטה וחיובית – מהנתון.
  - היוריסטיקה h0 שבה הוא משתמש היא קבילה:
    - אם s מוגדר ב-h’ אזי h0(h’,s) = h’(s), ונתון כי h’ מוגדרת רק על צומת ההתחלה, שעבורו מתקיים h’(s) = h’*(s). בנוסף h’*(s) <= h0*(h’,s) שכן הערך h’*(s) מעצם הגדרתו הוא מינימלי עבור הצומת s, והיוריסטיקה h0 משתמשת ב-h’ לצורך פעולתה. מכאן ש- 0 <= h0(h’,s) <= h0*(h’,s).
    - אם s אינו מוגדר ב-h’ אזי h0(h’,s) = 0 לפי הגדרת h0, ובפרט 0 <= h0*(h’,s), כלומר מתקיים h0*(h’,s) => 0. ובפרט 0 = h0(h’,s) <= h0*(h’,s) <= h0(h’,s) => 0.

- <u>זמן הריצה חסום:</u>

נבחן את A* עם h0 לעומת IDA* עם h0. האלגוריתם A* עם h0, עבור כל צומת שאינו צומת ההתחלה, מקבל את הערך 0. אזי h0 = 0 כלומר A* יסתמך על ערך g בלבד, דהיינו הוא יבצע Uniform Cost סטנדרטי כפי שנלמד.
לעומתו, ב-IDA* אמנם גם מתקיים h0 = 0 לכל צומת שאינו צומת ההתחלה, אך אלגוריתם זה מבצע חיפוש לעומק עם הגבלת העומק ולא Uniform Cost. הגבלת העומק מסתמכת על ערך היוריסטיקה של צומת ההתחלה, שכפי הנתון היא יוריסטיקה מושלמת, כלומר IDA* לא יצטרך במקרה זה לעדכן את עומק ההעמקה שלו.

לפי ההנחה שבסעיף, עבור כל מצב, סדר היורשים שלו מוגדר היטב ושני האלגוריתמים עוברים על היורשים שלו באותו הסדר. כלומר יתכן כי A* ו-IDA* יגיעו לפתרון כלשהו באותו הזמן (מפתחים את הצמתים היורשים של צומת כלשהו באותו הסדר), אולם כאשר A* הגיע לפתרון, הוא עשוי להמשיך ולחפש פתרון טוב יותר עבור הצמתים שב-open, דבר ש-IDA* לא יעשה.
מכיוון שסדר היורשים של כל מצב מוגדר היטב, לא יתכן ש-A* יגיע לפתרון לפני IDA* ויסיים את ריצתו לפני IDA* אלא במקרה הגרוע (עבור IDA*) הם יגיעו לפתרון באותו הזמן, ובמקרה הטוב (עבור IDA*) הם יגיעו לפתרון כלשהו אך בעוד ש-IDA* יסיים את ריצתו, A* ימשיך לחפש פתרונות טובים יותר וירוץ למשך יותר זמן.