ליאור רמתי 316490291 אסף אנטר 301019733

בינה מלאכותית – גיליון 1

פרק 1

שאלה 1

K	1	2	3	4	5
ללא אילוץ	1	2	6	24	120
עם אילוץ(5)	1	10	150	3000	75,000

K	6	7	8	9	10
ללא אילוץ	720	5040	40,320	362,880	3,628,800
עם אילוץ(5)	2,250,000	78,750,000	3,150,000,000	1.4175e11	7.0875e12

שאלה 2

ערכי הקיצון הם $b_{\min} = 0, b_{\max} = |\mathrm{Ord}| + |\mathrm{GasStations}|$.

תרחיש עבור ערך מינימום: אין הזמנות ואין תחנות דלק.

תרחיש עבור ערך מקסימום: מצב התחלתי שכל ההזמנות פתוחות ויש מספיק דלק להגיע לכל הזמנה או לכל תחנת דלק.

שאלה 3

ייתכנו מעגלים, יהיו 2 תחנות דלק f_i, f_j , ניתן להפעיל אופרטור תדלוק מתחנת דלק i $O_{f_i}(f_j, d_{full} T, F)$ ואז להפעיל אופרטור תדלוק מתחנת דלק j $O_{f_j}(f_i, d_{full} T, F)$ וכך הלאה.

שאלה 4

עבור מצב $S = \{v, d, T, F\}$ ננתח את מספר המצבים באופן הבא:

נסמן $k=|Ord|$ <- מספר ההזמנות וכמו כן $l=|GasStations|$ <- מספר תחנות הדלק.

עבור v ברור כי יש $k+l$ אפשרויות.

עבור d יש $d_{full}+1$ אפשרויות.

עבור T יש $2^k=\sum_{m=0}^{k}\binom{k}{m}=\binom{k}{0}+\binom{k}{1}+..\binom{k}{k}$ (הבינום של ניוטון).

ולכן סך הכול ישנם: $(k+l)(d_{full}+1)2^k$ מצבים.

<u>שאלה 5</u>

ייתכנו בורות, במידה והגענו ליעד שממנו אין מספיק דלק להגיע לאף יעד אחר.

<u>שאלה 6</u>

היעד יכול להיות הוא יעד של הזמנה או תחנת דלק.

כמו הדלק יכולה להיות קטנה או שווה למיכל מלא וגדולה או שווה ל – 0.

קבוצת ההזמנות שנותרו מוכלת או שווה לקבוצה הקודמת לה.

קבוצת ההזמנות שנסגרו מכילה או שווה לקבוצה הקודמת לה.

לכן הפונקציה היא:

$$succ(v_1,d_1,T_1,F_1)=\{(v_2,d_2,T_2,F_2)\,|\,v_2\in(O_d\cup G_d),0\le d_2\le d_{full},T_2\subseteq T_1,F_2\supseteq F_1\}$$

<u>שאלה 7</u>

העומק המינימאלי הוא $k=|Ord|$ אם למשל יש לנו מספיק דלק לעבור בכל התחנות ללא צורך במילוי דלק, אז נעבור בכל התחנות ולכן לפחות k מצבים.

<u>שאלה 8</u>

הפלט שהתקבל ל MapProblem:

```
Solve the map problem.
Map(src: 54 dst: 549)               UniformCost               time:   0.63   #dev: 17355   total_cost:  7465.52960   |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]
```

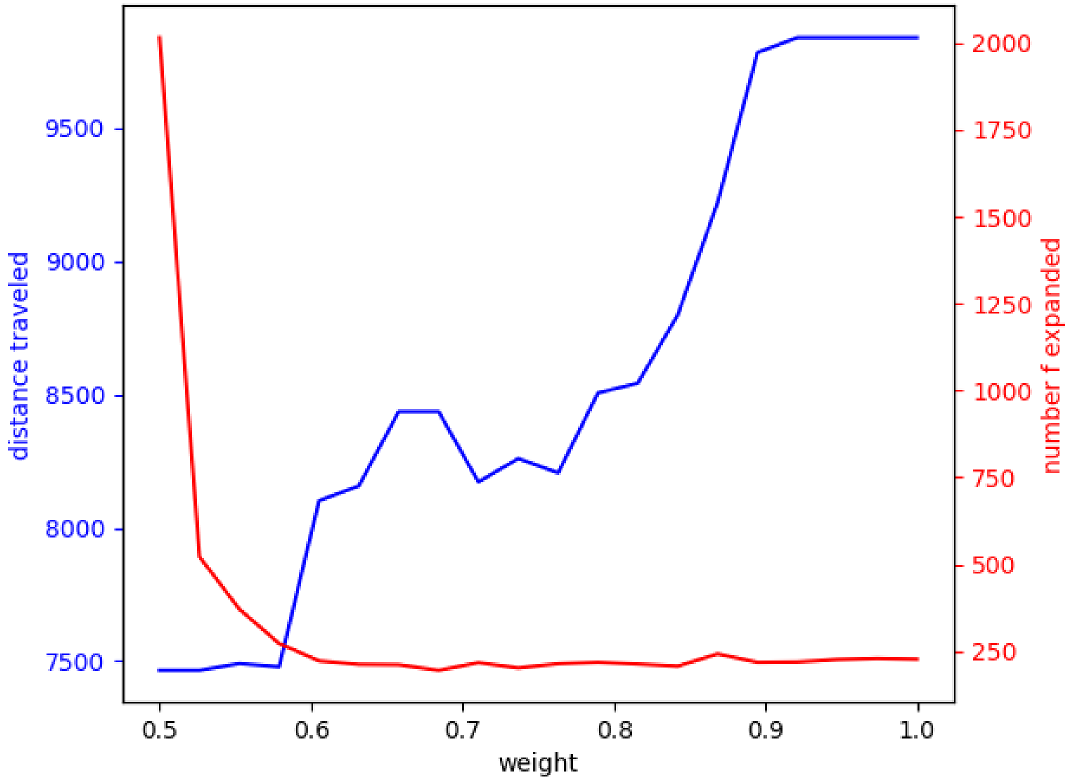
<u>שאלה 11:</u>

הפלט שהתקבל ל MapProblem:

```
Solve the map problem.
Map(src: 54 dst: 549)           UniformCost             time:   0.63   #dev: 17355   total_cost:  7465.52960   |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]
Map(src: 54 dst: 549)           A* (h=0, w=0.500)       time:   0.62   #dev: 17355   total_cost:  7465.52960   |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]
Map(src: 54 dst: 549)           A* (h=AirDist, w=0.500)   time:   0.10   #dev: 2016   total_cost:  7465.52960   |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]
```

<u>שאלה 12:</u>

מספר הפיתוחים והמרחק כפונקציה של המשקל:



ניתן להסיק מהגרף שככל שהמשקל גדל כמות הפיתוחים יורדת ואילו איכות הפתרון, כלומר המרחק, גדל.

<u>שאלה 14</u>

ידוע כי ישר בין שתי נקודות הוא המרחק הקצר ביותר בניהם שזהו למעשה מרחק אווירי ולכן היוריסטיקה קבילה ומתקיים: $0 \le h(s) \le h^*(s)$.

שאלה 16

פלט הריצה:

```
Solve the relaxed deliveries problem.
RelaxedDeliveries(big_delivery)     A* (h=MaxAirDist, w=0.500)    time:   4.93   #dev: 3908    total_cost: 40844.21165   |path|: 11    path: [33919, 18409, 77726, 26690,
 31221, 63050, 84034, 60664, 70557, 94941, 31008]   gas-stations: [31221, 70557]
```

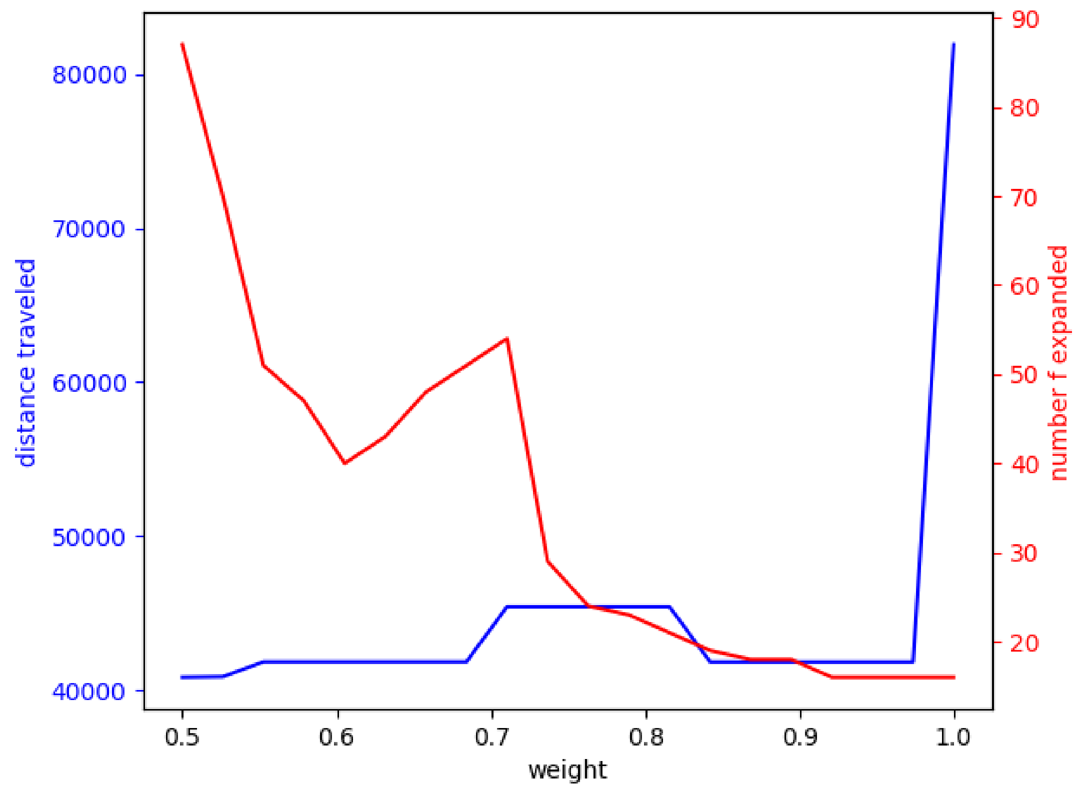
שאלה 17

פלט הריצה:

```
RelaxedDeliveries(big_delivery)     A* (h=MSTAirDist, w=0.500)    time:   1.45   #dev: 87     total_cost: 40844.21165   |path|: 11    path: [33919, 18409, 77726, 26690,
 31221, 63050, 84034, 60664, 70557, 94941, 31008]   gas-stations: [31221, 70557]
```

שאלה 18

שימוש ב A* עם משקלים שונים עם יוריסטיקה MSTAirDistHeuristic הניבה את הגרף:



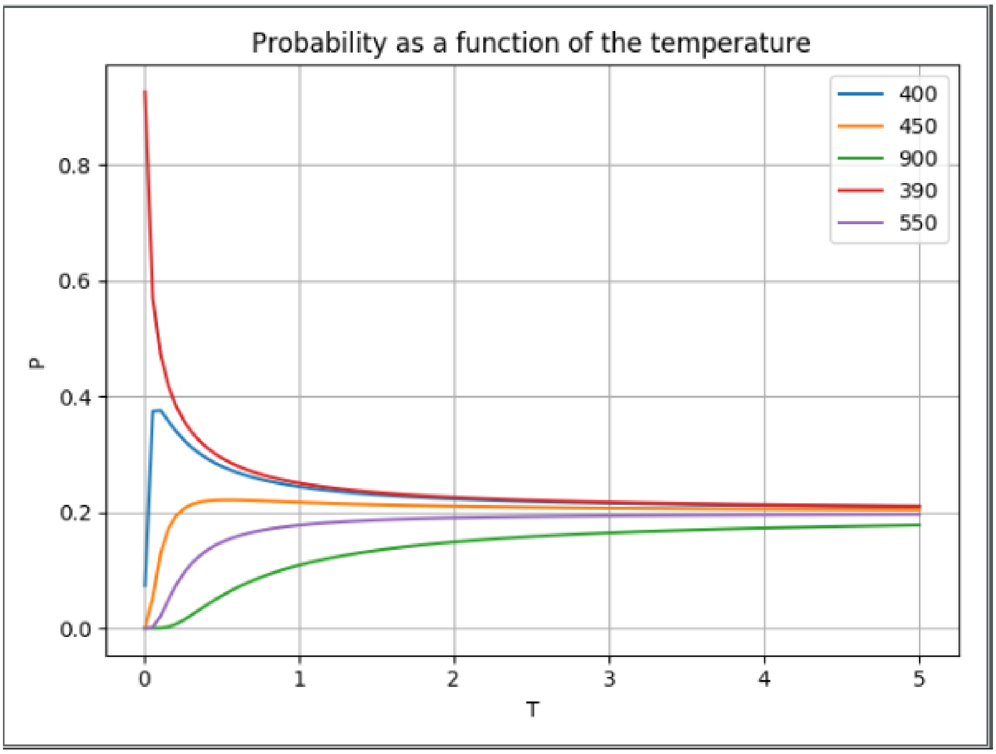
שאלה 19

ה- α הוא קבוע ולכן יוצא מחוץ לסכום ומקבלים:

$$\frac{\left(\frac{x_i}{\alpha}\right)^{\frac{-1}{T}}}{\sum\left(\frac{x_h}{\alpha}\right)^{\frac{-1}{T}}} = \frac{\left(\frac{1}{\alpha}\right)^{\frac{-1}{T}} * (x_i)^{\frac{-1}{T}}}{\left(\frac{1}{\alpha}\right)^{\frac{-1}{T}} * \sum (x_h)^{\frac{-1}{T}}} = \frac{(x_i)^{\frac{-1}{T}}}{\sum (x_h)^{\frac{-1}{T}}}$$

שאלה 20

נקבל את הגרף הבא:



שאלה 21

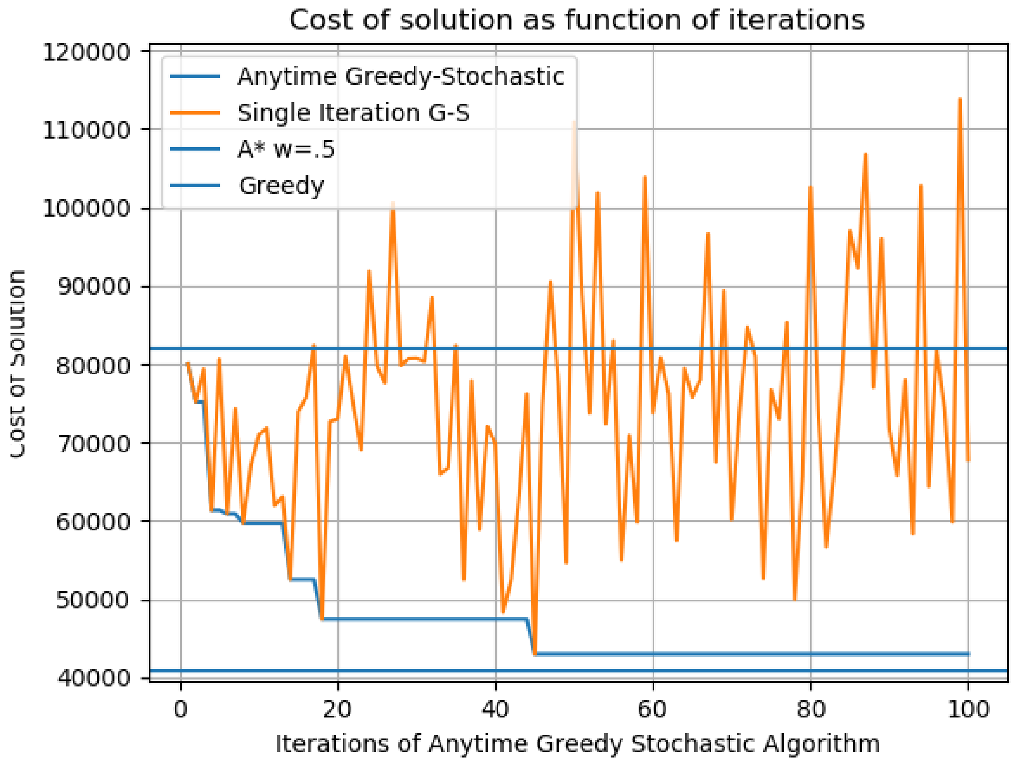
כאשר T שואף ל-0 נקבל המונה שואף ל-0 והמכנה שואף ל-1(במכנה כל המחוברים ישאפו ל-0 מלבד המינימלי ששווה ל- α ואז יהיה 1 זהותית ולכן כל הביטוי שואף ל-0, למעט המצב בו x_i הוא 1 ואז המונה שואף ל-1 וגם המכנה שואף ל-1 ולכן הביטוי הכולל שואף ל-1.

שאלה 22

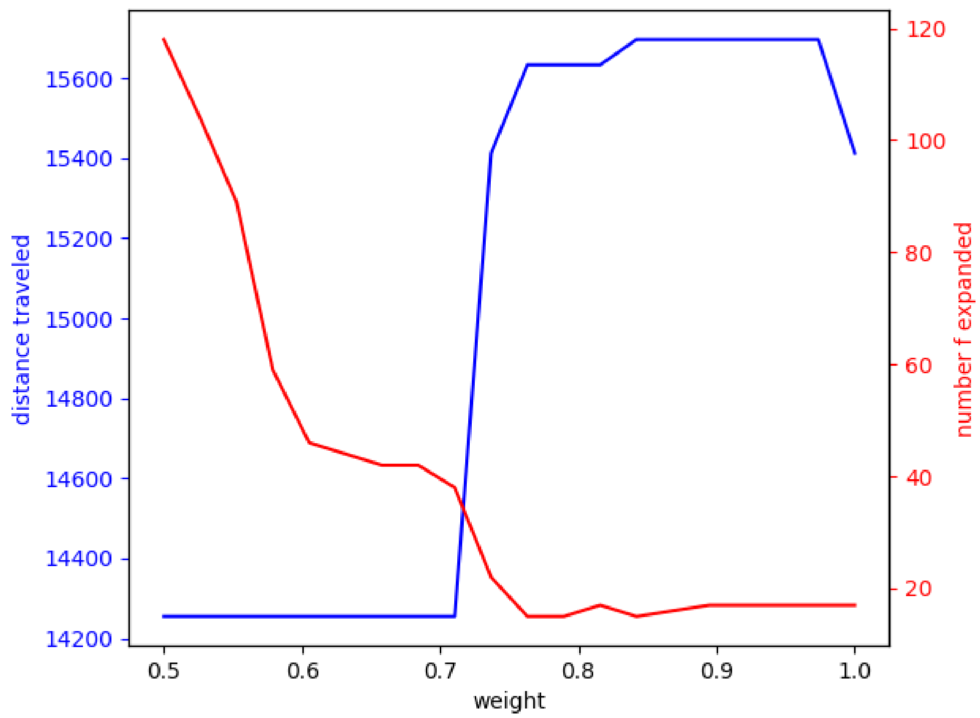
כאשר T שואף לאינסוף אז המונה שואף ל-1 והמכנה שואף ל-5(5 מחוברים שכל אחד מהם שואף ל-1) ולכן הגבול הוא 1/5=0.2.

ליאור רמתי 316490291 אסף אנטר 301019733

שאלה 24
הגרף שהתקבל בסעיף זה:



שאלה 26
הגרף שהתקבל בסעיף זה:



<u>שאלה 27</u>

הפתרון של Relaxed Deliveries מבוסס מרחק אווירי שידוע שהוא קביל, ולכן ברור שכל היוריסטיקה שמתבססת עליו גם תתן ערכים אופטימיים ולכן גם תהיה קבילה.

<u>שאלה 28</u>

```
StrictDeliveries(small_delivery)      A* (h=RelaxedProb, w=0.500)   time:   9.22   #dev: 80
     total_cost: 14254.79234   |path|: 8     path: [43516, 67260, 17719, 43454, 43217, 328
63, 7873, 42607]   gas-stations: [17719, 32863]
```

בסעיף 26, w=.5 נתן תוצאה של dist=14254.8, שזה דומה למרחק שקיבלנו בסעיף 28, אבל זה דרש פיתוח של 118 צמתים במקום רק 80 בסעיף 28. הw עבורו פיתחנו קמות דומה של צמתים, w=0.579 פיתח 59 צמתים ונתן מסלול סופי בעורך 14254.8. לגבי זמן הרצה, סעיף 28 לקח 9.22 שניות , וסעיף 26 לקח X שניות עבור w=.5, ו-Y שניות עבור w=.57894

<u>פרק 2</u>

<u>א</u>

נפריד למקרים: במקרה ש $Applicable_h(s)\ is\ True$ קבילות נובעת מהקבילות של h, ובמקרה השני, $h_0(h,s)=0$,
כלומר $Cost(s) > \delta > h_0(h,s)=0$ ולכן h_0 עומד בדרישות של קבילות גם במקרים האלה.

<u>ב</u>

נגדיר היוריסטיקה h_1 להיות שקולה לh_0 אבל במקרה בו $Applicable_h(s)\ is\ not\ True$ נגדיר שההיוריסטיקה מחזירה את $max(h_0(parent) - cost, 0)$, הערך של ההיוריסטיקה הישנה עבור ההורה פחות המחיר של המעבר מאב לבן (או 0). בנוסף, נגדיר שההיוריסטיקה מחזירה 0 עבור מצב סופי, שניתן לבדוק בעזרת הפונקציה $isGoal(s)$. ברור שמתקיים ש $\forall s: h_1(s) \geq h_0(s)$, ונוכיח בשלילה ש$h_1$ לא קבילה.
נניח קיים S כך ש-$Cost(S) < h_1(S)$, ברור ש-$Applicable(S)\ is\ not\ True$ וגם $isGoal(S)\ is\ not\ True$. נניח שגם $Applicable_h(S.parent)\ is\ True$ (אחרת $h_1(S)=0$ וברור שמצב זה לא שובר קבילות) ונקבל:

$$h(S.parent) - cost(S.parent -> S) > Cost(S) \Rightarrow$$
$$h(S.parent) > Cost(S) + cost(S.parent -> S) = Cost(S.parent)$$

ולכן נקבל שגם h לא קבילה, בסתירה להנחה.

<u>ג</u>

בגלל שבסעיף זה לא ידוע צורת מרחב המצבים, נשתמש בהורה עם הערך המינימלי שלא שווה ל0, במקום בהורה היחיד. בהינתן שינוי זה, שער הלוגיקה של סעיף ב נשמר.

<u>ד</u>

קיים אלגוריתם יותר מוצלח. *IDA עם אותו היוריסטיקה ($h_0(h', s)$). לפי הגדרת *IDA, ההיוריסטיקה קבילה, ולכן גם האלגוריתם קביל. בנוסף, הוא מפתח פחות צמתחם מ-*A. בשתי האלגוריתמים, כיון ש-$\forall s \neq s_i\colon h_0(s) = 0$ מתקיים שאנחנו בעצם מריצים Uniform Cost כי אחרי שאנחנו מפתחים את הצומת הראשון, $f(s) = g(s) + h_0(s) = Cost(s) + 0$. *IDA מרוויח מזה שהוא לא מפתח את הצמתים שבהם f (או במקרה שלנו, Cost) עובר את $h_0(s_i)$.