בינה מלאכותית

תרגיל בית 1

מגישים:

יקיר חלץ 305028441

גל פלייסיג 302912985

פרק ראשון – משלוחי פיצה

חלק א׳ – מבוא והנחיות

|  |  |  |
| --- | --- | --- |
| **k value** | **Without fueling** | **With fueling (l = 5)** |
| **1** | 1 | 5 |
| **2** | 2 | 50 |
| **3** | 6 | 750 |
| **4** | 24 | 15000 |
| **5** | 120 | 375000 |
| **6** | 720 | 11250000 |
| **7** | 5040 | 393750000 |
| **8** | 40320 | 15750000000 |
| **9** | 362880 | 708750000000 |
| **10** | 3628800 | 35437500000000 |

חלק ב׳ – הגדרת מרחב החיפוש במפה

חלק ג׳ – הגדרת מרחב החיפוש של מסלולי נסיעת הטוסטוס

* 1. 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, ומקדם הסיעוף של כל צומת הינו k + l – 1.
  2. 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.

1. יתכנו מעגלים במרחב המצבים שלנו. לדוגמה עבור הגרף הבא כאשר V1 הוא צומת המתאים להזמנה t1 ו-V2 הוא צומת תחנת דלק. נניח ש-d0=5, ו-d\_refuel=10, ו-dist(v0, v2)=5. אזי המצב ההתחלתי יהיה:
   1. S0 = (v0, 5, {v1}, emptyset)
   2. מכאן נעבור לתחנת הדלק ונגיע למצב S1 = (v2, 10, {v1}, emptyset).
   3. מכאן נחזור למצב ההתחלתי ונגיע למצב S2 = (v0, 5, {v1}, emptyset).
   4. קל לראות ש-S0=S2, כלומר ישנו מעגל במרחב המצבים.

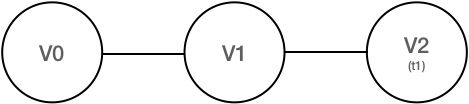


* 1. למיקום הנוכחי יש |V| אפשרויות.
  2. לכמות הדלק שנותרה יש d\_refuel אפשרויות.
  3. לקבוצת ההזמנות המחכות יש k אפשרויות.
  4. קבוצת ההזמנות הגמורות תלויה ב-T.

לסיכום, מספר המצבים במרחב זה הוא |V| \* d\_refuel \* k מצבים.

לא כל המצבים ישיגים. למשל, יתכן צומת v1, שהוא צומת המתאר הזמנה, אשר ישיג רק דרך צומת v2 שהינו צומת תחנת דלק, כאשר dist(v1, v2) = d\_refuel + 1. אזי גם אם יעצור השליח לתדלק לפני המעבר למסור את ההזמנה ב-v1 – הוא לעולם לא יוכל להגיע ל-v1. ולכן כל מצב המכיל את v1 כצומת נוכחי – אינו ישיג.

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



נניח ש-d0 = d\_refuel = 5. כמו-כן נניח ש-dist(v0, v1) = dist(v1, v2) = d\_refuel = 5.

אזי המצב ההתחלתי הינו S0=(v0, d\_refuel=5, {v2}, emptyset}.

כעת נעבור ל-v1 ונקבל את המצב: S1=(v1, 0, {v2}, emptyset}.

מצב זה הינו בור ישיג. נשים לב כי לא ניתן להמשיך ממנו לאף מצב אחר, ובנוסף זהו אינו מצב מטרה (v1 אינו צומת המתאים להזמנה).

1. פונקצית העוקב מוגדרת כדלהלן:

Succ((v1,d1,F1,T1)) = {(v2,d2,T2,F2) | v2 isin V, (v1,v2) isin E, Dist(v1,v2) <= d1, T2=T1, F2=F1}

Unity {(v2,d2,T2,F2) | v2 isin V, (v1,v2) isin E, Dist(v1,v2) <= d1, v2 isin Ord, T2=T1\_unity\_{v2}, F2=F1/{v1}}

הסבר: v2 הוא צומת ב-V, יש דרך המקשרת בין הצמתים v1,v2, הדלק הנותר במיכל לפני תחילת הנסיעה מספיק כדי לנסוע מ-v1 ל-v2 (מבחינת מרחק – קילומטרים), ובנוסף אחת משתי אפשרויות לגבי T ו-F: הצומת v2 הוא צומת המתאים להזמנה, ואז T2 הוא T1 בתוספת הצומת v2, ו-F2 הוא F1 ללא הצומת v2 (כלומר ההזמנה v2 הושלמה), **או** שהצומת v2 אינו צומת הזמנה ואז T2 = T1 ו-F2 = F1 כלומר הם נותרים ללא שינוי.

(האם להתייחס לכך ש-d2 יכול להיות d\_refuel אם v2 isin GasStations?)

1. בהנחה שאין שתי הזמנות במיקומים זהים, חסם תחתון לעומק המינימאלי של מצב מטרה כלשהו במרחב החיפוש הוא 1, וזאת מכיוון שיתכן מצב שבו מצומת ההתחלה יש דרך ישירה לכל אחד מצמתי ההזמנות, ואז כדי להשלים את כל ההזמנות נצא מצומת ההתחלה ונגיע אל צומת ההזמנה הראשון בעומק 1, ואז נשוב לצומת ההתחלה ומשם נצא לצומת ההזמנה הבא בעומק 1 שוב, ונחזור לצומת ההתחלה, וכן הלאה לגבי כל צמתי ההזמנה. ואז נגיע למצב: (v\_t, d, emptyset, Ord) 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.

חלק ד׳ – מתחילים לתכנת

להלן פלט הריצה לפני התיקון:

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]

להלן פלט הריצה המתוקנת:

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\*

1. מימוש בקוד.
2. מימוש בקוד.
3. 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]



1. 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.

1. מימוש בקוד.
2. 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 = emptyset שכן חלוקת כלל ההזמנות נסתיימה, ולפי הגדרת היוריסטיקה בסעיף ערכה הוא 0.

לסיכום, היוריסטיקה MaxAirDist קבילה.

1. מימוש בקוד.
2. 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]

1. 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]

1. מימוש בקוד. להלן פלט הריצה:



1. שינוי הסקאלה לא משנה את ההתפלגות לוקטור x נתון. הוכחה:
2. מימוש בקוד. להלן הגרף שהתקבל:



1. 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.
2. 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-1/5 = 0.2. 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.
3. מימוש בקוד.