מבוא לבינה מלאכותית

236501

תרגיל בית 1

10/12/2019

מגישים:

רוני אנגלנדר 312168354

נדב אורזך 311549455

1. הביטוי המתמטי המתאים למספר הסידורים האפשרי הוא: $\frac{(2k)!}{2^k}$ (בעזרת מקדם מולטינומי).
2.

K	#possiblePaths	Log_2(#possiblePaths)	Calculation time
5	113400	16.79	<sec
8	$8.17 * 10^{10}$	36.25	76.11[sec]
9	$1.25 * 10^{13}$	43.5	3.23[hours]
10	$2.37 * 10^{15}$	51.07	0.07[years]
11	$5.48 * 10^{17}$	58.929	16.21[years]
15	$8.09 * 10^{27}$	92.709	239058[million years]

3. הערך המקסימלי האפשרי של דרגת יציאה במרחב החיפוש הוא k, מכיוון שבמצב ההתחלתי קיימות k נקודות איסוף ולכן דרגת היציאה תהיה k. לא ייתכן כי דרגת היציאה תהיה גדולה מ-k כי לאחר כל איסוף יורדת נקודת אחת מהתחום של אופרטור האיסוף ונוספת נקודת פריקה אחת לתחום של אופרטור הפריקה. כלומר נוספת אפשרות אחת לדרגת היציאה של הצומת אך גם יורדת אחת. כמו כן לאחר פריקה של חבילה תרד דרגת היציאה ב-1 ולכן k הוא אכן הערך המקסימלי. הערך המינימלי האפשרי הוא 0 וזה אפשרי באחד ממצבי המטרה, בו נפרקו כל החבילות.
4. לא ייתכנו מעגלים מכיוון שנתון כי קבוצות נקודות האיסוף, הפריקה וההתחלה הן קבוצות זרות, ולכן כאשר נאספה או נפרקה חבילה בנקודה מסוימת לא ייתכן שנעבור בנקודה זו שוב, כלומר לא ייתכן מעגל.
5. מצב נוכחי מיוצג על ידי שלישייה של (מיקום נוכחית, קבוצת החבילות שבמשאית, קבוצת החבילות שנפרקו מהמשאית). לכן הביטוי עבור מספר המצבים האפשרי הוא:

$$1 + 2k * 3^{k-1}$$

הסבר – קיימים k משלוחים ועבור כל אחד מהמשלוחים האלו נקודת איסוף ונקודת פריקה. המשאית יכולה להימצא בכל אחת מנקודות אלו במצב מסוים, כלומר קיימות 2k אפשרויות עבור מיקום נוכחי. בנוסף כל חבילה יכולה להימצא באחת מבין שלוש האפשרויות הבאות – טרם נאספה, נאספה אך טרם הורדה, או נאספה והורדה. עבור החבילה המטופלת בנקודה הנוכחית אנחנו יודעים בדיוק מה מצבה לפי הגדרות המצבים. כלומר נותרו k-1 חבילות שעבור על אחת מהן קיימות 3 אפשרויות למצבן. סה"כ קיבלנו עד כה $2k * 3^{k-1}$ אפשרויות למצבים ולמספר זה נוסיף מצב אחד עבור המצב ההתחלתי. לסיכום נקבל את התשובה המוזכרת לעיל.
6. לא ייתכנן בורות ישיגים שאינם מצבי מטרה. אם הגענו לבור משמע לא קיימות קשתות יוצאות מהמצב, כלומר אין חבילות שנותרו לאיסוף, או חבילות שעל המשאית שצריכות להגיע לנק' פריקה שלהם. אך זהו בדיוק מצב מטרה, שכל החבילות נאספו ונפרקו. נוסיף כי ייתכן תחת הגבלת TruckCapacity שיווצרו בורות ישיגים שהם לא מצב מטרה, כאשר במצב בו המשאית ריקה, ונותר לנו חבילות לאסוף אך כל נקודת איסוף שטרם עברנו בה מספר החבילות בה גדול מהTruckCapacity.
7. כל מסלול מנק' ההתחלה אל מצב סופי באורך 2k. אנו מניחים שכל נק' האיסוף והפריקה זרות, ועל מנת לאסוף ולפרוק את כל החבילות צריך לעבור בכל נק' האיסוף והפריקה, לכן בספירת המצב ההתחלתי, נעבור ב-2k+1 מצבים עד שנגיע למצב סופי – בו אספנו ופרקנו את כל החבילות, כלומר אורך המסלול הוא 2k קשתות.
8. הגרף הינו DAG, גרף חסר מעגלים, לפי ההסבר בסעיף 4. בנוסף הוא גרף מכוון, כי כל קשת *(u,v)* היא קשת מכוונת שמתקבלת כאשר אנו בצומת *u* מפעילים אופרטור להגיע לצומת *v*.

.9

$$Succ(s) = \left\{(di.pick, s.loaded \cup \{di\}, s.dropped \middle| \begin{array}{c} di \notin s.loaded \cup s.dropped \ \wedge \\ di.pkgs \leq TruckCapacity - \sum_{d \in s.loaded} d.pkgs \end{array}\right\} \cup \{(di.drop, s.loaded \setminus \{di\}, s.dropped \cup \{di\} | di \notin s.dropped\}$$

פונקציית העוקב פועלת באופן הבא:

- אם ההובלה ה – i לא נמצאת על המשאית ולא סופקה בעבר ויש מספיק מקום פנוי במשאית עבור ההעמסת כל החבילות של ההובלה הזו – קיים מצב עוקב עבורו נאסוף את החבילה ה-i. המיקום הנוכחי יהיה di.pick, ונוסיף את החבילה למשאית.
- אם ההובלה ה – i נמצאת כרגע על המשאית – קיים מצב עוקב עבורו נפרוק את החבילה ה-i. המיקום הנוכחי יהיה di.drop ונסיר את החבילה מהמשאית.

.10 **<u>סעיף f:</u>**

פלט הריצה המתוקנת:

```
StreetsMap(src: 54 dst: 549)        UniformCost              time:   0.49   #dev: 17354
|space|: 17514    total_g_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]
```

סעיף g:

```
@dataclass(frozen=True)
class MapState(GraphProblemState):
```

משתנה MapState הוא מסוג GraphProblemSate והוא מייצג מיקום נוכחי במפה. הוא מכיל שדה יחיד, הjunction id שמגדיר את מיקומנו כרגע. במידה וטיפוס זה לא היה מוגדר כ-frozen, היינו יכולים לשנות את הjunction id ובכך לשנות את מרחב החיפוש שלנו ולהשפיע על הפתרון, ואף לגרום לפתרון שגוי.

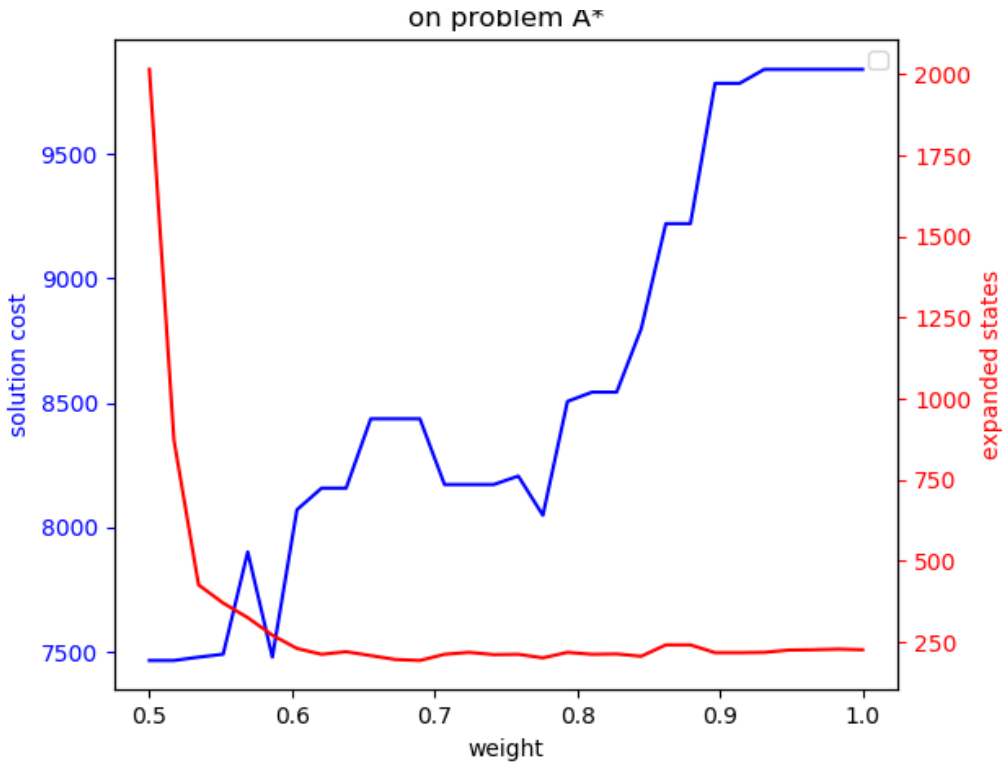
לדוג', בפונק' expand_state_with_costs , היא מפתחת את הsuccessors של הצומת הנוכחי (מקבלת משתנה מטיפוס GraphProblemState), אם היינו משנים את הid בGraphProblemState היינו מקבלים תוצאות חישוב שגויות לsuccessors בexpand_state_with_costs.

13. להלן:

```
StreetsMap(src: 54 dst: 549)      A* (h=AirDist, w=0.500)      time:  0.10  #dev:
2015   |space|: 2229    total_g_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]
```

מספר פיתוחי המצבים היחסי שחסכנו לעומת הריצה העיוורת הוא – 7.61

.14



קיבלנו גרף בו העקומה הכחולה מתארת את טיב הפתרון כתלות במשקל, ואילו העקומה האדומה מתארת את כמות הצמתים שפותחו כתלות במשקל. ניתן לראות כי ככל שהמשקל עולה, עלות הפתרון עולה ואילו כמות הצמתים שמפתחים קטנה. ניתן לראות כי העקומה האדומה מפסיקה לרדת החל מ- w=0.6 וניתן להבין כי זה מכיוון שעבור ערך זה מספר הצמתים המפותחים הוא אופטימלי.

הכלל אצבע "ככל ש-w קטן יותר כך הפתרון איכותי יותר ומס' הפיתוחים גדול יותר" נכון לרוב המשקלים המיוצגים בגרף, אך לדוג' עבור W~0.59 מתקיים שטיב הפתרון יותר טוב ממשקל של W~0.57 וגם כמות הצמתים שפותחו יותר קטנה, אך המשקל יותר גדול בניגוד לכלל האצבע.

המשקל W~0.59 הוא המשקל בו היינו בוחרים לאור תוצאות הגרף, כיוון שבשכלול 2 הפרמטרים (טיב הפתרון וכמות המצבים שפותחו) הפתרון האופטימלי שמתקבל הוא בערך זה.

19. נוכיח כי ההיוריסטיקה TruckDeliveriesMaxAirDistHeuristic הינה קבילה. נסמן ב$h(n)$ את תוצאת ההיוריסטיקה הנ"ל ונראה כי מתקיים $0 \leq h(n) \leq h^*(n)$. h מחשבת מרחק אווירי מקס' בין כל זוג צמתים ע"י שימוש בפונק' get_air_distance_between_junctions. כיוון שהמרחק אווירי תמיד אי שלילי, והפונק' הנ"ל מחזירה את המרחק האווירי המקס' שהיא מוצאת אז בפרט היא מחזירה מספר אי שלילי, ולכן מתקיים בהכרח $0 \leq h(n)$.

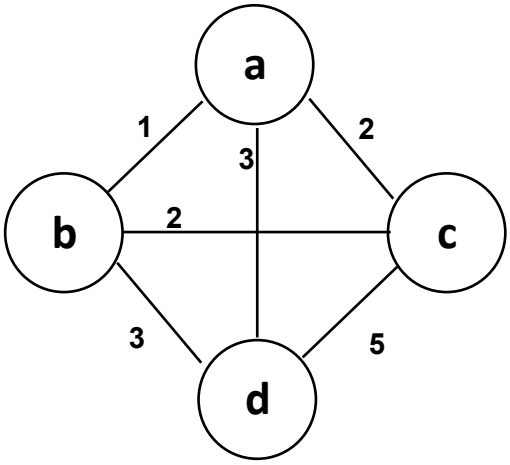
הפונק' מחזירה מרחק בין 2 צמתים במסלול שנותר למשאית לעבור, שהמרחק האווירי ביניהם הוא מקס' מכל צמדי הצמתים האפשריים במסלול, נסמן את זוג הצמתים הללו a,b.
תהי $h^*(n)$ ההיוריסטיקה המושלמת, אזי המסלול שהיא מחשבת חייב לעבור בשני הצמתים הללו, בה"כ הוא עובר בa לפני b, ויש שתי אפשרויות למעבר.

אפשרות ראשונה שמהצומת a נעבור ישירות לצומת b, ובמקרה הזה המרחק אשר תחזיר $h^*(n)$ יהיה שווה למרחק ש$h(n)$ תחזיר ויתקיים $h^*(n) = h(n)$.

אפשרות שניה היא לעבור בין a לb בצמתים נוספים ולא במעבר ישיר, ואז במצב הזה המרחק שנעבור בהכרח יותר גדול מהמרחק האווירי בין a לb, ולכן $h^*(n) < h(n)$.

בסה"כ קיבלנו כי מתקיים $0 \leq h(n) \leq h^*(n)$ ולכן ההיוריסטיקה קבילה.

22. נראה כי ההיוריסטיקה TruckDeliveriesSumAirDistHeuristic אינה קבילה ע"י דוגמא נגדית. נסתכל על המסלול שנותר לעבור הבא, כאשר a הצומת הנוכחית: (כל הצמתים שצריך לעבור בהם והכבישים המחברים אותן)

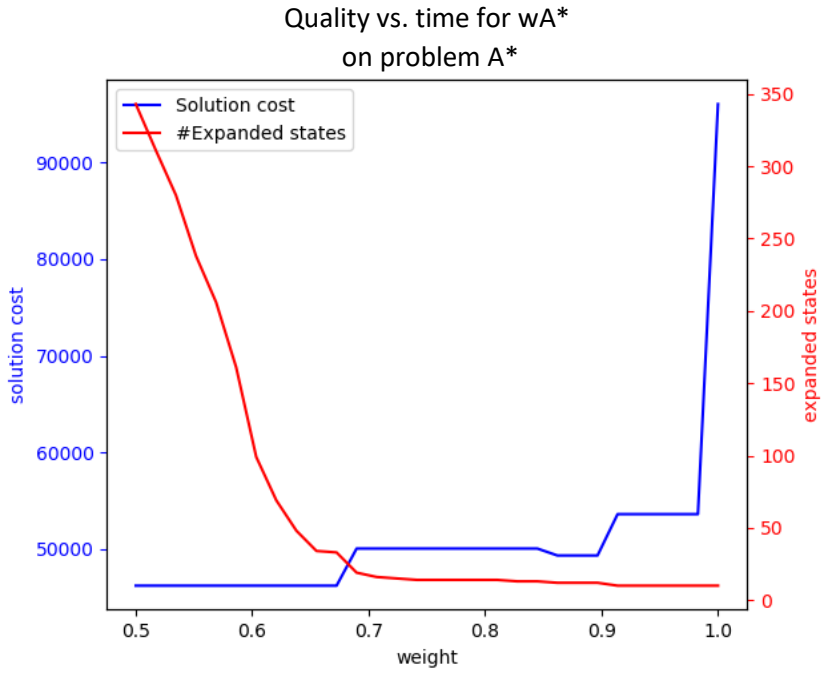


נסמן ב$h(n)$ את תוצאת ההיוריסטיקה הנ"ל. המסלול שיבחר לפי החישוב שלה הוא a->b>c>d ונקבל ש $h(n) = 8$. אך קיים מסלול אופטימלי עובר בצמתים *a->c->b->d* ועבורו נקבל $h^*(n) = 7$, כלומר $h(n) > h^*(n)$ ולכן ההיוריסטיקה אינה קבילה.

25. נוכיח כי ההיוריסטיקה TruckDeliveriesMSTAirDistHeuristic הינה קבילה. נסמן ב$h(n)$ את תוצאת ההיוריסטיקה הנ"ל ונראה כי מתקיים $0 \leq h(n) \leq h^*(n)$. h מחשבת את משקל העפ"מ של גרף הצמתים שנותר לנו לעבור בהם, כאשר משקל כל קשת הוא המרחק האווירי בין 2 צמתי הקשת. נסמן את הגרף G ואת משקל העפ"מ ש-*h* מחשבת - w(T). מרחק אווירי בין כל זוג צמתים בבעיה הוא אי שלילי, ולכן סכום משקלי קשתות (כל קב' של קשתות בגרף) הוא אי שלילי, ולכן מתקיים בהכרח $0 \leq h(n)$.

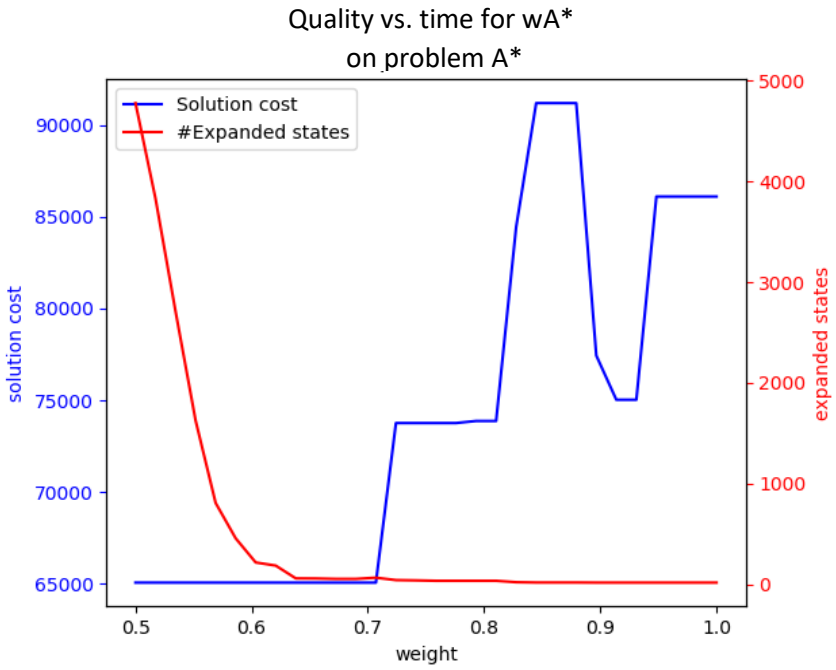
נניח בשלילה כי $h(n)$ אינה קבילה, משמע קיים פתרון אופטימלי המקיים $h(n) > h^*(n)$. הפתרון $h^*(n)$ הוא משקל עפ"מ של גרף G. נסמן את משקל העפ"מ הנ"ל w(T'), כלומר קיבלנו ש w(T') > w(T). אך זה בסתירה לכך ש$h(n)$ מחשבת משקל של עץ פורש מינימלי של G. ולכן $h(n)$ אכן מחשבת את הפתרון האופטימלי, משמע ההיוריסטיקה קבילה.

26. להלן הגרף עבור הרצת *run_astar_for_weights_in_range()* לבעיה *small_delivery_problem_with_distance_cost* כאשר ההיוריסטיקה לפיה החישוב מתבצע היא *TruckDeliveriesMSTAirDistHeuristic*:



ניתן לראות כאן כי ככל שהמשקל של הפונקציה היוריסטית גדל, כך יורדת כמות הפיתוחים בגרף ואיכות הפתרון יורדת. מגמת הגרף תואמת ברובה את "כלל האצבע" אך כפי שהוסבר בסעיף 14 נציין כי זו המגמה הכללית והכלל לא נכון לכל נקודה בעקום באופן גורף. היינו בוחרים ערך של w בטווח של [0.67,0.68] מכיוון שבשכלול 2 הפרמטרים (טיב הפתרון וכמות המצבים שפותחו) הפתרון האופטימלי שמתקבל הוא בטווח זה.

להלן הגרף עבור הרצת *run_astar_for_weights_in_range()* לבעיה *moderate_delivery_problem_with_distance_cost* כאשר ההיוריסטיקה לפיה החישוב מתבצע היא *TruckDeliveriesSumAirDistHeuristic*:



בגרף זה ניתן לראות כי המגמה של טיב הפתרון אינה עולה באופן קבוע ככל שה-w גדל, למרות שמגמת פיתוח הצמתים היא מגמה יורדת כפי שציפינו. ניתן להסביר את ההתנהגות החריגה של הגרף כתוצאה מכך שלפי סעיף 22 ההיורסטיקה הנוכחית אינה קבילה. היינו בוחרים ערך של W~0.65, משקל עבורו 2 המדדים מתלכדים בערך אופטימלי.

29. התוצאות הרלוונטיות:

חיפוש לפי זמן

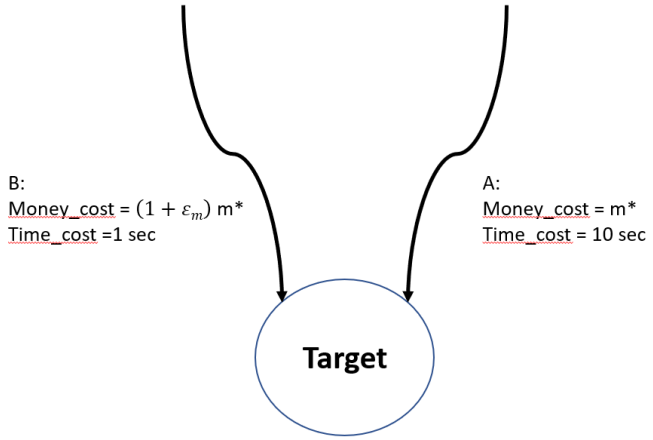
Deliveries(small_delivery(5):Time) A* (h=TruckDeliveriesMSTAirDist, w=0.500)
time: 34.94 #dev: 425 |space|: 562 total_g_cost: 30.18082 total_cost: DeliveryCost(dist= 46771.762 meter, time= 30.181 minutes, money= 151.961 nis) |path|: 11

חיפוש לפי מחיר

Deliveries(small_delivery(5):Money) A* (h=TruckDeliveriesMSTAirDist, w=0.500)
time: 61.48 #dev: 518 |space|: 591 total_g_cost: 104.23259 total_cost: DeliveryCost(dist= 46763.990 meter, time= 35.248 minutes, money= 104.233 nis) |path|: 11

ניתן לראות כי הפתרון המתקבל בכל הרצה מחשב לפי המדד הרלוונטי בעזרת פונקציית העלות המתאימה. כלומר, בחישוב הבעיה בעזרת פונקציית הזמן, עלות הזמן שהתקבלה היא 30 דקות אבל לעומת זאת בעזרת פונקציית המחיר, עלות הזמן שהתקבלה היא 35 דקות. באותו אופן, בחישוב הבעיה בעזרת פונקציית המחיר, עלות המחיר שהתקבלה היא 104 ₪ אבל לעומת זאת בעזרת פונקציית הזמן, עלות המחיר שהתקבלה היא 152 ₪ . כצפוי קיבלנו פתרון טוב יותר לפי מדד מסוים כאשר השתמשנו בפונקציית העלות המתאימה עבור מדד זה.

30. הפתרון שהוצע אינו מחזיר פתרון אופטימלי. נראה זאת בעזרת דוגמא נגדית:



נשים לב כי עבור מצב הנ"ל האלגוריתם יבחר את אופרטור A מכיוון שהוא בעל המחיר המינימלי שנכנס למצב Target ולכן ערך הזמן עבור מצב זה יהיה 10sec (לפי אופרטור A). כמו כן המחיר עבור אופרטור B נמצא בתחום של ה-Focal אך לא יבחר מכיוון שיש אופרטור טוב יותר לאותו מצב לפי שיקולי מחיר בלבד. אבל כאשר נבחן את 2 המדדים לפי ההצעה לשילובם אופרטור B עדיף על A, ולכן הפתרון שהסטודנט עומר הציע אינו מחזיר פתרון אופטימלי.

31. נציע את התיקון הבא עבור *A: עבור כל מצב שנפתח, במידה ויש מספר מסלולים הנכנסים לאותו מצב בעלי מחירי עלות שונים, במקום לשמור את הטוב ביותר בלבד, נשמור סט עבור אותו מצב עם כל המסלולים הנכנסים אליו. כאשר נבוא לבחור את המצב הבא שנפתח, נגדיר את סביבת האפסילון (FOCAL), לפי המצב בעל עלות המחיר המינימלית (נבחר אותו מבין כל הסטים שהגדרנו שנמצאים בחזית החיפוש), ונכניס לקבוצת ה-FOCAL את כל המצבים מכל הסטים שבסביבת אפסילון זו. לאחר מכן, נבחר את המצב בעל עלות הזמן המינימלית מבין קבוצה זו, מה שיספק את הבחירה המיטבית לפי המדד שקיבלנו בסעיף הקודם ויפתור את הבעיה שהצגנו. החיסרון המרכזי באלגוריתם שהצענו הוא מבחינת ביצועים – המיון שנבצע בכל בחירת מצב לפיתוח יבוצע על קבוצה גדולה יותר, מכיוון שעבור כל מצב בחזית החיפוש אנחנו מאפשרים לבחור מקבוצת המסלולים שנכנסים למצב זה ולא רק מסלול יחיד.

33. התוצאות הרלוונטיות:

*<u>*A רגיל (לצורך השוואה)</u>*

Deliveries(moderate_delivery(8):Distance) A* (h=TruckDeliveriesMSTAirDist, w=0.500) time: 21.85 #dev: 6376 |space|: 9045 total_g_cost: 65062.81195 total_cost: DeliveryCost(dist= 65062.812 meter, time= 42.946 minutes, money= 210.589 nis) |path|: 17

*<u>A*epsilon</u>*

Deliveries(moderate_delivery(8):Distance) A*eps (h=TruckDeliveriesMSTAirDist, w=0.500) time: 62.29 #dev: 6339 |space|: 9025 total_g_cost: 65062.81195 total_cost: DeliveryCost(dist= 65062.812 meter, time= 42.946 minutes, money= 210.589 nis) |path|: 17

כמות הפיתוחים שחסכנו היא 37 צמתים. ידענו לצפות כי כמות הפיתוחים תהיה טובה יותר מכיוון שלפי עקרון האלגוריתם של A*eps אנחנו מוכנים לוותר עד כדי אפסילון על איכות הפתרון בתמורה לקבלת פתרון מהיר יותר. נציין כי במקרה זה איכות הפתרון כלל לא נפגעה לעומת חישוב ה-*A הרגיל.

35. התוצאות הרלוונטיות:

*<u>*Anytime-A עבור k=8 בעזרת היוריסטיקת עפ"מ</u>*

Deliveries(moderate_delivery(8):Distance) Anytime-A* (h=TruckDeliveriesMSTAirDist, w=0.688) time: 2.32 #dev: 335 |space|: 200 total_g_cost: 65459.89155 total_cost: DeliveryCost(dist= 65459.892 meter, time= 43.581 minutes, money= 212.405 nis) |path|: 17

*<u>*Anytime-A עבור k=15 בעזרת היוריסטיקת סכום מרחקים</u>*

Deliveries(big_delivery(15):Distance) Anytime-A* (h=TruckDeliveriesSumAirDist, w=0.844) time: 150.70 #dev: 2030 |space|: 1861 total_g_cost: 155572.84122 total_cost: DeliveryCost(dist= 155572.841 meter, time= 103.868 minutes, money= 508.621 nis) |path|: 31

*<u>*Anytime-A עבור k=15 בעזרת היוריסטיקת עפ"מ</u>*

Deliveries(big_delivery(15):Distance) Anytime-A* (h=TruckDeliveriesMSTAirDist, w=0.875) time: 83.03 #dev: 2400 |space|: 1869 total_g_cost: 147869.82115 total_cost: DeliveryCost(dist= 147869.821 meter, time= 98.608 minutes, money= 483.292 nis) |path|: 31

לפי חלק א' עבור הרצת *A על הבעיות moderateDelivery, bigDelivery, בעיות בעלות ערך k=8,15 בהתאמה, זמן החישוב הנדרש לעבור על כל הסידורים האפשריים מאוד גדול עבור k=8 ואילו עבור k=15 הזמן הנדרש אינו אפשרי ליישום. *Anytime A מאפשר לנו להגביל את זמן החישוב על חשבון טיב הפתרון. במקרה שלנו אכן זמן החישוב ירד (לעומת חלק א) וקיבלנו פתרון מסוים, שניתן להניח כי הוא אינו הטוב ביותר.