מבוא לבינה מלאכותית

חורף תשפ"ד

מספר קורס: 02360501

תרגיל בית 1

אלעד גרוס

ת.ז: 213456932

גלעד שמרלר

ת.ז: 212139240

שאלה 1 – מבוא

סעיף 2

עבור סביבת כדורי הדרקון נגדיר מרחב חיפוש באופן הבא: (S, O, I, G) כאשר:

נניח שגודלו של הלוח הוא $n \times m$.

$$S = \{(i, d_1, d_2) \mid i \in [0, nm-1], d_k \in \{True, False\}, k \in \{1,2\}\}$$

$$O = \{0,1,2,3\}$$

$$I = (0, False, False)$$

$$G = \{(nm-1, True, True)\}$$

במקרה המתואר בשאלה:

$$G = \{(63, True, True)\}$$

$$|S| = nm \cdot 2 \cdot 2 = 4nm$$

ובמקרה שלנו (המופיע במחברת) בו $n = m = 8$, נקבל כי $|S| = 4 \cdot 8 \cdot 8 = 256$ מצבים.

סעיף 3

$$Domain(2) = \{s \mid s \in S, s \text{ is not } H\}$$

סעיף 4

הפונקציה Succ תחזיר לנו עבור המצב ההתחלתי את קבוצת המצבים העוקבים:

$$Succ\big((0, False, False)\big) = \{(0, False, False), (1, False, False), (8, False, False)\}$$

סעיף 5

כן, קיימים מעגלים במרחב החיפוש, מכיוון שאנחנו יכולים לדוגמה ללכת כלפי מטה (במקום שחוקי לעשות כך ושקיים משבצת מלמטה), ולאחר מכן ללכת למעלה (שוב במקרה בו זה אפשרי – שלא הגענו לבור לדוגמה). כך בעצם ביצענו מעגל בכך שחזרנו למצב בו היינו קודם לכן, ונוכל לעשות זאת אינסוף פעמים.

סעיף 6

מקדם הסיעוף הינו 4 שכן ניתן להגיע מכל מצב לכל היותר ל-4 מצבים שונים ע"י תזוזה בלוח ב-4 כיוונים אפשריים למעלה, למטה, ימינה ושמאלה.

סעיף 7

עבור סוכן כללי, במקרה הגרוע ביותר, מספר הפעולות אותן הוא נדרש לבצע אינו חסום. מכיוון שהסוכן יכול ללכת במעגל (שכן ראינו שקיימים כאלה במרחב החיפוש שלנו). לכן, לכל מספר טבעי $n \in \mathbb{N}$, נוכל להראות מקרה בו הסוכן ילך במעגלים ויבצע יותר מ-n עד אשר הוא יגיע אל המצב הסופי.

סעיף 8

עבור סוכן כללי, במקרה הטוב ביותר יידרשו לסוכן בדיוק 16 פעולות, מכיוון שהוא ילך כלפי מטה עד אשר יגיע לכדור הראשון, לאחר מכן יפנה ימינה אל עבר הכדור השני תוך שהוא עוקף את הבור שנמצא בדרך מלמעלה, אוסף את הכדור ולאחר מכן מגיע אל נקודת הסיום.

רצף הפעולות אותן הסוכן יעשה הוא: [0, 0, 0, 0, 0, 0, 1, 1, 1, 2, 1, 1, 0, 1, 1, 0].

סעיף 9

הטענה אינה נכונה, דוגמה נגדית:

S	L	L	D
F	F	F	D
G	F	F	**G**
F	F	F	F

חישוב עלות המסלול הראשון ל- G שנמצא בתא מספר 8 הוא:

$$cost(S, F, G) = cost(F) + cost(G) = 10 + 1 = 11$$

מצד שני עבור המסלול מ-S ל-G שנמצא בתא מספר 11 שאכן רחוק יותר (במונחי מרחק מנהטן) מהמצב ההתחלתי מתקיים:

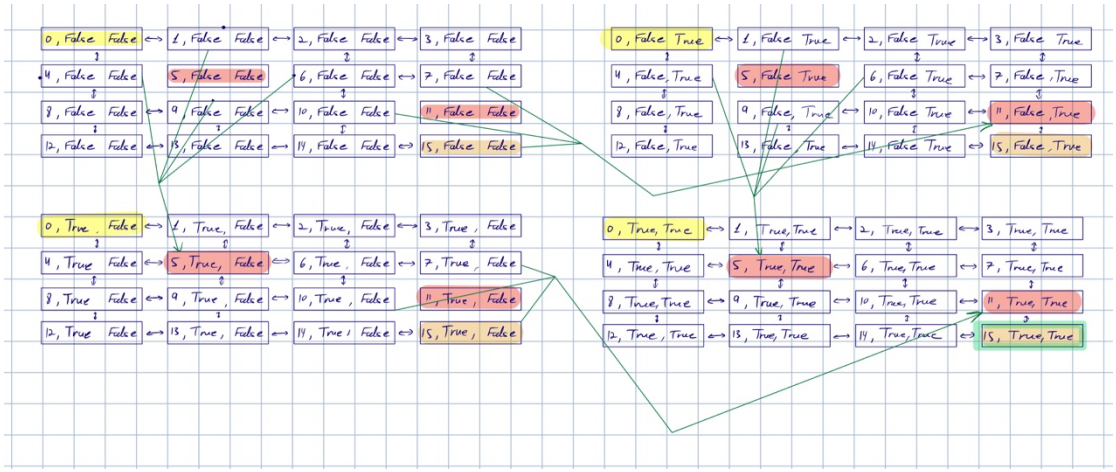
$$cost(S, L, L, D, D, G) = 2 \cdot cost(L) + 2 \cdot cost(D) + cost(G) = 5$$

שאלה 2 – Breadth First Search-G

סעיף 2

כפי שראינו בתרגול, BFS-G מתחזק מבנה נתונים שבו נשמרים כל המצבים בהם ביקרנו כבר, בניגוד ל-BFS על עץ שאינו מתחזק מבנה נתונים כזה. לכן, על מנת לדרוש ששני האלגוריתמים יפתחו וייצרו צמתים זהים באותו הסדר, יש לדרוש שלא נוכל להגיע אל אותו המצב משני מסלולים שונים בגרף המצבים (כלומר לדרוש שגרף המצבים יהיה עץ). מכיוון שאם נוכל להגיע אל אותו המצב משני מסלולים שונים, בהרצה של BFS נכניס אותו פעמיים אל OPEN ולכן נקבל שינוי לאחר מכן בסדר הפתיחה והייצור של הצמתים.

סעיף 3



סעיף 4

כפי שלמדנו בהרצאה, BFS-G מחזיר פתרון אופטימלי כאשר משקל כל הקשתות זהה. נרצה לקחת את הגרף המקורי $G = (V, E)$ וליצור ממנו גרף חדש $G' = (V', E')$ כך ש-"נפתח" כל קשת במשקל גדול מ-1 להיות שרוך של צמתים בו המשקל של כל צומת הוא 1 ויהיו באותו השרוך מספר קשתות בדיוק כמשקל אותה הקשת.

הסיבה לבניית גרף זה היא שאם בגרף המקורי היינו רוצים לעבור דרך קשת כלשהי שמשקלה k, אז כעת בגרף החדש נצטרך לעבור דרך k צמתים שונים שמשקל כל אחד מהם הוא 1, וכך בעצם יצרנו גרף חדש שמשקל כל הקשתות בו זהה (ולכן BFS-G עבורו הוא אופטימלי), אך הוא גם משמר את העובדה שעל מנת לעברו בקשת מסוימת בגרף המקורי היינו צריכים לעבור ב-k קשתות בגרף החדש – מכיוון שאותה קשת הפכה לשרוך שבו יש רק דרך אחת לעבור בו.

סעיף 5

צמתים שיפותחו: $N^2 - 2$

צמתים שייווצרו: N^2

במהלך ריצת האלגוריתם BFS-G מפתח את הצמתים שיוצאים מהתור כאשר אנחנו מכניסים את הצמתים לתור על פי המרחק שלהן (בצמתים, לא המשקל) מנקודת ההתחלה. לכן, במקרה שלנו, בו אנחנו מתחילים מהפינה השמאלית העליונה, נפתח כל פעם אלכסון שלם לפני שנעבור אל האלכסון הבא.

כאשר נגיע אל האלכסון האחד לפני האחרון – כלומר שיש בו רק שני צמתים, נפתח את אחד הצמתים שיצא ראשון מהמחסנית (במימוש במחברת – את הצומת הנמוך והשמאלי יותר), ואז בפיתוח זה נמצא את המצב הסופי אליו אנחנו רוצים להגיע. כלומר, יצרנו צמתים עבור כל אחד מהמצבים בגרף, לכן יצרנו סך הכל N^2 צמתים – כי יצרנו גם צמתים עבור המצב הסופי כאשר מצאנו אותו, וגם עבור המצב השני באלכסון האחד לפני האחרון למרות שלא פיתחנו אותו. כמו כן, פיתחנו את כל הצמתים מלבד הצומת השני באלכסון הזה וצומת המטרה, לכן סך הכל פיתחנו $N^2 - 2$ צמתים.

שאלה 3 – Depth First Search-G

סעיף 1

כפי שלמדנו בהרצאה, במקרה זה מכיוון שמרחב המצבים הוא סופי, לכן האלגוריתם שלם, מכיוון שאם קיים פתרון האלגוריתם ימצא אותו בסופו של דבר.

האלגוריתם אינו קביל, מכיוון שהאלגוריתם אינו בהכרח מוצא את הפתרון האופטימלי. לדוגמה עבור הלוח הבא, כאשר שני הכדורים יהיו גם הם בצומת המטרה (ולכן יש רק להגיע אל צומת המטרה):

S	L	L	L
F	F	F	L
F	F	F	L
F	F	F	**G**

אלגוריתם *DFS-G* עבור גרף זה ילך תחילה למטה עד שהוא ייתקע (כי זה סדר הפיתוח של הצמתים שלו המוגדר בשאלה) ולאחר מכן ילך ימינה עד שיגיע אל מצב המטרה (שוב כי זה סדר הפיתוח שמוגדר עבורו). מחיר הפתרון גדול יותר מאשר מחיר הפתרון אם נלך תחילה ימינה עד הסוף ולאחר מכן למטה עד שנגיע אל מצב המטרה. כלומר הראנו כי האלגוריתם אינו קביל.

סעיף 2

אלגוריתם *DFS* על עץ לא בהכרח היה מוצא פתרון במקרה שלנו, מכיוון שהאלגוריתם לא בודק אם הוא הולך לפתח צומת שהוא כבר ביקר בו בעבר. אנחנו ייצגנו את בעיית כדורי הדרקון כבעיה על גרף ויש בה מעגלים, ולכן אנחנו עלולים להיתקע בלולאה אינסופית. לדוגמה במקרה הבא ניתקע בלולאה אינסופית:

S	F	D	D
F	F	F	G
F	F	F	F (11)
F	F	F	F (15)

כפי שהסברנו בסעיף הקודם נלך מטה עד הסוף ואז ימינה עד הסוף, לאחר מכן, מכיוון שאנחנו לא יכולים ללכת לא למטה ולא ימינה, נעלה למעלה צעד אחד (וזאת לפי סדר הפעולות המוגדר בשאלה), ולאחר מכן נוכל שוב לרדת למטה, מכיוון שהאלגוריתם לא זוכר שהיינו כבר במצב הזה. כעת אנחנו ניתקע בלולאה אינסופית בין מצב 11 למצב 15.

סעיף 3

צמתים שיפותחו: $2N - 2$

צמתים שייווצרו: $4N - 5$

במהלך ריצת האלגוריתם DFS-G, הוא מפתח את הצמתים לפי סדר הפיתוח שהוגדר בתחילת המחברת, לכן תחילה נרד כלפי מטה עד הסוף ולאחר מכן נפנה ימינה עד הסוף ונגיע אל צומת המטרה. סך הכל פיתחנו במהלך הריצה הזו את כל N הצמתים שנמצאים בעמודה הימנית ביותר, ולאחר מכן עוד $N - 2$ צמתים בשורה התחתונה ביותר, מכיוון שאת צומת המטרה לא צריך לפתח ואת הצומת השמאלית ביותר כבר פיתחנו קודם לכן. כלומר סך הכל פיתחנו $N + N - 2 = 2N - 2$ צמתים.

בכל צומת שאנחנו מבקרים במהלך ריצת האלגוריתם אנחנו יוצרים גם את כל השכנים שלו. לכן בירידה מטה ניצור את N הצמתים בהם אנחנו נבקר, וכמו כן ניצור גם עוד את N הצמתים הסמוכים להם. לאחר מכן בהליכה ימינה על השורה התחתונה ביותר נבצע דבר דומה, ונפתח את כל הצמתים שבשורה מעליהם, חוץ מאת הצומת שמעל צומת המטרה, ואת הצמתים שמעל שני הצמתים השמאליים ביותר, מכיוון שכבר יצרנו אותם בירידה מטה. לכן סך הכל, אנחנו ניצור: $4N - 5$ צמתים.

סעיף 4

צמתים שיפותחו: $2N - 2$

צמתים שייווצרו: $2N - 1$

האלגוריתם backtracking DFS-G פועל באופן דומה מבחינת סדר הצמתים אליהם הוא ניגש ולכן כמות הצמתים שיפותחו תהיה זהה.

מבחינת כמות הצמתים שייווצרו, אז הוא יוצר צומת רק כאשר הוא צריך לפתח אותו, ולכן הוא לא ייצור את כל השכנים שהסברנו עליהם בסעיף הקודם, כי בשום שלב לא נצטרך להשתמש בהם. לכן, הוא ייצור את N השכנים במסלול מטה, ולאחר מכן עוד $N - 1$ צמתים עבור כל הצמתים שבשורה התחתונה (מלבד השמאלי ביותר שכבר יצרנו עבורו צומת). כלומר סך הכל $2N - 1$ צמתים.

שאלה 4 – ID-DFS

סעיף 1

a. כן, *ID-DFS* הוא אלגוריתם שלם. הוכחה: נניח שהפתרון האופטימלי x^* הוא בעומק k. בכל איטרציה של *ID-DFS* עם עומק $L < k$ לא נמצא פתרון, מכיוון שאם היינו מוצאים שם פתרון, אז הוא היה טוב יותר מאשר הפתרון האופטימלי וזו תהיה סתירה לאופטימליות x^*. כאשר נריץ את *ID-DFS* עם עומק $L = k$ אז נמצא את הפתרון x^* ונחזיר אותו, ובכך נחזיר את הפתרון האופטימלי. נשים לב כי אכן נמצא את הפתרון x^* מכיוון ש-*ID-DFS* הוא אלגוריתם שלם, ולכן הוא בהכרח ימצא את הפתרון.

b. כן, האלגוריתם במקרה זה יהיה קביל. הוכחה: נניח כי עומק הפתרון האופטימלי הוא k. אלגוריתם *ID-DFS* ירוץ על כל העומקים השונים עד k לפי הסדר, עד שהוא יגיע לעומק k ויחזיר את הפתרון האופטימלי שיש בעומק זה. נשים לב כי *ID-DFS* אכן ימצא את הפתרון האופטימלי בעומק k מכיוון שכפי שהוכחנו בסעיף הקודם, *ID-DFS* הוא שלם, ולכן הוא ימצא את הפתרון בעומק זה. כמו כן, האלגוריתם לא ימצא אף פתרון בהרצה עבור עומק $i < k$ מכיוון שאם נניח בשלילה שהיינו מוצאים פתרון בעומק זה, אז מכיוון שעל פי הנתון משקל כל הקשתות הוא 1, אז מצאנו פתרון שערכו טוב יותר מערך הפתרון האופטימלי וזו סתירה לאופטימליות הפתרון.

סעיף 3.a

נסמן ב-L את עומק הפתרון האופטימלי של הבעיה.

נציג דוגמה בה *ReverseDFS* עדיף על *ID-DFS*: במקרה בו מתקיים $L = D$, אז עדיף להריץ את *ReverseDFS* מכיוון שהוא ימצא את הפתרון האופטימלי כבר תוך איטרציה אחת של *DFS-L* (ולאחר מכן עוד איטרציה בשביל לוודא שאכן לא קיים פתרון טוב יותר), לעומת *ID-DFS* שיצטרך לבצע D איטרציות עד שהוא יוכל למצוא את הפתרון האופטימלי.

נציג דוגמה בה *ID-DFS* עדיף על *ReverseDFS*: במקרה בו מתקיים $L \ll D$, כמו לדוגמה מקרה בו $L = 1$. במקרה כזה, *ReverseDFS* יצטרך לבצע D איטרציות של *DFS-L* עד שהוא יוכל להבטיח כי הפתרון שהוא קיבל כבר בהתחלה הוא אופטימלי.

סעיף 3.b

```
function ReverseDFS (problem, D):
    L ← D
    result ← failure
    While Not Interrupted:
        new_result ← DFS-L (problem, L)
       if new_result = failure:
          break
       L ← new_result.length - 1
       result ← new_result

    return result
```

כלומר השינוי שביצענו בפונקציה הוא לא להקטין את גודל הפתרון שאנחנו מחפשים כל פעם באחד, אלא להגביל אותו להיות יותר קטן מאשר הפתרון הנוכחי שעד כה מצאנו. שיפור זה יכול לעזור לנו במקרה דומה

למקרה שתיארנו בסעיף הקודם בו ישנו פתרון יחיד בעומק 1 והחסם D אינו הדוק כלל. במקרה כזה נמצא את הפתרון בעומק 1, ואז ננסה לחפש פתרון שקטן יותר ממנו ונמצא שלא קיים כזה. כמו כן נשים לב כי שינוי זה אינו פוגע בכוח של האלגוריתם מכיוון שבגרסה הקודמת בכל מקרה היינו מבצעים חיפושים שהיו מיותרים כי הם היו מוצאים את אותו הפתרון, כל עוד לא הקטנו את L להיות מתחת לרף של הפתרון הקודם שמצאנו.

שאלה 6 – UCS

סעיף 1

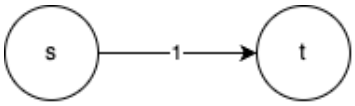
אלגוריתם *BFS* ואלגוריתם *UCS* יפעלו באותו האופן במקרה בו משקל כל הקשתות בגרף זהה. הסיבה לכך היא שכפי שלמדנו בהרצאה, *BFS* משתמש בתור רגיל בשביל לשמור את הצמתים אותם הוא רוצה לפתח בכל איטרציה, ואילו *UCS* שומר אותם בתור עדיפויות ממוינים לפי משקל המסלול המגיע אליהם מצומת המקור. במקרה בו ערכי הקשתות זהים, אז תור העדיפויות שנשמור יהיה בדיוק כמו תור רגיל, ולכן סדר פיתוח הצמתים במקרה זה יהיה לפי סדר הכניסה שלהם כמו ב-*BFS*.

סעיף 2

ראינו משפט בהרצאה שאלגוריתם *UCS* הוא שלם וקביל כאשר פונקציית המחיר חסומה מלמטה על ידי $\delta > 0$. זהו אכן המצב אצלנו מכיוון שעל פי טבלת המחירים המופיעה בתחילת המחברת ניתן לראות כי כל **המחירים** של הקשתות הם לפחות 1, לכן נוכל לבחור $\delta = \frac{1}{2} > 0$ לדוגמה, וכך נקבל על פי המשפט מההרצאה שהאלגוריתם שלם וקביל.

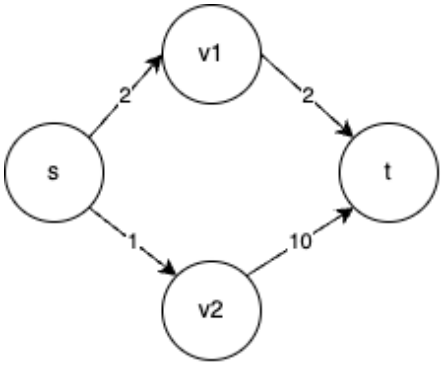
סעיף 3

דוגמה לגרף חיפוש שעבורו האלגוריתם השגוי יחזיר בכל מקרה את מחיר המסלול הקל ביותר:



הסיבה לכך היא שיש מסלול יחיד מצומת המקור אל צומת המטרה, ולכן האלגוריתם יחזיר את המסלול היחיד שקיים שהוא גם אופטימלי כמובן.

דוגמה לגרף חיפוש שעבורו האלגוריתם השגוי יחזיר מסלול שאינו המסלול הקל ביותר:



נסתכל על מסלול בין s ל-t. האלגוריתם השגוי יפעל בצורה הבאה:

1. s יצא מ-*OPEN* ויוכנסו הבנים שלו v_1, v_2 עם משקלי הקשתות המביאים אליהם.
2. v_2 יצא מ-*OPEN* לפני v_1 מכיוון שמשקל המסלול אליו קל יותר וניצור את הצומת עבור הילד שלו t.
3. נסיים את האלגוריתם בשלב זה, מכיוון שעל פי האלגוריתם השגוי אנחנו בודקים האם צומת מסוים הוא היעד בזמן היצירה שלו ולא בזמן הפיתוח שלו. לכן, בשלב הזה בוא יצרנו את צומת היעד, נסיים ונחזיר מסלול עם משקל $1 + 10 = 11$.

נשים לב שישנו מסלול קל יותר מ-s ל-t, המסלול $s \to v_1 \to v_2$ קל יותר מכיוון שמשקלו הוא $2 + 2 = 4$.

שאלה 7 - יוריסטיקות

הגדרה יוריסטיקה h תיקרא ε–קבילה אם קיים $\varepsilon \geqslant 1$ כך שלכל מצב $s \in S$ מתקיים $h(s) \leqslant \varepsilon \cdot h^*(s)$.
כאשר h^* הינה פונקציית המחיר המסלול האופטימלי מ- s לצומת היעד.

זהו פתרון השאלה כפי שהבנו אותה בהתחלה עם הקשר לתרגיל ולמשימה של לאסוף את הדרגול בול כיוון שצוין שמתסכלים על בעיית הניווט לכדור יחיד.

הוספנו פתרון נוסף של שאלה 7 בהמשך עבור מקרה כללי

סעיף 1

יהי $s \in S$ מתקיים כי המסלול הקצר ביותר הוא מרחק מנהטן בין הצומת לבין היעד ועל כן נקבל:
עבור $\varepsilon = 1$:

$$h_{MD}(s) = \varepsilon \cdot h^*(s) \Longrightarrow h_{MD}(s) \leqslant \varepsilon \cdot h^*(s)$$

סעיף 2

יהי $s \in S$ נגדיר $\varepsilon = 1$ ונקבל כי:

$$h(s) = \min\{|G_x - S_x|, |G_y - S_y|\} \leqslant |G_x - S_x| + |G_y - S_y| = h_{MD}(s) = \varepsilon \cdot h^*(s)$$

כיוון שהמינימום בין המרחק בציר x והמרחק בציר y קטן מסכום המרחקים אז זה מתקיים.

סעיף 3

יהי $s \in S$ ונגדיר $\varepsilon = 1$ ונקבל כי:

$$h(s) = ||\ |G_x - S_x| + |G_y - S_y|\ ||_3 \underset{\textit{triangle inequality of } L_p \textit{ norm}}{\leqslant} ||G_x - S_x||_3 + ||G_y - S_y||_3 =$$
$$|G_x - S_x| + |G_y - S_y| = h_{MD}(s) = \varepsilon \cdot h^*(s)$$

סעיף 4

יהי $s \in S$ מהגדרת ה-ε–קבילות של h_1 נקבל כי לכל s מתקיים:

$$h_1(s) \leqslant \varepsilon_1 \cdot h^*(s)$$

בנוסף מהגדרת ה-ε–קבילות של h_2 נקבל כי לכל s מתקיים:

$$h_2(s) \leqslant \varepsilon_2 \cdot h^*(s)$$

נגדיר $\varepsilon_3 = \varepsilon_1 + \varepsilon_2 \geqslant 1 + 1 = 2 \geqslant 1$, לכן מתקיים כי:

$$h_1(s) + h_2(s) \leqslant \varepsilon_1 h^*(s) + \varepsilon_2 h^*(s) = (\varepsilon_1 + \varepsilon_2)h^*(s) = \varepsilon_3 \cdot h^*(s)$$

קיבלנו את ההדוק ביותר שכן $\varepsilon_1, \varepsilon_2$ הם ההדוקים ביותר.

סעיפים 5-8 בפתרון של התרגיל 7 בעמודים הבאים

שאלה 7 - יוריסטיקות - מקרה כללי

הגדרה יוריסטיקה h תיקרא ε–קבילה אם קיים $\varepsilon \geqslant 1$ כך שלכל מצב $s \in S$ מתקיים $h(s) \leqslant \varepsilon \cdot h^*(s)$.
כאשר h^* הינה פונקציית המחיר המסלול האופטימלי מ- s לצומת היעד.

תחילה לפני הסעיפים נסמן:

$$h_e(s) = \sqrt{(G_x - S_x)^2 + (G_y - S_y)^2}$$

ונבחין כי מתקיים:

$$h_e(s) \leqslant h^*(s)$$

סעיף 1

יהי $s \in S$ מתקיים כי המסלול הקצר ביותר הוא המרחק האווירי לכן
עבור $\varepsilon = \sqrt{2}$:

$$h_{MD}(s) = |G_x - S_x| + |G_y - S_y| \underset{Cauchy-Schwarz\ inequality}{\leqslant}$$
$$\sqrt{2} \cdot \sqrt{(G_x - S_x)^2 + (G_y - S_y)^2} \leqslant \sqrt{2} \cdot h^*(s)$$
$$\Longrightarrow h_{MD}(s) \leqslant \varepsilon \cdot h^*(s)$$

זהו ההדוק ביותר שכן עבור $|G_x - S_x| = |G_y - S_y|$ נקבל כי האי שיוויון הופך לשיוויון ממש.

סעיף 2
יהי $s \in S$ נגדיר $\varepsilon = 1$ ונקבל כי:

$$h(s) = \min\{G_x - S_x, G_y - S_y\} \leqslant \min\{|G_x - S_x|, |G_y - S_y|\}$$
$$\underset{(*)}{\leqslant} 1 \cdot h_e(s) \leqslant 1 \cdot h^*(s)$$

המרחק האווירי גדול יותר מהמרחק שהוא רק על אחד הצירים כיוון שהוא הטלה שלו כתלות בזווית ולכן מעבר $(*)$ מתקיים.

סעיף 3

יהי $s \in S$ ונגדיר $\varepsilon = 1$ ונקבל כי:

$$h(s) = ||\ |G_x - S_x| + |G_y - S_y|\ ||_3 \underset{Cauchy-Schwarz\ inequality}{\leqslant} ||G_x - S_x||_3 + ||G_y - S_y||_2 =$$
$$h_e(s) \leqslant\ h^*(s) \leqslant \varepsilon \cdot h^*(s)$$

סעיף 4

יהי $s \in S$ מהגדרת ה-ε–קבילות של h_1 נקבל כי לכל s מתקיים:

$$h_1(s) \leqslant \varepsilon_1 \cdot h^*(s)$$

בנוסף מהגדרת ה-ε–קבילות של h_2 נקבל כי לכל s מתקיים:

$$h_2(s) \leqslant \varepsilon_2 \cdot h^*(s)$$

נגדיר $\varepsilon_3 = \varepsilon_1 + \varepsilon_2 \geqslant 1 + 1 = 2 \geqslant 1$, לכן מתקיים כי:

$$h_3(s) = h_1(s) + h_2(s) \leqslant \varepsilon_1 h^*(s) + \varepsilon_2 h^*(s) = (\varepsilon_1 + \varepsilon_2)h^*(s) = \varepsilon_3 \cdot h^*(s)$$

קיבלנו את ההדוק שכן $\varepsilon_1, \varepsilon_2$ הם ההדוקים ביותר.

בסעיפים 5-8 נאמר לחזור ולפתור בהקשר של לוח המשחק שלנו

סעיף 5

כעת $D = \{d_1, d_2\}$ ונגדיר יוריסטיקה חדשה:

$$h_{MSAP}(s) = \min\{h_{Manhatan}(s, g) | g \in G \cup D\}$$

היוריסטיקה הינה קביל שכן מתקיים לכל $s \in S$:

$$0 \underset{defenition\ of\ manhatan\ distance}{\leqslant} h_{MSAP}(s) =$$
$$\min\{h_{Manhatan}(s, g) | g \in G \cup D\} \underset{(*)}{\leqslant} h^*(s)$$

עתה אנחנו במקרה של הלוח המקורי ועל כן נקבל כי מרחק מנהטן הוא המרחק המינמלי האפשרי מצומת מסוימת לאחרת שכן המשקלים כולם גדולים מ- 1. כלומר $h_{MSAP}(s)$ מביא לנו את מרחק מנהטן הכי קטן ובנוסף לוקחים את המצב שהכי קרוב לכן כמובן שהערך הזה יהיה קטן מהאופטימלי שכן האופטימלי במקרה הכי טוב ירצה להתקדם ככמות צעדים של מרחק מנהטן על משקלים קטנים של 1 ולאו דווקא מתאפשר לו.

עוד הסבר נניח כי עדיין לא אספנו את 2 הכדורים לכן היוריסטיקה המושלמת תיתן לנו עלות של מסלול שעובר בין 2 הכדורים ואז מגיע ל- G אבל $h_{MSAP}(s)$ זה מרחק מנהטן לאחד הקרוב ביותר מבין הכדורים והמצב המקבל לכן בהכרח זה קטן יותר מעלות המסלול שעובר בין 2 בכדורים והמצב G כמו

כן שכל המשקלים גדולים או שווים ל-1. למשל אם כבר נאספו 2 הכדורים אז היוריסטיקה תחזיר את מרחק מנהטן למצב G שזה המסלול הכי קטן שאפשרי בלוח המשחק שלנו אם קיים מסלול שסכומי הקשתות רק 1 ולכן כמובן שקטן יותר מ h^*.

סעיף 6

כעת נעבור להוכחת עקביות (אנחנו מוכיחים עקביות וזה גורר קבילות לכן אין צורך בהוכחה של הסעיף הקודם).
מחלק למקרים עבור $s = s'$ מתקיים כי: $h(s) = h(s')$ ולכן:

$$h(s) - h(s') = 0 = cost(s, s')$$

עבור $s \neq s'$ נקבל כי $h(s) - h(s') \leqslant 1$ שכן $s' \in succ(s)$ כלומר פעולה חוקית שזה תזוזה בציר x או תזוזה בציר y לכן ההפרש בין מרחק מנהטן הנוכחי s למרחק של s' הוא לכל היותר 1. והמחיר הוא גדול או שווה ל-1 לפי ההגדרות של לוח המשחק במידה והמצב השתנה לכן מתקיים:

$$h(s) - h(s') \leqslant 1 \leqslant cost(s, s')$$

נבחין כי עבור 2 מצבים:
לקחנו את אותו מצב לחישוב $h_{MSAP}(s)$ או שנבחר מצב חדש לחישוב של $h_{MSAP}(s)$ אז מתקיים $h(s) - h(s') \leqslant 1$.

סעיף 7

נגדיר

$$h_{new}(s) = \max\{h_{Manhatan}(s, g) | g \in G \cup D\}$$

נראה כי היוריסטיקה אינה קבילה ע"י דוגמה נגדית:
נקח את לוח המשחק הבא ונניח כי כבר אספנו את 2 כדורי הדרקון:

$$\begin{bmatrix} S & \mathbf{D} & F & F \\ \mathbf{D} & F & F & F \\ F & F & F & F \\ \mathbf{G_1} & F & F & \mathbf{G_2} \end{bmatrix}$$

ראינו כי עבור $G_2 = (15, True, True)$ מתקיים $h(G_2) = 0$ עבור יוריסטיקה קבילה אבל במקרה כאן נקבל

$$h(G_1) = 3$$

בפיאצה נאמר כי יכול להיות יותר ממצב G יחיד לכן במקרה הזה $G = \{G_1, G_2\}$.

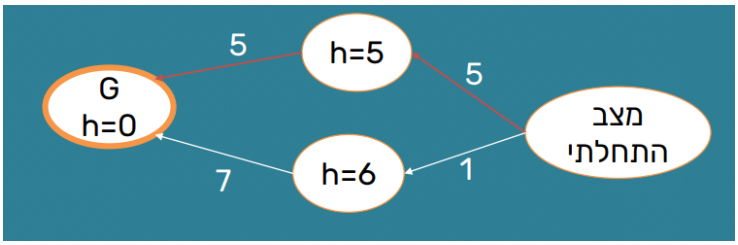
סעיף 8

ראינו בתרגול כי עקבית גורר קבילה והראנו שהיוריסטיקה אינה קבילה ולכן אינה עקבית. כלומר אותה דוגמה נגדית.

שאלה 8 - Greedy Best First Search

סעיף 1

האלגוריתם הינו שלם, לפי הנלמד בתרגול עבור מרחב סופי וקשיר כל המצבים יפותחו והיוריסטיקה אינה משפיע על האם מצב כלשהו ייפתח או לא, אלא רק קובעת את סדר הפתיחה שלהם לכן אם יש פתרון נמצא אותו.
אך האלגוריתם אינו קביל כיוון שלפי יוריסטיקה מסוימת הוא עלול לפתוח צמתים שונים ולהגיע לפתרון שאינו אופטימלי. דוגמה:



בדוגמה אנחנו רואים איך הוא מפתח את המסלול העליון בגלל היוריסטיקה ולכן לא מגיע לאופטימלי.

סעיף 2

כפי שראינו בתרגול *Greedy Best first Search* ככל הנראה יחזיר פתרון טוב יותר מהאלגוריתם *Beam Search* מצד שני האלגוריתם *Beam Search* חוסך בזמן ריצה וזיכרון.

שאלה 9 - W-A*

סעיף 2

נתון $w_1 < w_2 \leqslant 1$ נסמן את המסלולים המוחזרים ע"י $W-A^*$ תחת הפורמליזציה $f = g + w \cdot h$ ב- p_1, p_2 עבור w_1, w_2 בהתאמה. אזי $cost(p_1) < cost(p_2)$.

a. הטענה אינה נכונה, ראינו בתרגול ש- $W-A^*$ הוא האלגוריתם UCS עבור היוריסטיקה $h = 0$.

כמו כן ראינו כי h קבילה, לכן מתקיים:

$$f = g + w \cdot h = f = g + w \cdot 0 = g$$

ראינו כי UCS שלם וקביל ומתקיים כי $cost(p_1) = cost(p_2)$.

b. נבחר את אותה היורסטיקה מסעיף a, והדוגמה הנגדית היא אותה דוגמה נגדית.

שאלה 10 – IDA*

סעיף 1

IDA^* מבצע חיפוש בעץ ועל כן מפתח מצבים חוזרים מבלי לדעת שביקר בהם כבר, כלומר בזבזני בזמן ריצה וחוסך בזיכרון כי אינו שומר את המצבים שכבר היה בהם. בניגוד ל-A^* שמבצע חיפוש בגרף ועל כן שומר את המצבים בהם ביקר כלומר בזבזני בזיכרון אבל זמן הריצה שלו מהיר יותר.

לפי אותם יתרונות וחסרונות נוכל לסווג מקרים בהם היינו רוצים להשתמש בכל אחד בהתאם לדרישות זיכרון וזמן ריצה. במידה ונרצה לחסוך בזיכרון נבחר ב-IDA^* ובמידה ונרצה לחסוך בזמן ריצה נבחר ב-A^* נזכור ששניהם קבילים ושלמים לכן העדפה שלנו תיבחר על סמך שיקולי יעילות זיכרון וזמן ריצה ולא קבלת פתרון אופטימלי כי שניהם יספקו לנו בכל מקרה פתרון אופטימלי).

סעיף 2

מטרת האלגוריתם *IDA היא למצוא מסלול קצר ביותר בין צומת התחלה לצומת יעד בצורה של העמקה הדרגתית על פי ערך f כפי שנתאר. היתרון של האלגוריתם הוא בכך שהוא משלב את היתרונות *DFS* ושל *A*, ולכן הוא גם יעיל יותר בזיכרון וכמובן גם אופטימלי.

האלגוריתם פועל בצורה הבאה:

באתחול של האלגוריתם, נגדיר את ה-*threshold* להיות ערך היוריסטיקה של צומת המקור. לאחר מכן, נבצע העמקה איטרטיבית על ידי כך שהוא מבצע *DFS* עם מגבלת עומק שגדלה עם הזמן עד שהוא מוצא פתרון או עד שהוא חורג מה-*threshold* שהגדרנו עבורו.

החיפוש של *DFS* שאנחנו מבצעים הוא לא בצורה הרגילה שראינו בתחילת הקורס, אלא עם ערך *threshold* שהוא עוצר בו כאשר אנחנו חורגים ממנו. כך בעצם אנחנו מבצעים חיפוש לעומק של הגרף שאינו דורש הרבה זיכרון מכיוון שזה *DFS*, אך אנחנו גם מגבילים את עומק החיפוש של *DFS* בכך שיש לו *threshold* שהוא אינו יכול לעבור.

במהלך החיפוש של ה-*DFS*, אם לא נמצא פתרון בתוך עם ערך ה-*threshold* הנוכחי, אז הוא מתעדכן לערך *f* של המצב המינימלי שחרג מה-*threshold*. כעת, נבצע את השלבים האחרונים שוב ושוב, עד שנמצא פתרון או שנוכל להכריע כי לא קיים פתרון במקרה זה.

נשים לב כי העדכון של ה-*threshold* בסוף כל איטרציה להיות הערך המינימלי של *f* של הצמתים שחרגו מה-*threshold* מבטיח שבאיטרציה הבאה של החיפוש, בהכרח ניצור צמתים חדשים שיש להם ערך של *f* שהיה גדול מדי יחסית לאיטרציה הקודמת, אבל כעת כן נסתכל עליהם.

יתר על כן, העובדה שאנחנו מגדילים את ערך ה-*threshold* במינימום האפשרי בכל איטרציה גורמת לכך שנמצא פתרון אופטימלי מכיוון שלא נגיע אל פתרון שהוא חוקי אלא אם כן הוא הפתרון עם הערך הנמוך ביותר – כלומר אופטימלי. אך בו זמנית גם נעבור על מרחב החיפוש בצורה חכמה כדי לא לפתח יותר מדי צמתים וכדי לחסוך בעלויות זמן הריצה והזיכרון.

באופן זה, נוכל להריץ את האלגוריתם על הדוגמה המופיעה בשאלה ולקבל את הפתרון הדרוש.

ניתן למצוא את הפתרון של שאלה זו על ידי הרצה ידנית של האלגוריתם או לחלופין על ידי הרצה של האלגוריתם במחשב. התוצאה שיוצאת היא מאוד ארוכה ותיאור מפורט של האלגוריתם כדרוש בשאלה יהיה מאוד ארוך. הסברנו קודם לכן בפירוט כיצד האלגוריתם אמור להתנהג וניתן ליישם זאת על הדוגמה

הקונקרטית הנתונה בשאלה, ולמצוא את הפתרון אותו האלגוריתם יחזיר וכן את כמות הצמתים אותם הוא יפתח ואת סדר הפיתוח.

שאלה 11 – A* epsilon

סעיף 2

יתרון של A^*_ε לעומת A^* הוא מציאת פתרון מהיר יותר, אבל חיסרון שלו שהוא אינו בהכרח אופטימלי שזה מובטח לנו ב- A^*. בסך הכל, האלגוריתם נותן לנו אפשרות לקבוע עד כמה אנחנו מוכנים להתפשר על איכות הפתרון כדי לקבל זמן ריצה מהיר יותר שזו יכולת מאוד חזקה. בנוסף ישנו יתרון נוסף של A^*_ε והוא גמישות - כלומר הוא מאפשר לקבוע קריטריון משני כדי לבחור בצומת הבאה לפתרון מתוך $Focal$ וכך ניתן לשנות את אופן התנהגות האלגוריתם בהתאם להבנה שלנו את עולם הבעיה.

סעיף 3

נגדיר את היוריסטיקה הבאה: נבחר את היוריסטיקה h^{new}_{MSAP} שתוגדר באופן דומה להגדרת h_{MSAP} בשאלה 7, עם השינוי הבא: כל עוד לא אספנו את שני הכדורים, אז לא נתחשב במצבי המטרה בחישוב המינימום, מכיוון שאין לנו סיבה להגיע אל מצב המטרה במקרה זה, כי לא נוכל לנצח. כאשר נאסוף כדור כלשהו, לא נתחשב בו בחישוב ערך היוריסטיקה עבור מצב מסוים, וכאשר נאסוף את כל הכדורים, אז ערך היוריסטיקה יהיה מרחק מנהטן ממצב המטרה.

מטרת היוריסטיקה הזו היא לומר לנו איפה המרחק הקרוב ביותר אל אחד מהצמתים שמעניינים אותנו – צומת היעד או צמתים בהם יש כדור שעדיין לא אספנו, כתלות במצב שאנו נמצאים בו. החיסרון ביוריסטיקה זו הוא שהיא אינה מסתכלת על אילו צעדים אנחנו צריכים לעשות במסלול – מבחינת האם יש חור בדרך שלא ניתן לעבור בו, או מבחינת משקלי הקשתות בהם אנחנו עוברים בדרך. אבל, יוריסטיקה זו מביאה לנו באופן כללי, הסתכלות טובה על הבעיה בכך שאנחנו יודעים שההתקדמות שלנו היא לכיוון שיכול לעזור לנו בכך שאנחנו (בתקווה) מתקרבים אל מצב שהוא מעניין אותנו (כדור או מצב מטרה).

נציג את התוצאות של שימוש ביוריסטיקה זו על הבעיה הנתונה:

```
AStarEpsilon_agent = AStarEpsilonAgent()
actions, total_cost, expanded = AStarEpsilon_agent.search(env, epsilon=100)
print(f"Total_cost: {total_cost}")
print(f"Expanded: {expanded}")
print(f"Actions: {actions}")
Executed at 2024.03.05 00:53:23 in 61ms

Total_cost: 103.0
Expanded: 53
Actions: [0, 0, 0, 0, 0, 0, 1, 1, 1, 2, 1, 1, 0, 1, 1, 0]
```

שימוש ביוריסטיקה המקורית h_{MSAP} מביא לנו:

```
AStarEpsilon_agent = AStarEpsilonAgent()
actions, total_cost, expanded = AStarEpsilon_agent.search(env, epsilon=100)
print(f"Total_cost: {total_cost}")
print(f"Expanded: {expanded}")
print(f"Actions: {actions}")
Executed at 2024.03.05 00:52:33 in 72ms

Total_cost: 103.0
Expanded: 88
Actions: [0, 0, 0, 0, 0, 0, 1, 1, 1, 2, 1, 1, 0, 1, 1, 0]
```

ולעומת זאת שימוש ב-$g(s)$ שבה נאמר לנו בשאלה להשתמש מביא לנו:

```
AStarEpsilon_agent = AStarEpsilonAgent()
actions, total_cost, expanded = AStarEpsilon_agent.search(env, epsilon=100)
print(f"Total_cost: {total_cost}")
print(f"Expanded: {expanded}")
print(f"Actions: {actions}")
Executed at 2024.03.05 00:50:37 in 62ms

Total_cost: 103.0
Expanded: 81
Actions: [0, 0, 0, 0, 0, 0, 1, 1, 1, 2, 1, 1, 0, 1, 1, 0]
```

כלומר, אנחנו רואים שהיוריסטיקה החדש שהצענו משפרת את h_{MSAP} מבחינת הביצועים עבור בעיה ספציפית זו מבחינת כמות הצמתים אותה אנו מפתחים, וזאת מכיוון שיוריסטיקה זו מיודעת יותר.

סעיף 4

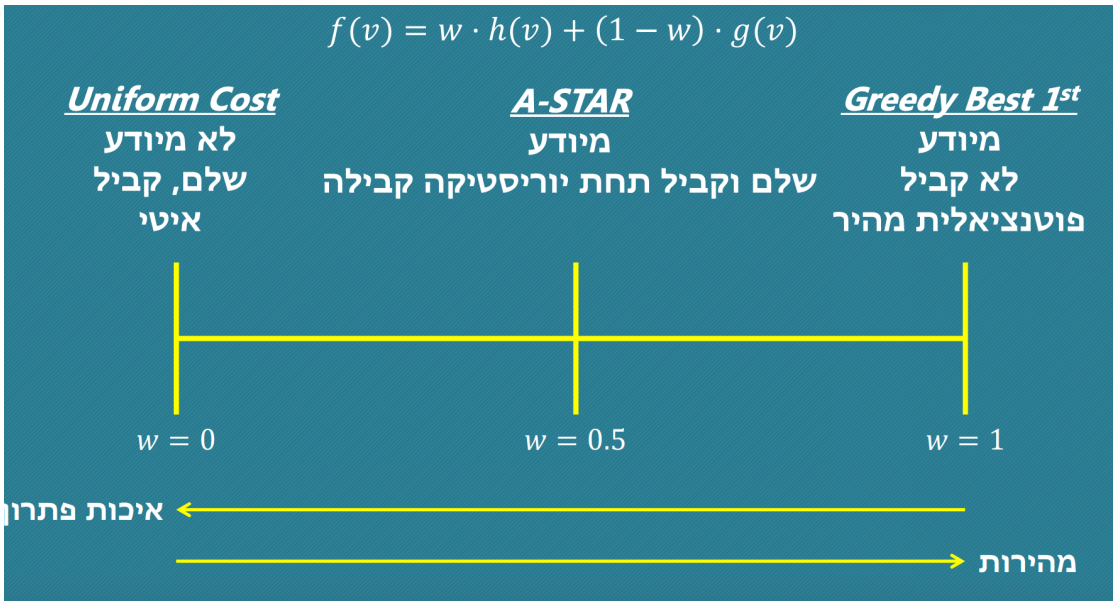
במקרה בו נגדיר כי $\varepsilon = \infty$, אז כאשר נגדיר את הקבוצה $Focal$ נכניס את כל הצמתים שהם ב-$Succ$ של המצב הנוכחי בו אנחנו נמצאים. לאחר מכן, כאשר נוציא מתור העדיפויות את הצומת הבא לפיתוח, נוציא את הצומת שערך היוריסטיקה עליו הוא הנמוך ביותר. במקרה שלנו, בחרנו את היוריסטיקה להיות $g(s)$, כלומר אנחנו נפתח בהכרח את הצומת עם המרחק הקטן ביותר מצומת המקור. כלומר סך הכל, האלגוריתם שנקבל יהיה שקול לחלוטין ל-*UCS*.

שאלה 12 – Local Search

מצורף צילום של התוצאות שקיבלנו מהרצת האלגוריתמים השונים.

	1	2	3	4
C1	map	map12x12	map15x15	map20x20
C2	BFS-G cost	140.0	215.0	203.0
C3	BFS-G expanded	445	858	1045
C4	WA* (0.5) cost	118.0	178.0	188.0
C5	WA* (0.5) expanded	224	651	684
C6	WA* (0.7) cost	118.0	178.0	188.0
C7	WA* (0.7) expanded	200	604	587
C8	WA* (0.9) cost	118.0	195.0	188.0
C9	WA* (0.9) expanded	240	707	1002

על פי מה שלמדנו בהרצאה:



כלומר, אנחנו מצפים שכאשר נגדיל את ערכו של הפרמטר w, איכות הפתרון תקטן אך נקבל זמן ריצה מהיר יותר של האלגוריתם. יתר על כן, אנחנו מצפים שעבור $w = \frac{1}{2}$, שמצב זה מתלכד בדיוק עם ההגדרה של A^*, נקבל פתרון אופטימלי. כמו כן הפתרון יהיה מהיר יותר מאשר BFS-G שאינו בהכרח אופטימלי והוא גם יהיה יחסית איטי יותר מאשר שאר האלגוריתמים שבחנו מכיוון שהוא אינו מיודע ואינו משתמש ביוריסטיקה.

אכן, קיבלנו שעבור $w = \frac{1}{2}$ ועבור $w = 0.7$ הפתרון שהתקבל הינו אופטימלי, ואילו עבור $w = 0.9$ הפתרון שהתקבל אינו אופטימלי, אך כפי שניתן לראות כאשר מגדילים את ערך הפרמטר w אל עבר הערך 1, האלגוריתם נהיה מהיר יותר.

כמו כן, התוצאות של BFS-G אינן טובות מכיוון שהוא אינו מיודע וכמו כן משקלי הקשתות לא שווים, ולכן הוא אינו מחזיר פתרון אופטימלי במקרה זה, וכמו כן הוא אינו מנצל את הידע שיש לנו על עולם הבעיה כמו שאר האלגוריתמים, ולכן הוא גם איטי יותר.

כלומר, התוצאות שהתקבלו תואמות את הציפיות שלנו.

שאלה 13 – Local Search

סעיף 1

כפי שראינו בהרצאה, מתקיים:

$$\Pr(b \mid a) = \frac{2}{5}$$

$$\Pr(c \mid a) = \frac{2}{5}$$

$$\Pr(d \mid a) = \frac{1}{5}$$

סעיף 2

אם $4 < \beta$, אז מספר הצעדים המקסימלי הוא 3 וזה קורה עבור המסלול: $A \to B \to F \to G$. המעבר האחרון יהיה חוקי במקרה זה כי אנחנו נשפר את ערך הפונקציה במעבר הזה.

סעיף 3

לא, הסיבה לכך היא שהמקסימום הגלובלי הוא בהכרח F או G כתלות בערך של β, ולכן אם נלך בהתחלה למצב C, נלך לאחר מכן בהכרח אל H, מכיוון שהוא המצב היחיד שמשפר מ-C, וכך ניתקע במקסימום לוקלי.

סעיף 4

כפי שהסברנו בסעיף הקודם, אם נלך תחילה אל C, בהכרח נתכנס לפתרון שהוא לא אופטימלי. באופן דומה, אם נלך בהתחלה אל D, גם אז לא נגיע אל פתרון אופטימלי. כלומר, בהסתברות של $\frac{1}{5} + \frac{2}{5} = \frac{3}{5}$ לא נתכנס לפתרון אופטימלי.

עד כאן, הגענו אל מצב B בהסתברות של $\frac{2}{5}$. כעת נחלק למקרים לפי ערכו של β:

1. אם $3 < \beta \leq 4$, אז נרצה להגיע אל F כי הוא המצב האופטימלי. זה יקרה בהסתברות $\frac{\beta-2}{\beta-2+2} = \frac{\beta-2}{\beta}$. לכן במקרה זה, ההסתברות להגיע אל מקסימום שאינו גלובלי היא:

$$\frac{3}{5} + \frac{2}{5} \cdot \frac{\beta - 2}{\beta}$$

2. אם $4 < \beta$, אז אם האלגוריתם ילך ישירות מ-B אל G נגיע למקסימום גלובלי, אבל גם הוא ילך קודם לכן אל F, הוא ימשיך באיטרציה הבאה אל G שהוא המקסימום הגלובלי, מכיוון שהערך שלו גבוה יותר מאשר של F. לכן, במקרה זה, אנחנו נתכנס למקסימום גלובלי אמ"מ נלך אל B בהתחלה, כלומר ההסתברות לכך היא: $\frac{3}{5}$.

סעיף 5

אין ערכים של β כאלו. נוכיח זאת: כפי שראינו בסעיפים הקודמים, המסלול הארוך ביותר בגרף הוא באורך של בדיוק 3 צעדים, והוא מסתיים כאשר מגיעים אל G ואין מסלול באורך זה שאינו מגיע אל G לבסוף. כמו כן צריך לדרוש במקרה זה שיתקיים: $\beta > 4$ על מנת שאכן G יהיה המקסימום הגלובלי.

כפי שראינו על מנת שזה יקרה צריך תחילה ללכת אל B ולאחר מכן ללכת אל F ואז לבסוף ללכת אל G. המעבר אל B קורה בהסתברות $\frac{2}{5}$, לאחר מכן המעבר אל F קורה בהסתברות $\frac{2}{\beta}$, ולבסוף המעבר אל G קורה בוודאות. כלומר, ההסתברות לכך שאורך המסלול יהיה בדיוק 3 צעדים היא: $\frac{2}{5} \cdot \frac{2}{\beta}$. נרצה לדרוש שהסתברות זאת תהיה גדולה מ-$\frac{1}{5}$, לכן נקבל:

$$\frac{2}{5} \cdot \frac{2}{\beta} > \frac{1}{5}$$

$$\frac{4}{\beta} > 1$$

$$\beta < 4$$

וזו סתירה לכך שהנחנו שבמקרה זה צריך להתקיים $\beta > 4$ על מנת שהמקסימום הגלובלי אכן יהיה ב-G.