בינה מלאכותית | מטלה 1 | 206118754

1. <u>היוריסטיקה</u>

בחרתי ליישם 2 סוגים שונים של היוריסטיקות, אחת עבור כל סוג סוכן. עבור סוכן מסוג "מלך" אני בחרתי ליישם את ההיוריסטיקה באמצעות גרסה שונה של אלגוריתם "**מרחק מנהטן**" (**Manhattan Distance**) המוכר. עבור כל מצב לוח (מצב עולם) ועבור כל סוכן, אני מחשב את "מרחק מנהטן" בין הסוכן הנוכחי במצב הנוכחי לעבר כל סוכן בלוח המטרה, את כל ערכים אלו אני שומר ברשימה ולאחר מכן אני מחלץ מלוח המטרה את הערך המתקבל עבור הזזת הסוכן הספציפי (הסוכן הנוכחי במצב הנוכחי), כלומר הסוכן הקרוב ביותר לסוכן הנוכחי במצב הלוח הנוכחי. וכך חוזר על התהליך הזה עד שנסיים לעבור על כל האופציות אל מול הסוכנים הנמצאים בלוח המטרה.

מפני שסוכן מסוג "בישוף" הולך בצורה שונה מסוכן ה"מלך" ויכול להתקדם יותר ממשבצת אחת בצעד אחד, אני בחרתי ליישם את ההיוריסטיקה באמצעות שילוב של אלגוריתם חישוב "**מרחק המינג**" (**Hamming Distance**) ביחד עם אלגוריתם חישוב "**המרחק האוקלידי**" (**Euclidean Distance**). מרחק המניג עובד בדך כלל על שתי מחזרות בעלות אורך זהה, וערכו המספרי הינו מספר השינויים המזערי שיש לעשות באותה מחרוזת (מחרוזת התחלה) כדי לקבל את המחרוזת השנייה (מחרוזת המטרה). בהיוריסטיקה זו ניקח את המצב הנוכחי\התחלתי כמחרוזת ההתחלה, ואת מצב המטרה כמחרוזת שנייה להשוואה (מחרוזת המטרה). מספר השינויים יהיה מספר הבישופים שאינם נמצאים במקום שלהם ביחס למצב המטרה, לכן עבור כל בישוף במצב הנוכחי אנחנו נבדוק האם הוא נמצא ברשימת הבישופים של מצב המטרה, ועבור כל אי התאמה נוסיף 1 לערך ההיוריסטיקה.

בנוסף לערך ההיוריסטיקה המתקבל ע"י מרחק המניג של אותו מצב, נחשב בנוסף את המרחק האוקלידי שיכול לתת אינדיקציה להתקרבות סוכן הבישוף אל המטרה וייתן יתרון נוסף ומשמעותי לאותו מצב שנרצה "לקחת" כצעד הבא שלנו אל הפתרון. הוספה של מרחק זה תשפר לנו את היכולת לחישוב היוריסטיקה מתאימה לפתרון טוב יותר, שכן מרחק המינג לבדו, בהינתן רשימת בישופים של בישוף אחד יכולה לתת ערך זהה בכל מצב למרות התקדמות הבישוף למטרה. לאחר חישוב ההיוריסטיקה (h = Hamming Distance + Euclidean Distance) עבור הבישוף, בצורה דומה אל סוכן המלך, נשמור את על ערכים אלו ברשימה ונחלץ את המהלך עם הערך המינימלי לקבלת החלטה למהלך הבא.

בנוסף, עבור שני היוריסטיקות, אם מספר הסוכנים בלוח הנוכחי גדול ממספר הסוכנים שיש ללוח המטרה, אני ארצה שהיוריסטיקה תעדיף מצבי לוח שבהן הסוכנים קרובים יותר לשורה השישית. דבר זה יעודד את האלגוריתם לבחור נתיבים שיובילו לעודף סוכנים לעזוב את הלוח (על ידי מעבר לשורה השביעית). ולכן כאשר יש עודף של סוכנים בלוח הנוכחי, אני אחשב את המרחק של כל סוכן לשורה האחרונה, אסכם את המרחקים האלו ואכפיל אותם במקדם משקל להיוריסטיקה, כאשר המשקל שנבחר הינו 2.

אם מספר הסוכנים בלוח הנוכחי קטן ממספר הסוכנים בלוח היעד (או אם נותרו 0 סוכנים בלוח הנוכחי)
ההיוריסטיקה תחזיר אינסוף למצב זה, בגלל שאלו מצבים לא רצויים - מצבים שניסינו לחקור אך לא יביאו פתרון.

ההיוריסטיקה קבילה אם אין סוכנים עודפים בלוח ההתחלתי בהשוואה ללוח המטרה. הסיבה לכך היא שמרחק מנהטן ברשת מלבנית שווה למרחק האמיתי. יתרה מכך, אם יש שדות כוח בדרך, מרחק מנהטן נותן ערך קטן יותר מהערך האמיתי (אם הסוכן צריך לעשות "עקיפה" כדי להתחמק מהחומה) אבל, ההיוריסטיקה אינה קבילה אם יש עודף סוכנים.

הסיבה לכך היא שאנו מוסיפים לחישוב היוריסטיקות את המרחק של כל סוכן מהשורה האחרונה כפול מקדם משקל במקרים אלו, היא שההיוריסטיקה נותנת מספר גבוה יותר מהמרחק האמיתי מהמטרה. ההיוריסטיקה אינה עקבית, ולכן הקשר בין הערכות היוריסטיות לכל סוכן אינו ברור ולכן בחרתי לחשב את היוריסטיקה לכל הלוח ולא לכל סוכן בנפרד.

2. IDS vs DFS

DFS – אלגוריתם חיפוש זה חיפוש לעומק בודק את הצומת עליו הוא הופעל, ולאחר מכן מפעיל את עצמו רקורסיבית על כל אחד מהצמתים שמקושרים לצומת אליו הופעל, אם הוא טרם ביקר בהם וכך בעצם מצליח להגיע לעלה התחתון ביותר (אל התא העמוק ביותר).

IDS – אלגוריתם זה קורא לDFS עבור עומקים שונים, כלומר עבור כל רמה בגרף.
עבור כל קריאה, הDFS מוגבל מלעבור לרמה עמוקה יותר ולכן אלגוריתם זה בעצם מבצע DFS בצורה של BFS.

נתייחס למקרה בו נתון לנו גרף "שרשרת" בגדול n שכל קודקוד מחובר אל הקודקוד הבא ע"י קשת :

[0] - > [1] -> [2] -> ... ->[n-1] ->[n]

סיבוכיות הזמן במקרה זה :

- בעבור מצב כזה אלגוריתם DFS הינו יעיל הרבה יותר מIDS , וזמן הריצה שלו יהיה $O(n)$ מכוון שיתחיל בקודקוד הראשון ואז יעבור להבא אחריו וכך האלה עד לקודקוד האחרון והעמוק ביותר (קודקוד n).
- עבור אלגוריתם IDS , יקיים n איטרציות סה"כ ובעבור כל איטרציה יצבע את אלגוריתם DFS בעומק הקטן-שווה למספר האיטרציות ולכן במקרה הגרוע ביותר אנחנו נקבל סיבוכיות זמן של $O(n^2)$ עד העלה האחרון בגרף השרשרת.

סיבוכיות המקום במקרה זה :

- באלגוריתם DFS סיבוכיות המקום יהיה שוו לעומק המקסימלי של הגרף, כלומר עבור המקרה שלנו הוא יהיה $O(n)$ עבור גרף שרשרת באורך n.
- באלגוריתם IDS סיבוכיות המקום תהיה דומה לאלגוריתם ה DFS - $O(d)$ כאשר d יהיה העומק המטרה המקסימלי בגרף, כלומר עבור המקרה שלנו בגרף שרשרת סיבוכיות המקום יהיה $O(n)$.
- הבדל בין אלגוריתמים אלו יכול להיות כאשר מדובר בגרפים גדולים (כמעט אינסופיים) או בעלי מעגלים שבהם ה DFS עלול שלא לעצור גם אם האיבר שהוא מחפש נמצא בגרף כי המסלול שבו הוא יבחר עלול להיות מסלול אינסופי שבו האיבר שמחפשים לא נמצא, ואז האלגוריתם יתקדם ללא הפסקה באותו מסלול מבלי לחזור על עקבותיו, פתרון לבעיה זו יכולה להיפתר ע"י אלגוריתם הIDS שבו החיפוש לעומק מתבצע כל פעם עד עומק קבוע מראש, כאשר אם לא נמצאה התשובה מגדילים את עומק החיפוש.