תרגיל 2

הערות:

- באלגוריתם simulated annealing השתמשתי ברשימת visited אליה הכנסתי את כל הnodes שהגענו אליהם (גם כאלו שבסוף לא נבחרו על סמך ההסתברות) על אף שלפי הפסאודו-קוד של האלגוריתם זה לא נדרש. עשיתי זאת מכיוון שבעבודה הוגדר לנו מקסימום t של 100 צעדים ולכן העדפתי שלא "לבזבז" איטרציה על node שאפילו לא נבחר בהסתברות יותר גדולה באיטרציה קודמת והסיכוי שהוא ייבחר באיטרציה מאוחרת אפילו קטנה יותר.

- למדנו בכיתה שאלגוריתם hill climbing הוא למעשה אלגוריתם beam search עם k=1 ולכן המימוש שלו נעשה ע"י קריאת לפונקצייתbeam search .

- בbeam search הגדרתי max sidewat move כדי להמנע ממצב שבו אנחנו נתקעים ב"כתף" (ערך פונקציית העלות נשארת זהה ולא משתנה). במידה ויהיו מעל 50 צעדים שבהם אין התקדמות יתבצע ריסטרט.

שאלה 1

פונקצית הטמפרטורה היא $T(t) = 1 - \frac{t}{t_{\max}}$ כאשר הגדרתי $t_{max} = 100 + 1$. בתרגיל דרשו שהאלגוריתם ימשיך עד לאיטרציה $t = 100$ ולכן ברגע שהאלגוריתם יגיע לאיטרציה ה-101 ערך T יהיה שווה ל-0 והוא ייעצר.

בנוסף הגדרתי כי ההסתברות לצעד הבא היא
$$P = \begin{cases} 1 & , \ delta > 0 \\ e^{\frac{delta-1}{T}} & , \ delta \leq 0 \end{cases}$$

כאשר $delta$ היא ההפרש בין ערך ההיוריסטיקה הנוכחית לבין ערך ההיוריסטיקה של הערך הבא ולכן כאשר $delta$ חיובית המשמעות היא שהצעד הבא על פי ההיוריסטיקה יותר קרוב ליעד ולכן במקרה זה נרצה לבצע את הצעד בוודאות ולכן $P = 1$.

כאשר $delta$ לא חיובית הצעד לא בטוח משתלם ולכן נרצה לבצעו בהסתברות כלשהי, כאשר ההסתברות הולכת ופוחתת ככל ש- t גדל. כלומר ככל שנתקדם עם האיטרציות נרצה פחות ופחות לבצע צעדים "מסוכנים". ניתן לראות ב- $e^{\frac{delta-1}{T}}$ שהמונה שלילי בהכרח והמכנה חיובי ולכן האקספוננט שלילי, כמו כן T היא פונקציה מונוטונית יורדת של t ולכן ככל ש- t גדל אזי האקספוננט קטן אפילו יותר ולכן ההסתברות הולכת וקטנה עד שהיא שואפת לאפס.

שאלה 2

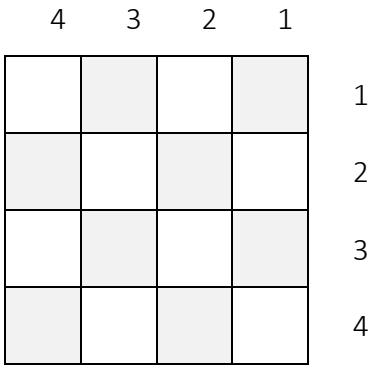
בעיית 4 מלכות כבעיית CSP:

אור נחשון אגש
311330500

בשביל להקל ולשפט את כתיבת האילוצים בחרתי להסתכל על הבעיה קצת הפוך, שהמשתנים הם דווקא משבצות המשחק ולא המלכות.

- $\mathcal{X}$- קבוצת המשתנים. יש 16 משבצות. נסמן כל משבצת כמשתנה $\mathcal{X}_{ij}$ כאשר:

  $1 \le j \le 4\ ,\ 1 \le i \le 4$ (i- שורה, j- עמודה)



ולכן $\mathcal{X} = \{X_{11}, X_{12}, X_{13}, X_{14}, X_{21}, X_{22}, X_{23}, X_{24}, X_{31}, X_{32}, X_{33}, X_{34}, X_{41}, X_{42}, X_{43}, X_{44}\}$

- D- קבוצת הדומיינים. כל דומיין D_{ij} מכיל סט ערכים אפשריים להשמה עבור משתנה X_{ij} בהתאמה ולכן קבוצת האילוצים תהיה:

$$D = \{D_{11}, D_{12}, D_{13}, D_{14}, D_{21}, D_{22}, D_{23}, D_{24}, D_{31}, D_{32}, D_{33}, D_{34}, D_{41}, D_{42}, D_{43}, D_{44}\}$$

כל משבצת יכולה להיות מאויישת (1) או לא (0), כלומר: $\forall i,j,\ \ \mathcal{D}_{ij} = \{0,1\}$.

- C - קבוצת האילוצים.

ראשית יש הגבלה להשמה של ארבע מלכות ולכן נגדיר אילוץ: $\sum_{\substack{1\le i\le 4\\ 1\le j\le 4}} X_{ij} = 4$

בנוסף, אם יש השמה כלשהי של מלכה במשבצת, אסור לבצע השמה של מלכה נוספת באותה שורה:

$$X_{ij} = 1 \rightarrow \qquad X_{ik} = 0\ \ for\ 1 \le k \le 4, k \ne j$$

וגם אסור לבצע השמה באותה עמודה:

$$X_{ij} = 1 \rightarrow \qquad X_{kj} = 0\ \ for\ 1 \le k \le 4, k \ne i$$

בנוסף אסור שתהיה השמה באותו אלכסון ולכן:

$$X_{ij} = 1 \rightarrow \quad X_{km} = 0\ \ for\ \ |i-j| = |k-m|, \qquad 1 \le k\ , m \le 4\ , k \ne i\ , m \ne j$$

$$X_{ij} = 1 \rightarrow \quad X_{km} = 0\ \ for\ \ |i+j| = |k+m|, \qquad 1 \le k\ , m \le 4\ , k \ne i\ , m \ne j$$

לדוגמא עבור השמה $X_{3,2}$ נרצה את האילוצים הבאים:

4 3 2 1

1		0		0
2	0	0	0	
3	0	1	0	0
4	0	0	0	

איסור שורה:

$X_{32} = 1 \rightarrow X_{31} = 0$

$X_{32} = 1 \rightarrow X_{33} = 0$

$X_{32} = 1 \rightarrow X_{34} = 0$

איסור עמודה:

$X_{32} = 1 \rightarrow X_{12} = 0$

$X_{32} = 1 \rightarrow X_{22} = 0$

$X_{32} = 1 \rightarrow X_{42} = 0$

איסור אלכסונים:

$X_{32} = 1 \rightarrow X_{21} = 0$

$X_{32} = 1 \rightarrow X_{23} = 0$

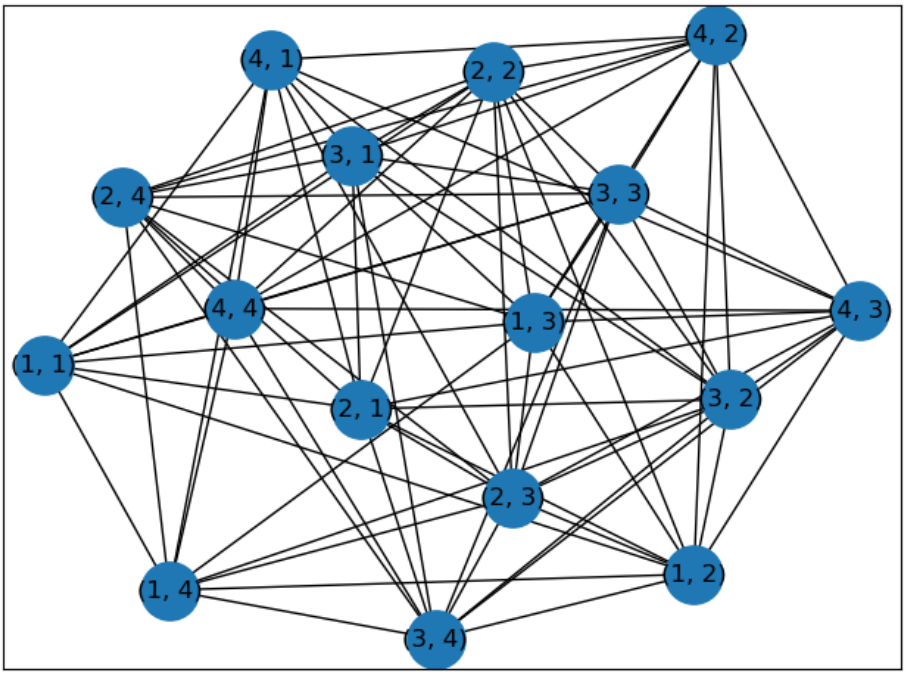
$X_{32} = 1 \rightarrow X_{43} = 0$

$X_{32} = 1 \rightarrow X_{14} = 0$

$X_{32} = 1 \rightarrow X_{41} = 0$

גרף האילוצים יראה כך:

כדי לראות את הקוד python שבעזרתו בניתי את הגרף ראה נספח.



שאלה 3

ראשית אחשב את המורכבות של הפונקציה revise: בעצם זו פונקציה שעבור שני משתנים מאולצים, עבור כל ערך בדומיין i היא עוברת על כל הערכים בדומיין j ובוחנת האם הם מתאימים לו בהתחשב באילוץ. אז למעשה מורכבות הפונקציה היא d^2 כאשר d הוא גודל הדומיין המקסימלי מבין השניים.

כמה פעמים במקרה הגרוע נקרא לפונקציה revise? בהתחלה כל האילוצים (=כל קשתות |E|) נמצאים בתור, ובכל פעם שקוראים ל-revise מסירים אילוץ אחד מהתור וכאשר התור יתרוקן יסתיימו הקריאות לפונקציה. אבל ייתכן גם שנוסיף אילוצים חזרה לתור במקרה ומחקנו ערך מהדומיין שהקשת מצביעה אליו. קשת מצביעה לדומיין אחד כל פעם, לכן אנחנו יכולים להוסיף קשתות כמספר הערכים באותו דומיין. לכן במקרה הגרוע נוסיף כל קשת חזרה לתור d פעמים (כגודל הדומיין). מכאן שהפונקציה נקראת לכל היותר $|E| * d$

סה"כ ניתן לומר שהסיבוכיות של כל האלגוריתם היא:

$$O((|E| * d) * d^2) = O(|E| * d^3)$$

נספחים

נספח 1 – הקוד לבניית גרף האילוצים של שאלה 2

```
import networkx as nx
import matplotlib.pyplot as plt

# Defining a Class
class GraphVisualization:
```

```
    def __init__(self):
        # visual is a list which stores all
        # the set of edges that constitutes a
        # graph
        self.visual = []

        # addEdge function inputs the vertices of an

    # edge and appends it to the visual list
    def addEdge(self, a, b):
        temp = [a, b]
        self.visual.append(temp)

        # In visualize function G is an object of

    # class Graph given by networkx G.add_edges_from(visual)
    # creates a graph with a given list
    # nx.draw_networkx(G) - plots the graph
    # plt.show() - displays the graph
    def visualize(self):
        G = nx.Graph()
        G.add_edges_from(self.visual)
        nx.draw_networkx(G, node_size=800)
        plt.show()

    # Driver code


G = GraphVisualization()
for i in range(1,5):
    for j in range(1, 5):
        for k in range(1,5):
            if k != j:
                G.addEdge((i,j), (i,k))
            if k !=i:
                G.addEdge((i, j), (k,j))
            for m in range(1,5):
                if abs(i-j)==abs(k-m):
                    G.addEdge((i, j), (k, m))
                if abs(i+j)==abs(k+m):
                    G.addEdge((i, j), (k, m))
G.visualize()
```