BIO COMPUTATION FINAL PROJECT

GitHub Link:

GitHub Link Bio-Computation-Final-Project

Hadas Hannasab 213486764

Shoham Galili 208010785

שאלה 1:

ישנם ארבעה גורמים המשפיעים על הגן המרכזי: שני מפעילים ושני מעכבים, אותם נציג כרביעייה: (activator 1, activator 2, inhibitor 1, inhibitor 2).

יצרנו רשימה של המצבים האפשריים:

```
#input scenarios
inputs = [
    (0, 0, 0, 0),
    (1, 0, 0, 0),
    (1, 1, 0, 0),
    (0, 0, 1, 0),
    (1, 0, 1, 0),
    (1, 1, 1, 0),
    (0, 0, 1, 1),
    (1, 0, 1, 1),
    (1, 1, 1, 1),
]
```

תרחישים מסוימים דומים, כגון המצב (0, 1, 0, 0) דומים ל- (1, 0, 0, 0) מכיוון שלשניהם יש אקטיבטור אחד מופעל ומעכב אחד כבוי. כתוצאה מכך, ישנם 9 תרחישים נפרדים בלבד ולא 16.

אנו ניצור את כל השילובים האפשריים של פונקציות באופן הבא:

```
#Generate all Boolean functions and filter for monotonicity
possible_functions = list(product([0, 1], repeat=len(inputs)))
```

לאחר מכן, אנו בודקים האם הערך בעמודה השלישית בפונקציה הוא 1 - דבר שחייב להתקיים, בנוסף חייב להתקיים שהפונקציה היא עם 0 בעמודה השישית שלה. נוודא זאת בצורה הבאה:

```
if func[2] != 1 or func[6] != 0:
    return False
```

כאשר func זו הפונקציה.

לאחר מכן נבחן כל זוג עמודות (כל אחד מייצג תרחיש, כגון 0, 0, 0, 0, התואם לעמודה הראשונה בהרצאה) כדי לבדוק אם יש מונוטוניות. לדוגמה, שלושת העמודות הראשונות של ההרצאה מראים כי:

$$(0,0,0,0) \leq (1,0,0,0) \leq (1,1,0,0) = 1$$

זאת משום שכיבוי של מפעיל אמור לגרום לכך שערך הפונקציה יישאר זהה או יקטן.

```
for i, sc1 in enumerate(inputs):
    for j, sc2 in enumerate(inputs):
        if scenarios_comparable(sc1, sc2) and func[i] > func[j]:
            return False
```

אם הערך בעמודה i קטן מאשר בעמודה j, אבל הפונקציה ב-i גדולה מהפונקציה ב-j, הפונקציה לא חוקית מכיוון שהיא לא שומרת על מונוטוניות.

אנו מעריכים את השקילות הזו על ידי השוואת זוגות של עמודות: אם שני מפעילים שווים, אנו בודקים את המעכבים, ואם המונוטוניות מתקיימת, נחזיר TRUE. באופן דומה, אם המעכבים שווים, אנו בודקים את המפעילים.

```
#This function compare scenarios based on activation and inhibition
1 usage
def scenarios_comparable(s1, s2):
    if (s1[2] == s2[2] and s1[3] == s2[3] and s1[0] <= s2[0] and s1[1] <= s2[1]) or \
            (s1[0] == s2[0] and s1[1] == s2[1] and s1[2] >= s2[2] and s1[3] >= s2[3]):
        return True
    return False
```

לבסוף, נציג את התוצאות הן בקונסולה והן בטבלה חיצונית:

```
#This function create a DataFrame to store the data
def prepare_dataframe(inputs,monotonic_funcs):
    formatted_data = []
    for func in monotonic_funcs:
        formatted_row = [colored('1', 'red') if bit == 1 else
colored('0', 'white') for bit in func]
        formatted_data.append(formatted_row)

    return pd.DataFrame(formatted_data, columns=[str(inp) for inp in
inputs])

#This function display the table using tkinter
def display_table(df):
    window = tk.Tk()
    window.title("Monotonic Functions Overview")

    tree = ttk.Treeview(window, columns=list(df.columns),
show='headings')

    for col in df.columns:
        tree.heading(col, text=col)
        tree.column(col, width=100)

    for _, row in df.iterrows():
        tree.insert("", "end", values=list(row))

    tree.pack(expand=True, fill='both')
    window.mainloop()


#This function print outputs to console
def console_output(df,monotonic_funcs):
```

```
    print(tabulate(df, headers='keys', tablefmt='pretty'))
    print(f"Total monotonic functions: {len(monotonic_funcs)}")
    print("Monotonic functions listing:")
    for func in monotonic_funcs:
        print(func)
```

התוצאה מראה כי לאחר החלת כל האילוצים, אנו מגיעים ל-18 פונקציות שונות, בהתאם למה שראינו בהרצאה:

```
C:\Users\yosef\AppData\Local\Programs\Python\Python39\python.exe "C:\Users\yosef\OneDrive\שולחן העבודה\לימודים\שנה ד\סמסטר ב\חישוב ביולוגי\פרויקט
```

	(0, 0, 0, 0)	(1, 0, 0, 0)	(1, 1, 0, 0)	(0, 0, 1, 0)	(1, 0, 1, 0)	(1, 1, 1, 0)	(0, 0, 1, 1)	(1, 0, 1, 1)	(1, 1, 1, 1)
0	0	0	1	0	0	0	0	0	0
1	0	0	1	0	0	1	0	0	0
2	0	0	1	0	0	1	0	0	1
3	0	1	1	0	0	0	0	0	0
4	0	1	1	0	0	1	0	0	0
5	0	1	1	0	0	1	0	0	1
6	0	1	1	0	1	1	0	0	0
7	0	1	1	0	1	1	0	0	1
8	0	1	1	0	1	1	0	1	1
9	1	1	1	0	0	0	0	0	0
10	1	1	1	0	0	1	0	0	0
11	1	1	1	0	0	1	0	0	1
12	1	1	1	0	1	1	0	0	0
13	1	1	1	0	1	1	0	0	1
14	1	1	1	0	1	1	0	1	1
15	1	1	1	1	1	1	0	0	0
16	1	1	1	1	1	1	0	0	1
17	1	1	1	1	1	1	0	1	1

```
Total monotonic functions: 18
Monotonic functions listing:
(0, 0, 1, 0, 0, 0, 0, 0, 0)
(0, 0, 1, 0, 0, 1, 0, 0, 0)
(0, 0, 1, 0, 0, 1, 0, 0, 1)
(0, 1, 1, 0, 0, 0, 0, 0, 0)
(0, 1, 1, 0, 0, 1, 0, 0, 0)
(0, 1, 1, 0, 0, 1, 0, 0, 1)
(0, 1, 1, 0, 1, 1, 0, 0, 0)
(0, 1, 1, 0, 1, 1, 0, 0, 1)
(0, 1, 1, 0, 1, 1, 0, 1, 1)
(1, 1, 1, 0, 0, 0, 0, 0, 0)
(1, 1, 1, 0, 0, 1, 0, 0, 0)
(1, 1, 1, 0, 0, 1, 0, 0, 1)
(1, 1, 1, 0, 1, 1, 0, 0, 0)
(1, 1, 1, 0, 1, 1, 0, 0, 1)
(1, 1, 1, 0, 1, 1, 0, 1, 1)
(1, 1, 1, 1, 1, 1, 0, 0, 0)
(1, 1, 1, 1, 1, 1, 0, 0, 1)
(1, 1, 1, 1, 1, 1, 0, 1, 1)
```

Monotonic Functions Overview

(0, 0, 0, 0)	(1, 0, 0, 0)	(1, 1, 0, 0)	(0, 0, 1, 0)	(1, 0, 1, 0)	(1, 1, 1, 0)	(0, 0, 1, 1)	(1, 0, 1, 1)	(1, 1, 1, 1)
0	0	1	0	0	0	0	0	0
0	0	1	0	0	1	0	0	0
0	0	1	0	0	1	0	0	1
0	1	1	0	0	0	0	0	0
0	1	1	0	0	1	0	0	0
0	1	1	0	0	1	0	0	1
0	1	1	0	1	1	0	0	0
0	1	1	0	1	1	0	0	1
0	1	1	0	1	1	0	1	1
1	1	1	0	0	0	0	0	0
1	1	1	0	0	1	0	0	0
1	1	1	0	0	1	0	0	1
1	1	1	0	1	1	0	0	0
1	1	1	0	1	1	0	0	1
1	1	1	0	1	1	0	1	1
1	1	1	1	1	1	0	0	0
1	1	1	1	1	1	0	0	1
1	1	1	1	1	1	0	1	1

שאלה 2:

סיכום המאמר:

המאמר מציג מאגר מקיף של מודלים של רשתות בוליאניות שנאספו ממקורות שונים בעולם האמיתי, תוך התמקדות בהבטחת העקביות הלוגית של מודלים אלה. רשתות בוליאניות חיוניות בביולוגיה של מערכות למידול רשתות וויסות גנים ומערכות ביולוגיות אחרות. עם זאת, דגמים רבים סובלים מחוסר עקביות לוגית, שעלול לערער את האמינות והישימות שלהם. מערך הנתונים כולל למעלה מ-230 מודלים שמקורם במאגרי מידע וספרות שונים. התכונות העיקריות של מאגר זה כוללות:

- מגוון רחב של מודלים: מערך הנתונים כולל מודלים במספר פורמטים כגון bnet, aeon ו-sbml כדי להקל על שימושיות רחבה. כמו כן, המאגר מכיל למעלה מ-230 דגמי רשת בוליאנית. המודלים נאספים ממקורות מגוונים, כולל ספרות שפורסמה ומאגרי מידע ביולוגיים.
- Validation Pipeline: Validation Pipeline נועד לבדוק את העקביות הלוגית של כל מודל. ה - pipeline זיהה ותיקן למעלה מ-400 בעיות פוטנציאליות במספר דגמים. ה - pipeline מבטיח שכל מודל יעמוד בהתנהגות הלוגית הצפויה, מה שהופך אותם לאמינים להמשך מחקר ויישומים.
- נגישות, כלים ושימוש חוזר: המאגר מספק כלים להמרת מודלים לפורמטים שונים ויצירת מהדורות מותאמות אישית המותאמות לצרכים ספציפיים. זה חיוני לשיפור השימוש החוזר וההשוואה של מודלים בקהילת המחקר. המאגר והכלים שלו שואפים לסטנדרטיזציה של השימוש ברשתות בוליאניות בביולוגיה חישובית, לשפר את המהימנות והשחזור של מחקרים בתחום זה.

תובנות ויישומים ביולוגיים

המודלים במאגר מכסים מגוון רחב של תהליכים ביולוגיים, כולל ויסות גנים, העברת אותות ובקרת מחזור התא. חוקרים יכולים להשתמש במודלים אלה כדי לדמות מערכות ביולוגיות,

לזהות רכיבים קריטיים ולחזות התנהגות מערכת בתנאים שונים. המאגר מקל על זיהוי מטרות טיפוליות פוטנציאליות על ידי מתן מודלים אמינים לרשתות ביולוגיות הקשורות למחלות.

סיכום

בסך הכל, המאמר תורם תרומה משמעותית לתחום הביולוגיה החישובית. יצירת מאגר עקבי ונגיש מבחינה לוגית של מודלים של רשתות בוליאניות היא משאב רב ערך לחוקרים. צינור האימות והכלים המסופקים יעילים מאוד בהבטחת מהימנות המודלים. עם זאת, הכללת אימות ניסיוני יותר ומאמצים להגדיל את מורכבות המודל עשויה לשפר עוד יותר את השפעת המאגר.

מאגר ה- GitHub ובחירת מודל:

ניתן למצוא את המאגר ב-GitHub תחת הפרויקט sybila/biodivine-boolean-models בקישור הבא: github_biodivine-boolean-models. פרויקט זה מכיל ספריות עם מודלים בודדים, כל אחד מלווה בנתונים וקבצי מקור. ניתן לנווט אל ספריית המודלים (תיקיית models) כדי לחקור את המודלים הזמינים.

אנחנו בחרנו במודל "T-LGL Leukemia Network".

מודל זה הוא מודל לוקמיה מסוג T-cell large granular lymphocyte (LGL). המאמר המקורי נקרא: "Network Model of Survival Signaling in LGL Leukemia" מאת: Ranran Zhang 1, Mithun Vinod Shah, Jun Yang, Susan B Nyland, Xin Liu, Jong K Yun, Réka Albert, Thomas P Loughran Jr.[1] מאמר זה פורסם ב-PNAS בשנת 2008. מאמר זה מתאר בנייה וניתוח של מודל רשת המדמה מסלולי איתות הישרדות בתאי לוקמיה של LGL ומתן תובנות לגבי מנגנוני הרגולציה של המחלה.

סיכום המאמר:

נתחיל מסקירה כללית של התרומות העיקריות של המאמר:

המאמר מציג מודל רשת מקיף לאיתות הישרדות בלוקמיה לימפוציטים גרגיריים גדולים (LGL) של תאי T. מחברי המאמר משתמשים בגישת רשת בוליאנית כדי להדגים את האינטראקציות בין מולקולות איתות שונות ומסלולים המעורבים בלוקמיה של LGL. התרומות העיקריות של המאמר כוללות זיהוי של צמתים רגולטוריים מרכזיים ואינטראקציות קריטיות להישרדות תאי לוקמיה של LGL. המודל עוזר להבין את המנגנון המעורב בהתפתחותה של המחלה ומציע מטרות טיפוליות פוטנציאליות.

לסיכום, התרומות העיקריות כוללות:

- פיתוח מודל: בניית מודל רשת בוליאני מקיף הלוכד את המרכיבים והאינטראקציות העיקריות בתוך רשת איתות ההישרדות של לוקמיה מסוג T-LGL.

[1] קישור למאמר: https://www.pnas.org/doi/full/10.1073/pnas.0806447105

- <u>סימולציה וניתוח</u>: המודל משמש כדי לדמות את התנהגות הרשת בתנאים שונים, זיהוי צמתים ומסלולים קריטיים המווסתים את הישרדות התא ואפופטוזיס (מנגנון גנטי האחראי להשמדה עצמית של תאים בגוף).
- <u>תובנות ביולוגיות</u>: הממצאים מספקים תובנות לגבי מטרות טיפוליות פוטנציאליות על ידי הדגשת צמתים שעיכוב או הפעלתם יכולים להשפיע באופן משמעותי על ההישרדות של תאי לוקמיה מסוג T-LGL.

חסרונות במאמר ושיפורים:

אחד החסרונות הפוטנציאליים של המאמר הוא ההסתמכות על מודלים סטטיים של רשת בוליאנית, מה שעשוי לפשט יתר על המידה את האופי הדינמי והאקראי של מערכות ביולוגיות. בעוד שמודלים בוליאניים הם שימושיים ללכידת המבנה הלוגי של רשתות רגולטוריות, ייתכן שהם לא מסבירים באופן מלא את ההיבטים הכמותיים והזמניים של אינטראקציות מולקולריות.

חסרון נוסף של המאמר הוא האימות הניסיוני המוגבל של תחזיות המודל. בעוד שהניתוח החישובי מספק תובנות חשובות, שילוב נתונים ניסיוניים נוספים כדי לאמת את התחזיות של המודל עשוי לשפר את אמינותו והיישום שלו. עבודה עתידית יכולה להתמקד בבדיקה ניסיונית של הצמתים והמסלולים הקריטיים שזוהו כדי לאשש את ממצאי המודל.

לסיכום המאמר, המאמר מהווה תרומה משמעותית לתחום הביולוגיה החישובית וחקר הסרטן. השימוש במודל של רשת בוליאנית מספק תובנות חשובות לגבי מנגנוני הרגולציה של לוקמיה מסוג LGL. היכולת של המודל לזהות צמתים רגולטוריים מרכזיים מציעה דרכים פוטנציאליות להתערבות טיפולית. עם זאת, ניתן לחזק את המאמר על ידי שילוב גישות של מודלים דינמיים יותר כדי ללכוד את ההתנהגות הזמנית של רשתות האיתות בצורה מדויקת יותר. בנוסף, הכללת אימות ניסיוני יותר תחזק את הרלוונטיות המעשית של המודל.

ניתן לדמות את המודל המתואר במאמר באמצעות הכלי Booleanet, התומך בסכימות עדכון שונות ומאפשר חקירה של התנהגויות דינמיות שונות. על ידי ביצוע הוראות ההתקנה והסימולציה הניתנות במאגר, החוקרים יכולים לשחזר את התוצאות העיקריות של המאמר המקורי ולאמת את התחזיות של המודל. הגמישות של הכלי בטיפול בסכמות עדכון והגדרות פרמטרים שונות הופכת אותו למשאב רב עוצמה לחקר רשתות רגולטוריות מורכבות במערכות ביולוגיות.

לסיכום, המאגר והמאמר המקורי על מודל לוקמיה מסוג T-cell LGL מספקים משאבים ותובנות יקרי ערך עבור קהילת המחקר, ומקדמים את הפיתוח והאימות של מודלים חישוביים בביולוגיה של מערכות.

כעת ננסה להריץ את המודל:

שלבים להרצת המודל:

תחילה, נוריד את המודל מהגיטהאב באמצעות הפקודה הבאה:

Wget
https://raw.githubusercontent.com/ialbert/booleannet/master/examples/projects/LGL/LGL-simulation.py

ולבסוף נריץ את המודל על ידי הפקודה:

python LGL-simulation.py

לאחר שהרצנו את הפקודות קיבלנו את התוצאה הבאה בטרמינל:

```
- starting simulation with REPEAT=10, STEPS=50
- completed
- completed
- completed
- completed
- data saved into LGL-run.bin
```

כפי שניתן לראות הסימולציה רצה בהצלחה והנתונים נשמרו בקובץ LGL-run.bin. כדי לשחזר את התוצאות העיקריות ולהמחיש אותן, נצטרך לטעון את הנתונים השמורים ולהשתמש בהם כדי ליצור גרפים או סטטיסטיקות. ספריית boolean2 שסופקה על ידי תיקיית הגיט עובדת בדרך כלל עם אספנים האוספים את מצב הצמתים לאורך זמן, ומאפשרים לנתח את התנהגות הרשת.

ננסה ליצור גרפים על ידי השלבים הבאים:

- טעינת הנתונים: הנתונים נשמרו בפורמט בינארי בשיטת util.bsave. נטען נתונים אלה באמצעות שיטת util.bload שסופקה על ידי ספריית boolean2.
- הדמיה או ניתוח הנתונים: לאחר טעינת הנתונים, ננתח אותם באמצעות ספריית matplotlib.

להלן קטע קוד המדפיס מצב של קודקוד בגרף לאורך זמן:

```
import matplotlib.pyplot as plt
import boolean2
from boolean2 import util

# Load the data
data = util.bload('LGL-run.bin')

# Inspect the structure of the data
if not data:
    print("No data loaded.")
    exit(1)

print("Data type:", type(data))  # Should be list
print("First item type:", type(data[0]))  # Should be dict

# Print the keys of the first few items for debugging
for i, run in enumerate(data[:3]):  # Inspect the first 3 runs
    if isinstance(run, dict):
        print(f"Run {i+1} keys: {run.keys()}")
    else:
        print(f"Run {i+1} is not a dict, found type: {type(run)}")

node_name = 'Apoptosis'  # Replace with the node you are interested
in

# Initialize time_series
```

```
time_series = []

# Process each run
for run in data:
    # Check if run is a dictionary
    if isinstance(run, dict):
        # Check if node_name is in run keys
        if node_name in run:
            state = run[node_name]
            time_series.append(state)
        else:
            print(f"Node '{node_name}' not found in run.")
    else:
        print(f"Unexpected run data type: {type(run)}")

# Check if we have collected data
if not time_series:
    print(f"No data collected for node '{node_name}'.")
    exit(1)

# Create a plot
plt.plot(time_series)
plt.xlabel('Time')
plt.ylabel(f'State of {node_name}')
plt.title(f'State of {node_name} over time')
plt.show()
```

גרף עבור הקודקוד Apoptosis: