Technion Israel Institute of Technology
CRML Control Robotics and Machine Learning Laboratory
ANDREW AND ERNA VITERBI FACULTY OF ELECTRICAL ENGINEERING

המעבדה לבקרה רובוטיקה ולמידה חישובית

ספר פרויקט

אלגוריתם לניהול נחיתות והמראות

מבצעים:

טום-אבי שפירא Tom-Avi Shapira

בר ממרן Bar Mamran

מנחה:

אייל טייטלר Eyal Taitler

סמסטר רישום: אביב תש"ף

תאריך הגשה: יולי, 2020

מספר פרויקט: 5537

תודות

אנו מודים מקרב לב לאייל טייטלר, המנחה שלנו, על התמיכה הרבה והעזרה ללא רבב לכל אורכו של הפרויקט.
בנוסף אנו מודים לאדיר ירמיהו על הצעת הפרויקט מטעם חיל האוויר.
כמו כן, אנו מודים לקובי קוחאי ולמעבדת CRML על הסיוע הרב.

תוכן עניינים

רשימת איורים

תקציר

תכנון הוא יכולת אנושית חשובה, החל מהגילאים הצעירים. כדי לבצע משימה זו, על המוח לסנכרן ולשלב כמה מערכות קוגניטיביות, כגון : שליטה ועיבוד הנתונים המתקבלים.
בפרויקט זה נממש אלגוריתם המבצע תכנון יעיל במטרה למצוא סדר של המראות ונחיתות מטוסים. זאת כאשר ההיבט המרכזי הוא מציאת סדר אופטימלי אשר יגרור עבור כל מטוס מציאת זמני המראה/נחיתה מינימליים.

את האלגוריתם פיתחנו בסביבת python תוך שילוב ממשק GUI לנוחיות המשתמש.
בבסיסו האלגוריתם צריך לדעת לקבל נתוני קלט אודות זמני ההמראה והנחיתה של המטוסים ולהוציא את סדר הפעולות האופטימלי שלהם, כולל זמני הפעולות של כל מטוס. בנוסף עליו לספק יכולת תכנון מחדש במקרה של תקלות חירום שונות.

Abstract

Planning is an important human ability, starting at a young age. To perform this task, the brain must synchronize and combine several cognitive systems, such as controlling and processing the resulting data.

In this project, we will implement an algorithm that performs efficient planning in order to find an order of the takeoffs and landings of aircraft. This is when the main aspect is finding the optimal order which will result for each aircraft finding minimum takeoff / landing times.

We developed the algorithm in Python environment while combining a GUI interface for user convenience.

At its core, the algorithm needs to know how to obtain input data about the take-off and landing times of the aircraft and extract their optimal order of operations, including the operating times of each aircraft. In addition, it must provide redesign capability in case of various emergency malfunctions.

1. מבוא

תכנון (חיפוש) הוא משימה מורכבת. במשימה זו האלגוריתם צריך לבצע חיפוש על כמות גדולה מאוד של צמתי גרף וזאת במטרה למצוא מסלול מצומת התחלה לצומת מטרה כלשהו בגרף. כמו כן, במהלך תהליך החיפוש האלגוריתם צריך להיות מסוגל לפסול מסלולי פתרון לא אפשריים. נציין כי תהליך החיפוש מתבצע תחת מגבלות זמן וכוח עיבוד.

פרויקט זה נולד כתוצאה מצורך בסיסי לסדר לפי אילוצי נתוני טיסות שונות את סדר המראת\נחיתת המטוסים וסדר הפעולות שכל מטוס צריך לעשות.
הצורך הגיע מחיל האוויר, בו קיום הדבר מתבצע ידנית ולרוב בצורה לא אופטימלית מבחינת זמנים.

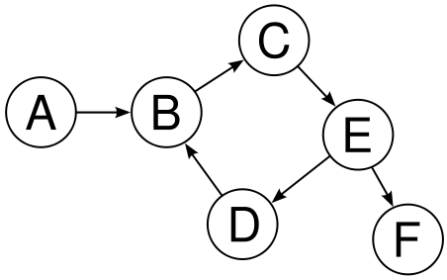
בפרויקט זה מטרתנו לענות על הצורך הנ״ל תוך מימוש אלגוריתם תכנון מתאים. כלומר, מטרת הפרויקט היא לייצר אלגוריתם שבהינתן אילוצי המטוסים, כמות המסלולים וכמות המטוסים יוכל לקבוע את סדר הפעולות שימזער הן את הזמן המינימלי לפעולת כל מטוס בנפרד והן את זמן מסלול הפתרון הכולל עבור כל המטוסים יחדיו.

האתגר בפרויקט זה הוא מימוש אלגוריתם יעיל ואמין למציאת הסדר אופטימלי תוך שימוש בהיוריסטיקה שמתאימה לבעיה. נרחיב על כך רבות בהמשך.

2.סקר ספרות

2.1. גרף מכוון

גרף הוא ייצוג מופשט של קבוצה של אובייקטים, כאשר כל זוג אובייקטים בקבוצה עשויים להיות מקושרים זה לזה. האובייקטים הניתנים לקישור מכונים צמתים והקישורים בניהם מכונים קשתות. כאשר ישנה משמעות לכיוון הקשתות הגרף נקרא מכוון, במקרה זה ניתן להגדיר את בניו של צומת להיות הצמתים אליהם מובילות הקשתות היוצאות מהצומת.
נציין כי מרחק של צומת מצומת ההתחלה נקבע לפי מספר הקשתות הנמצאות בניהם.
באמצעות גרף ובפרט גרף מכוון ניתן לייצג בעיית רבות. בפרויקט זה באופן כמעט טבעי בחרנו לייצג את הבעיה בצורת גרף מכוון, נרחיב על כך רבות בהמשך.



איור 1 - דוגמה לגרף מכוון

2.2. תכנון

נגדיר תחילה, תוכנית היא רצף פעולות אשר מביאות מערכת כלשהי ממצב התחלתי למצב היעד שלה. מטרת התוכנית יכולה להשתנות כתלות בבעיה לפי הבאים:

1. קביעה האם ישנו פתרון לבעיה אותה מנסים לפתור.
2. מציאת פתרון כלשהו (יכול להיות אחד מרבים).
3. מציאת פתרון אופטימלי (או תת אופטימלי).
4. מציאת חסם (או פתרון מדויק) לזמן ריצת התוכנית.

בעוד שהמשימות המוצגות לעיל נראות מאוד קרובות, ביצוען דורש מימוש טכניקות שונות ומגוונות בהן: תכנון, תכנון עם גרף דינאמי ותכנון עם זמנים ואילוצים.

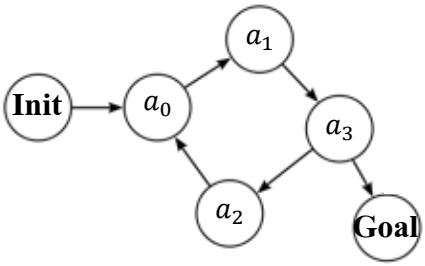
שנית נגדיר, מערכת חיפוש דטרמיניסטית היא מערכת הפועלת על צמתים בגרף באמצעות פעולות לצורך הגעה למצב סופי.
מערכת חיפוש דטרמיניסטית מגדירה את $< S, I, \{a_1, \ldots, a_n\}, G >$ כאשר:
S- סט סופי של המצבים האפשריים.
$I \in S$ - מצב התחלתי של הבעיה (שייך לקבוצת המצבים).
$\{a_1, \ldots, a_n\}$ -קבוצת הפעולות המעבירה בין מצבים בתוך קבוצת המצבים S.
G - קבוצת מצבי היעד הרצויים.

כעת, נוכל להגדיר את בעיית התכנון.
בבעיית התכנון נרצה למצוא תוכנית באופן הבא:

$$\pi = \{a_1, \ldots, a_n \ , \quad s_0, \ldots, s_n\}$$

הקבוצה **π** היא קבוצת פעולות $a_1, \dots, a_n$ המעבירות אותנו מ s_0 ל s_n, דרך מצבי ביניים $s_1, \dots, s_{n-1}$. מתוכה נרצה למצוא את סדר רצף הפעולות שיביא אותנו למצב היעד s_n.

תחת ההגדרות הנ"ל ניתן להציג את בעיית התכנון באופן טבעי כגרף מכוון באופן הבא:
קבוצת הפעולות $\{a_1, \dots, a_n\}$ תהווה את צמתי הגרף (פרט לצומת ההתחלה והמטרה) ומכל צומת ייגזר באופן ישיר מצב הבעיה הנוכחי s_i וזאת בהתאם לפעולות שהתבצעו עד אותו צומת עם חשיבות לסדרן כמובן.
חשוב לציין כבר בשלב זה כי יתכן יותר ממסלול אחד בגרף המוביל לקבוצת מצבי היעד G ומנגד יתכנו מסלולים שבהכרח לא יובילו לקבוצת מצבי היעד G, כלומר לא יובילו לפתרון. נשתמש בכך בתת הפרקים הבאים.

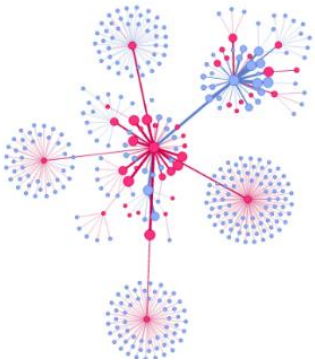


איור 2 - דוגמה להצגת הבעיה דרך גרף מכוון בעל צומת התחלתית, צומת מטרה וקבוצת פעולות

2.3. תכנון עם גרף דינאמי

בתכנון עם גרף דינאמי אנו נתקלים בבעיה חדשה שלא הוצגה בתכנון קלאסי. בגישה זו אנו מנסים לייצג בעזרת גרף בעיה גדולה מאוד שלא נרצה להגדיר את כולה מראש במטרה לחסוך במשאבי המחשב (למשל זיכרון) שאנו מוגבלים בהם. הבסיס לגישה זו הוא ההבנה שבהינתן צומת נוכחי נוכל לפרוש את בניו ובכך נוכל לבנות את גרף הבעיה בצורה דינאמית, שלב אחר שלב. כמו כן, חשוב לציין כי ברוב המקרים לא נצטרך את כל מרחב המצבים העומד לרשותנו אלא רק חלק ממנו וזאת מפני שהרבה כיוונים בבעיה אינם טובים לנו ולכן נוכל לפסול אותם מראש ובכך לייעל את הכוח החישובי העומד לרשותנו.

את הגרף הדינאמי נבנה באמצעות פונקציה, נגדיר את שמה להיות פונקציית ה successor.
בהינתן מצב נוכחי מקבוצת המצבים הסופית S פונקציה זו חושפת את כל המצבים החוקיים העוקבים לו.
חשוב להבין שמעצם הגדרתה פונקציה זו חושפת מצבים חוקיים בלבד, כלומר מצבים אשר יכולים להיות חלק מפתרון הבעיה הסופי (עם חשיבות לסדר בו נחשפו). כך אנו מייעלים את כוח החישוב ומשיגים את המטרה שהוגדרה בתחילת תת פרק זה.
בפועל, כפי שכבר ציינו הבעיה מיוצגת על-ידי גרף מכוון ולכן פונקציית ה- successor נקראת עבור צומת בגרף. למעשה באמצעותה אנו מגלים לכל צומת את בניו החוקיים בגרף, כלומר מהן הפעולות החוקיות שניתן לבצע ממנו. הדינאמיות באה לידי ביטוי בכך שבכל שלב נחשוף בנים של צומת אחר וכך נבנה את הגרף שלב אחר שלב תוך מציאת מסלולים חוקיים לפתרון הבעיה.



איור 3 - דוגמה לעץ הנפרש דינאמית.

באיור 3 ניתן לראות דוגמה לעץ הנפרש דינאמית. במרכז ניתן לראות את שורש העץ (המצב ההתחלתי). ממנו יוצאים בניו (הנקודות הוורודות) ומהם יוצאים ילדיהם (הנקודות הכחולות). גילוי כל אחד מהדורות מתבצע באמצעות פונקציית ה successor.

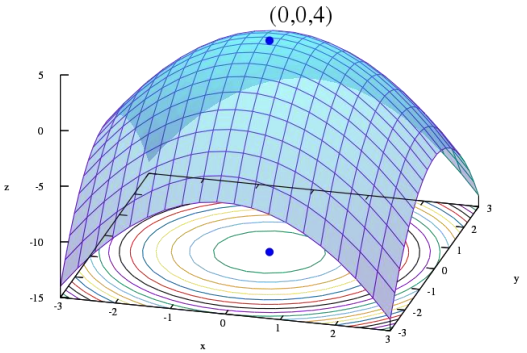
2.4. תכנון עם אילוצי זמנים

תכנון עם זמנים ואילוצים בשונה מתכנון קלאסי מתמקד בבעיה בה לכל פעולה יש התחלה וסוף. בעצם נוסף לבעיה ממד המעניק לפעולות מסוימות זמן ביצוע מסוים. בתכנון עם זמנים יש צורך לקבוע את זמני התחלת וסיום הפעולות וכן את האילוצים בין הפעולות השונות. לצורך פשטות נהוג לקבוע את כלל הזמנים ביחס לזמן תחילת הבעיה אשר נקבע להיות 0.
דוגמה לאילוץ אפשרי: בבעיה בה כלל הזמנים מוגדרים ברזולוציה של דקות. כאשר מטוס ממריא במשך 20 דקות בין זמן 0 לזמן 20 לא אפשרי שמטוס אחר ינחת באותו המסלול ובאותו חלון זמנים בו המריא הראשון ולכן צריך להיות סדר בין הפעולות ואילוצים בין המטוסים.

נציין כי בעיית הזמנים המאולצת היא בעיית אופטימיזציה עם אילוצים לכל דבר ועניין. לכן להגדרתה ומציאת פתרון חוקי כנדרש עשינו שימוש בכלי אופטימיזציה מתאים.
עוד נציין כי האילוצים כשלעצמם בדרך-כלל פוסלים הרבה כיווני פתרון ובכך מקטינים משמעותית את מרחב המצבים האפשרי של הבעיה.

2.4.1. בעיית אופטימיזציה עם אילוצים

אופטימיזציה היא תת תחום במתמטיקה העוסק במציאת הערך האופטימלי של פונקציה תחת מגבלות אילוצים נתונות. בבעיות מסוימות הערך האופטימלי הוא הערך המינימלי שמקבלת הפונקציה הנתונה תחת הגבלות האילוצים, ובבעיות מסוימות זהו הערך המקסימלי. בפתרון בעיות אופטימיזציה טמון קושי רב במציאת הערך האופטימלי וזאת כתלות בפונקציה המבוקשת ובתחום בו יש לבצע אופטימיזציה (האילוצים).



איור 4 - דוגמה לבעיית אופטימיזציה עם פתרון אופטימלי

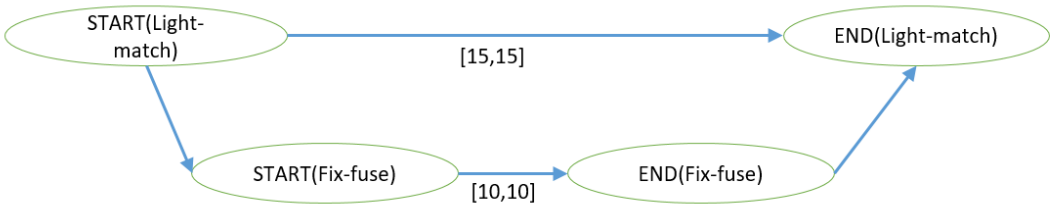
באיור 4 ניתן לראות דוגמה לבעיית אופטימיזציה הנתונה על-ידי גרף פרבולואיד בריבוע [-3,3]. המקסימום נמצא בנקודה המסומנת (0,0,4).

בפרויקט זה נפתור בעיית אופטימיזציה לינארית, כלומר בעיית אופטימיזציה בעלת פונקציה ואילוצים לינאריים. כמו כן, לפתרון בעיית האופטימיזציה עשינו שימוש בכלי אופטימיזציה מובנה מבית Gurobi.

2.4.2. Simple Temporal Network (STN)

את פתרון בעיית האופטימיזציה שתוארה בתחילת תת פרק 2.4 נהוג להציג בגרף הנקרא גרף STN. זהו למעשה גרף המייצג את הממד הזמני של הפעולות שנבצע בפתרון הבעיה, כאשר כל פעולה מיוצגת על-ידי זמן ההתחלה היחסי שלה (כפי שהוגדר בתחילת תת פרק 2.3) והזמן הכולל הדרוש לביצועה.
חשוב לציין כי מעצם היותו פתרון בעיית אופטימיזציה, הפתרון המוצג ב STN הינו פתרון אופטימלי עבור מסלול הפתרון עבורו נפתרה בעיית האופטימיזציה, אם כי אינו המסלול האופטימלי מבין כל המסלולים האפשריים המובילים לפתרון חוקי של הבעיה.

נדגים בעזרת בעיית האופטימיזציה הבאה:
תיקון פיוז לוקח בדיוק 10 דקות. התיקון חייב להתבצע כאשר הוא מלווה באור הבוקע מנר אשר יכול להיות דלוק למשך 15 דקות לכל היותר, זהו האילוץ.
יש למצוא פתרון אופטימלי הממזער את הזמן היחסי בו תיקון הפיוז הושלם.
פתרון הבעיה הנ"ל מתואר בגרף ה stn באיור 4 מטה:



איור 5 -פתרון בעיית האופטימיזציה הנ"ל מוצג בגרף stn.

ניתן לראות כי לפי הפתרון שהתקבל בגרף ה stn באיור 4 תיקון הפיוז החל אפסילון זמן לאחר הדלת הנר ונמשך 10 דקות.

2.5. חיפוש בגרף

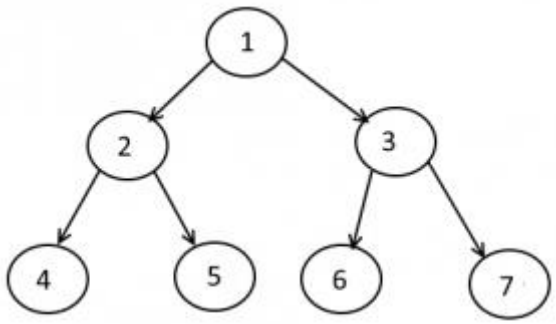
עד כה תיארנו את תהליך התכנון הבסיסי תוך התייחסות לתכנון דינאמי ותכנון משולב אילוצים ובפרט אילוצים זמניים.
בתת פרק זה נדון בתהליך החיפוש בגרף בעיה דינאמי משולב אילוצים. כלומר, נתאר את תהליך חיפוש מסלול פתרון חוקי כלשהו מצומת ההתחלה לצומת המטרה בגרף.

2.5.1. אלגוריתם Breadth first search (BFS)

אלגוריתם BFS (אלגוריתם חיפוש לרוחב) הוא אלגוריתם המשמש למעבר על צמתי גרף, לרוב תוך חיפוש צומת המקיים תכונה מסוימת (במקרה שלנו צומת המטרה).
תהליך החיפוש:

1. יש לקבוע צומת התחלה
2. האלגוריתם עובר על כל הצמתים במרחק 1 מצומת ההתחלה.
3. האלגוריתם עובר על כל הצמתים במרחק 2 מצומת ההתחלה.
4. וכן הלאה עד שאין יותר צמתים בגרף.

בפועל האלגוריתם ממומש באמצעות מבנה הנתונים תור (Queue) המכיל תחילה את צומת ההתחלה. כעת, בכל שלב מוציאים את הצומת העומד בראש התור, מגלים את בניו בעזרת פונקציית ה successor (כפי שהוגדרה בתת פרק 2.3) ומכניסים את הבנים שהתגלו לסוף התור.
בדרך זו מתקבל כי הגרף נבנה דינאמית ותהליך החיפוש בו הוא רוחבי.
התהליך נעצר כאשר מוציאים מראש התור את צומת המטרה או כאשר התור מתרוקן.



איור 6 - דוגמה לחיפוש רוחבי בגרף. (סדר החיפוש בהתאם למספר הצמתים).

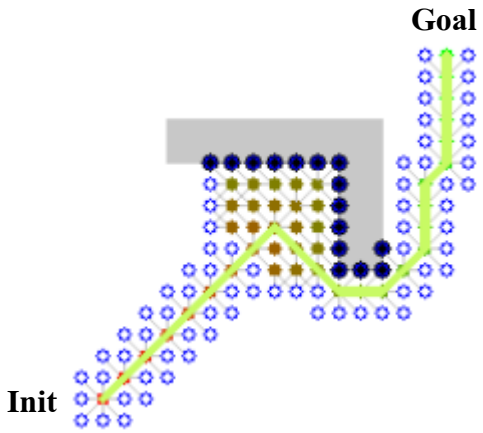
נציין כי בבעיות בהן מרחב המצבים גדול מאוד נקבל שחיפוש רוחבי אינו יעיל שכן אינו מוכוון מטרה ובהינתן כוח חישוב בסיסי לא ניתן להגיע לצומת המטרה בזמן סביר.
תת הפרק הבא מתמודד עם בעיה זו.

2.5.2. היוריסטיקה (Heuristic)

היוריסטיקה היא כל גישה לפתרון בעיות או גילוי עצמי, שמפעילה שיטה פרקטית שלא מבטיחה פתרון אופטימלי, מושלם או רציונלי, אלא מבטיחה תנאים מספיקים כדי להגיע לפתרון מיידי.
כאשר מציאת פתרון אופטימלי אינו אפשרי או פרקטי, השיטה ההיוריסטית יכולה לסייע להאיץ את תהליך מציאת פתרון סביר.
בפועל, תפקיד ההיוריסטיקה הוא לכוון את תהליך החיפוש בגרף הדינאמי שאנו בונים, כך שיתקבל פתרון טוב יותר ובזמן מהיר יותר. כמובן שיש תחילה להגדיר מהו פתרון טוב ובהתאם להגדיר את ההיוריסטיקה בבעיה.

2.5.3. אלגוריתם A-Star (A*)

אלגוריתם A* הוא אלגוריתם חיפוש BFS מונחה היוריסטיקה על צמתי גרף ממושקל.
האלגוריתם מחפש מסלול לצומת המקיים תכונה מסוימת, זהו צומת המטרה. זאת תוך מציאת המסלול הקצר ביותר מצומת ההתחלה לצומת המטרה. ניתן לחשוב עליו כעל אלגוריתם חיפוש רוחבי בעל תור עדיפויות. כלומר כל פעם שמכניסים את בניו של צומת לתור, הם נכנסים בסדר הנקבע לפי ההיוריסטיקה. כך הפתרון הכולל מוכוון.
האלגוריתם תואר לראשונה ב-1968 על ידי פיטר הרט, נילס נילסון וברטהם רפאל ממרכז המחקר של אוניברסיטת סטנפורד.



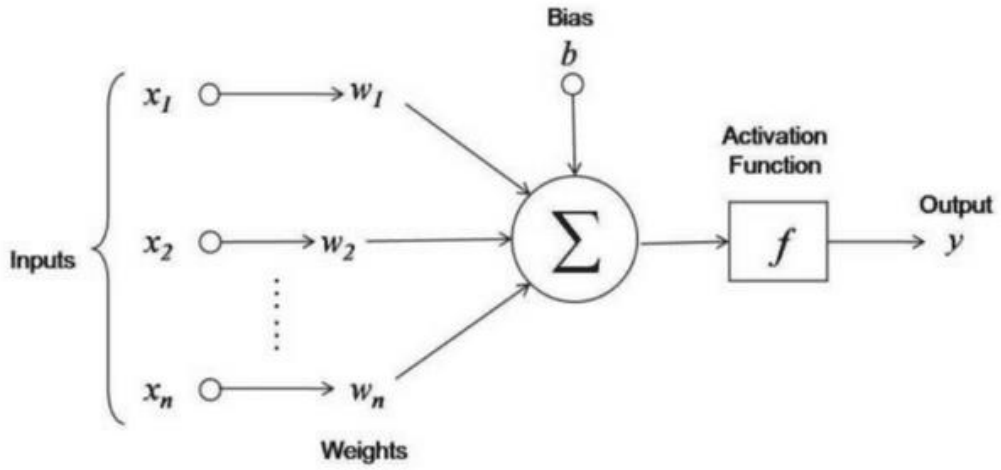
איור 7 - הדגמה גרפית לתוצאת אלגוריתם A עם אילוץ מרחבי.*

באיור 7 ניתן לראות הדגמה לתוצאת אלגוריתם A* אשר מוכוון למצוא את המסלול הקצר ביותר בין צומת ההתחלה לצומת המטרה. וזאת תחת האילוץ המרחבי המוצג כמחסום באיור.

2.6. רשת נוירונים

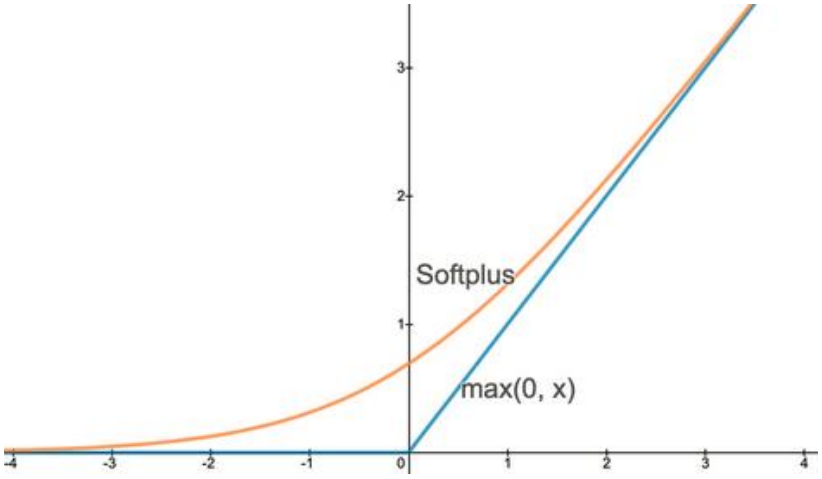
2.6.1. הנוירון הבודד

רשת נוירונים היא בעצם מודל מתמטי חישובי המורכב ממספר מסוים של יחידות חישוב, אלו הם הנוירונים. כל נוירון ברשת מסוגל לבצע פעולה מתמטית פשוטה כמתואר באיור הבא:



איור 8 - סכמה המדמה פעולה של נוירון בודד.

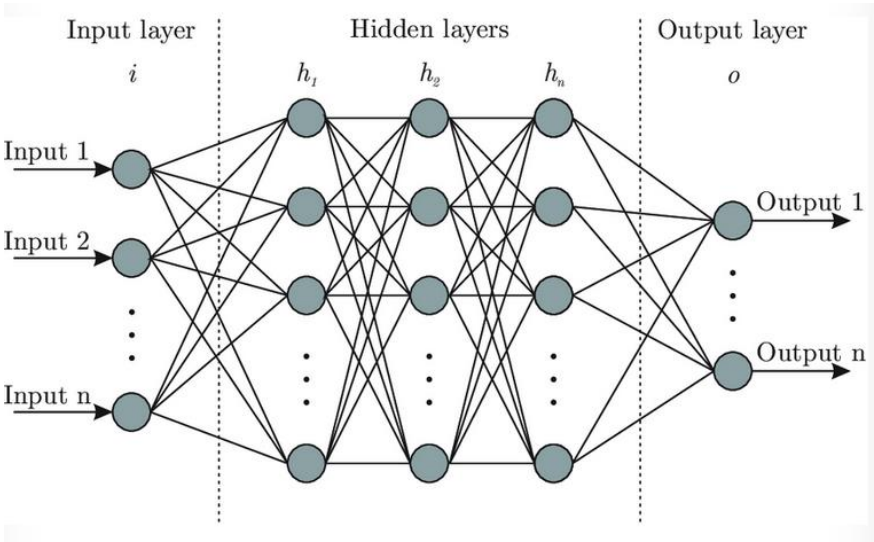
למעשה כל נוירון מבצע סכום משוקלל של כניסותיו בתוספת bias כלשהו, אשר מועבר בפונקציית אקטיבציה לא לינארית כלשהי.
משקלי כל נוירון, כולל ה bias אלו הם הפרמטרים שנכוון בתהליך הלמידה בהמשך.
אנו נשתמש בפונקציית האקטיבציה softplus (החלקה של פונקציית Relu): $y = \ln(1 + e^x)$



איור 9 - פונקציית Relu ופונקציית softplus.

2.6.2. רשת כללית (fully connected)

בשילוב מספר נוירונים נקבל רשת נוירונים אשר מסוגלת ללמוד ולקרב כל פונקציה שהיא.
כפי שציינו, הלמידה היא בכוונון פרמטרי הרשת, אלו הם משקלי כל נוירון כולל ה bias.
נציין כי כאשר ישנו חיבור בין כל שני נוירונים בשכבות עוקבות, הרשת מכונה fully connected.



איור 10 - רשת נוירונים כללית (fully connected).

כפי שניתן לראות באיור 10, רשת נוירונים מורכבת משכבות של נוירונים – שכבת כניסה ,שכבת מוצא ,ואחת או יותר שכבות פנימיות אשר מכונות שכבות נסתרות. כל נוירון בשכבה מסוימת מחובר עם משקל לכל נוירון בשכבת הנוירונים הבאה.
כפי שכבר ציינו, הלמידה באה לידי ביטוי בכוונון משקלי הנוירונים ובפועל היא נעשית באמצעות אלגוריתם ה back propagation (פעפוע לאחור). זהו האלגוריתם הידוע והנפוץ ביותר לכוונון רשת עצבית רב שכבתית. כעיקרון מדובר באלגוריתם גרדיאנט לעדכון המשקלים ברשת. התחכום הוא באופן חישוב הגרדיאנט וזאת באמצעות כלל השרשרת המתבצע בצורה יעילה.
בפועל האלגוריתם מחשב את הגרדיאנט של השגיאה במוצא לפי כל משקל.

2.6.3. אימון רשת עצבית

אנו נתמקד בלמידה מודרכת בה כוונון פרמטרי הרשת מתבצע על סמך דוגמאות מתויגות. כלומר, על סמך סט אימון הכולל קלטים ופלטים מתאימים נתונים נלמד את הפונקציה המתאימה להם.

טרם תהליך הלמידה חשוב לבצע את שני השלבים המקדימים הבאים:

1. נאתחל את משקלי הרשת. (ניתן לדון בנושא זה רבות במסגרת פרויקט זה נניח אתחול נתון).
2. נגדיר פונקציית שגיאה – L, שתגדיר את מידת שגיאת פלט הרשת מהתוצאה הרצויה (כפי שהיא נתנוה בסט האימון).

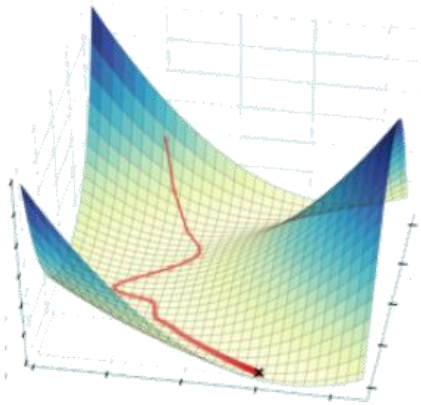
   אנו נשתמש בפונקציית השגיאה הריבועית: $L = (y' - y)^2$

   כאשר y הוא הפלט האמיתי כפי שנתון בסט האימון ו y' הוא מוצא רשת הנוירונים עבור הקלט המתאים ל y בסט האימון.

תהליך הלמידה שנבחר בפרויקט זה כולל שלושה שלבים עיקריים:

1. נבחר דוגמה מסט האימון ונחשב את הפלט המתאים לה לפי רשת הנוירונים במצב הנוכחי.
2. נבצע פעפוע לאחור. כלומר נחשב את הגרדיאנט של השגיאה במוצא לפי כל משקל ברשת.
3. נעדכן את המשקלים לפי הכלל הבא: $w_{t+1} = w_t - \eta \left(\frac{\Delta L}{\Delta w_t}\right)$. כלומר נעדכן את המשקולות כך שנתקדם בפונקציית השגיאה בכיוון מינוס הגרדיאנט. כיוון הגרדיאנט לפי ההגדרה הוא הכיוון בו נקבל מקסימום של פונקציה ולכן אם נתקדם בכיוון ההפוך לו נגיע למינימום, כלומר למינימום של פונקציית השגיאה. זאת כנדרש מרשת שלומדת.
נציין כי לא מובטח שהמינימום שנגיע אליו הינו מינימום גלובלי, לא נרחיב על כך במסגרת פרויקט זה.

η הוא גודל צעד העדכון, מהווה היפר פרמטר בבעיה. נציין כי גודל צעד טיפוסי הוא 0.0025.
כמו כן, בפרויקט זה נתמקד באימון בשיטת SGD, בה מעדכנים את המשקולות כל פעם לפי דוגמה אחת בלבד.



איור 11 - דוגמה להתכנסות השגיאה למינימום מקומי.

2.6.4. אימון בשיטת cross-validation:

נשתמש בשיטת אימון זו כאשר סט האימון שלנו אינו מכיל מספיק דוגמאות. למעשה נחלק את סט האימון שלנו באופן אקראי ל- K קבוצות זרות ושוות גודל (בקירוב).
עבור כל אחת מ- K הקבוצות מתבצע התהליך הבא:

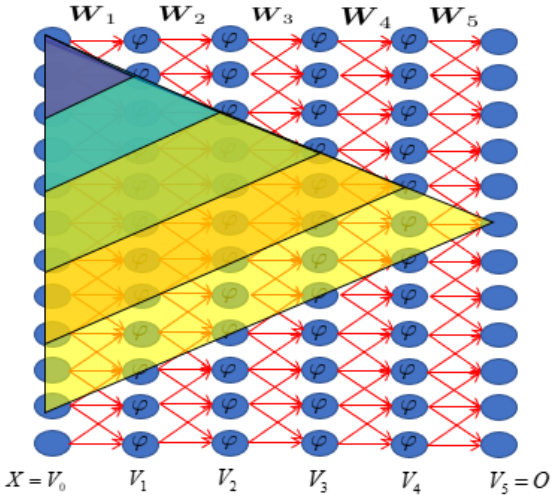
1. שלב האימון מתבצע על כלל הדוגמאות פרט לאילו השייכות לקבוצה K.
2. הדוגמאות השייכות לקבוצה K מהוות סט בוחן עליו נחשב את ביצועי אלגוריתם הלמידה.

נוכל לחזור על התהליך מספר פעמים.

בסיום האימון נמצע את ביצועי אלגוריתם הלמידה שחישבנו בנקודה 2 על פני K הקבוצות, זהו המדד הסופי להערכת ביצועי תהליך הלמידה הכולל.
נציין כי מדד מקובל להערכת ביצועי אלגוריתם הלמידה הוא מדד פונקציית השגיאה הריבועית המוזכר בתת הפרק הקודם.

2.6.5. רשת קונבולוציה

רשת קונבולוציה היא למעשה רשת נוירונים בה כל נוירון מחובר רק לחלק מהנוירונים בשכבה הקודמת. כלומר ישנה חיבוריות מקומיות במקום חיבוריות fully connected.



איור 12- הדגמה לחיבוריות מקומית ברשת קונבולוציה.

באיור 12 ניתן לראות הדגמה לחיבוריות המקומית הקיימת ברשת קונבולוציה. תכונה מרכזית של רשת קונבולוציה כפי שניתן לראות היא שככול שמעמיקים ברשת הנוירונים חשופים לאזור גדול יותר בקלט.
השימוש הנפוץ ביותר ברשתות קונבולוציה הוא בתחום עיבוד התמונה, שכן ניתן לנצל בצורה טובה ברשת את החיבוריות המקומית בין הפיקסלים בתמונה תוך שימוש בפילטר שיסרוק את התמונה בפעולת קונבולוציה ובכך ילמד לזהות (על ידי כוונון פרמטרי הפילטר שיתבצע לפי המתואר בתת הפרק הקודם) תבניות מסוימות בתמונה.
נציין כי בעצם סריקת כל התמונה על-ידי אותו הפילטר נקבע שכל הנוירונים חולקים את אותן המשקולות. יש בכך חסכון גדול מאוד.
כמו כן, ניתן להרחיב את הנאמר ולשלב מספר פילטרים במקביל.

2.6.6. Graph Convolutional Networks (GCN)

כאשר מדובר בקלט בעל קישוריות מסוימת כמו פיקסלים בתמונה ניתן להגדיר רשת קונבולוציה שמסתמכת על החיבוריות המקומית בין הפיקסלים. כאשר לקלט שלנו אין באופן טבעי חיבוריות מקומית נוכל ליצור אחת כזאת באמצעות GCN.
למעשה נציג את הקלט בצורת גרף וקישוריות הקלט תתואר באמצעות הקשתות שיחברו את צמתי הגרף. בצורה זו נוכל למדל במובן מסוים את הקלט שנקבל לתמונה עם פיקסלים ומכאן השימוש ברשת קונבולוציה הופך להיות טבעי.

3. תיאור האלגוריתם

3.1. תיאור כללי של הבעיה

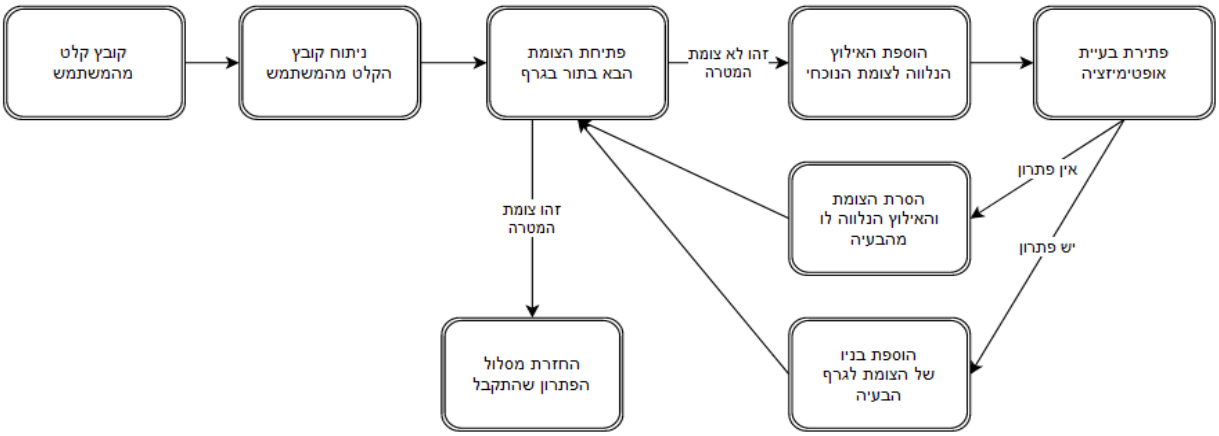
הבעיה אותה נרצה לפתור בפרויקט זה היא בעיית תכנון (planning problem).
בבסיסה יש למצוא מסלול (פתרון) ממצב התחלתי נתון למצב רצוי.
בפועל ובהתאם לנאמר בסקר הספרות בפרק הקודם נייצג את הבעיה בצורת גרף, כאשר נרצה למצוא מסלול המהווה פתרון חוקי מצומת ההתחלה הנתון לצומת המטרה שהוגדר מראש.
בפרויקט זה נרצה לנהל סדר המראות ונחיתות של מטוסים בצורה אוטומטית, יעילה ותת אופטימלית.
כלומר, בהינתן מצב התחלתי של הבעיה, בו חלק מהמטוסים נמצאים על הקרקע ומחכים להמראה וחלק מהמטוסים נמצאים באוויר ומחכים לנחיתה, נרצה למצוא סדר המראות-נחיתות מתאים אשר גורר זמן ביצוע כולל (עד שכל המטוסים נחתו בשלום) מינימלי וזאת בהתאם למסלול הפתרון הנבחר. מפני שאנו מחפשים מסלול פתרון כלשהו בגרף הבעיה ודורשים מינימום זמן ביצוע בו, מדובר בפתרון תת-אופטימלי ולא אופטימלי.

3.2. תיאור כללי של האלגוריתם

את אלגוריתם הפרויקט ניתן לתאר דרך כמה אבני דרך מרכזיות:

1. הגדרת הבעיה.
2. הגדרת קלט הבעיה.
3. מימוש אלגוריתם חיפוש ראשוני על גרף הבעיה.
4. שילוב אילוצים זמניים בבעיה ומימוש גרף STN בעזרת כלי אופטימיזציה מתאים.
5. הגדרת היוריסטיקה מותאמת לבעיה.
6. Learning Heuristic – למידת היוריסטיקה חכמה בעזרת כלים של מערכת לומדת.
7. שילוב ממשק גרפי עבור המשתמש (GUI).

לאחר בחינת הבעיה לעומק תוך הסתמכות על סקר הספרות המוצג בפרק הקודם החלטנו לפתור את הבעיה המוצגת בפרויקט זה על-ידי שימוש בגרף דינאמי ומכוון (כלומר לא כל הצמתים מוגדרים בתחילת התוכנית). זאת תוך שילוב של אילוצים זמניים.
נציין כי הן אופי בניית הגרף הדינאמי והן האילוצים הזמניים נקבעים בזמן אמת על ידי המשתמש עצמו. נרחיב על כך בתת פרק 3.4.



איור 13 - דיאגרמת בלוקים של האלגוריתם.

3.3. הגדרת הבעיה

החלק החשוב ביותר בפרויקט היה הגדרת הבעיה תוך מידול שלה לבעיה המיוצגת באמצאות גרף.

תחילה נגדיר כי תת מצבי הבעיה והפעולות שיוגדרו מטה בהמשך תת פרק זה מהווים את סדר היום של מטוס לפי הסדר בו הם כתובים.

במהלך הגדרת הבעיה נצמדנו להנחות הבאות:

1. מרחב מצבי הבעיה סופי. כלומר, למטוס יש מספר מצבים סופי שהוא יכול להימצא בהם.
2. כמות המטוסים והמסלולים מוגדרת בתחילת הבעיה.
3. המרחב האווירי יכול להיתפס בזמן נתון לכל היותר רק על ידי מטוס אחד (לצורך המראה/נחיתה).
4. מסלול המראה/נחיתה יכול להיתפס בזמן נתון לכל היותר רק על ידי מטוס אחד.
5. תהליכי התיישרות מטוס על מסלול, ההמראה, נחיתה, פינוי מסלול לאחר נחיתה אורכים 10 דקות.
6. בכל רגע נתון מצב הבעיה מורכב מתת מצבים קטנים יותר, כאשר כל תת מצב מייצג מצב של מטוס בודד.
7. לכל תת מצב יש מצב אחד בלבד המתאים להיות עוקבו. זאת מפני שסדר היום של מטוס הוא קבוע וידוע מראש.
8. מטוס יכול להימצא זמן סופי באוויר (פרמטר המייצג דלק).
9. למטוס יש חסם זמני עליון בכל הנוגע לזמן בו לוקח לו לבצע את סדר היום שלו.
10. במצב ההתחלתי של הבעיה כל אחד מהמטוסים יכול להימצא על הקרקע או באוויר. במידה ומטוס נמצא באוויר הוא מבצע חלק מסדר היום שלו בהתאם לתת מצבי הבעיה והפעולות שיוגדרו מטה.
11. הבעיה נפתרת כאשר כל המטוסים הגיעו לתת המצב האחרון בסדר היום שלהם, זהו למעשה צומת המטרה.

כעת, תת מצבי הבעיה (מצב לכל מטוס בנפרד) המגדירים את מצב הבעיה הכולל מוגדרים באמצעות האינדיקטורים והפרמטרים הבאים:

נציין כי כל אינדיקטור דלוק/כבוי (0/1) בהתאם למצב הבעיה.

<u>אינדיקטורים של מטוסים:</u>

p0 – מטוס מוכן להתיישרות על מסלול.
p1 – מטוס תופס מסלול המראה ונמצא במהלך התיישרות על מסלול.
p2 – מטוס סיים להתיישר על מסלול ומוכן להמראה.
p3 – מטוס תופס את המרחב האווירי ומתחיל בתהליך המראה.
p4 – מטוס סיים את תהליך ההמראה ומפנה את מסלול ההמראה שתפס והמרחב האווירי.
p5 – מטוס התחיל את משימתו באוויר.
p6 – מטוס סיים את משימתו באוויר ומוכן לנחיתה.
p7 – מטוס קיבל אישור לנחות, תופס את המרחב האווירי ונמצא בתהליך נחיתה.
p8 – מטוס סיים את תהליך הנחיתה ומפנה את המרחב האווירי.
p9 – מטוס מתחיל בתהליך פינוי מסלול הנחיתה.
p10 – מטוס סיים את תהליך פינוי מסלול הנחיתה.
p11 – המטוס סיים את כל סדר היום שלו.

<u>אינדיקטורים גלובליים:</u>

li – מסלול מספר i תפוס על ידי מטוס מסוים (לצורך המראה/נחיתה).
a – המרחב האווירי תפוס על ידי מטוס מסוים (לצורך המראה/נחיתה).

<u>פרמטרים גלובליים:</u>

num_of_planes - מספר המטוסים בבעיה (ידוע מראש).
num_of_lanes - מספר המסלולים בבעיה (ידוע מראש).
num_of_taken_lanes - מספר המסלולים התפוסים בבעיה.

<u>לאחר הגדרת האינדיקטורים והפרמטרים הנ"ל נוכל להגדיר את הפעולות בבעיה:</u>

פעולות אלה מגדירות לנו את אפשרויות המעבר בין המצבים השונים.

1. Start Clear To Take Off (SCTTO) – מטוס מתחיל את תהליך התיישרות על מסלול ההמראה.
2. End Clear To Take Off (ECTTO) – מטוס מסיים את תהליך ההתיישרות על מסלול ההמראה.
3. Start Take Off (STO) – מטוס מתחיל את תהליך המראה.
4. End Take Off (ETO) – מטוס מסיים את תהליך המראה.
5. Start Mission (SM) – מטוס מתחיל את משימתו באוויר.
6. End Mission (EM) – מטוס מסיים את משימתו באוויר.
7. Start Landing (SL) – מטוס מתחיל את תהליך נחיתה.
8. End Landing (EL) – מטוס מסיים את תהליך נחיתה.
9. Start Taxi (ST) – מטוס מתחיל את תהליך פינוי מסלול הנחיתה.
10. End Taxi (ET) – מטוס מסיים את תהליך פינוי מסלול הנחיתה.

נציין כי ניתן לראות קשר טבעי בין מצבי הבעיה לבין הפעולות שהוגדרו בה.

Successor function .3.3.1

בתת פרק זה נתאר קווים מנחים ליצירת פונקציית ה successor כפי שהוגדרה בפרק סקירת הספרות. בעזרתה נבנה את גרף הבעיה דינאמית.

כפי שכבר ציינו, סדר הפעולות של כל מטוס בנפרד הוא ידוע וקבוע מראש (זהו סדר היום של מטוס). לכן, בהינתן ש-n הוא מספר המטוסים, לכל צומת מועמדים n בנים המהווים פעולת המשך (פעולה לכל מטוס), כאשר בכל שלב של הבעיה ניתן לבחור באחד מהם. לכן, כל שאר הבנים שלא נבחרו יוצרו מחדש בצומת הגרף הבא על ידי פונקציית ה- successor.

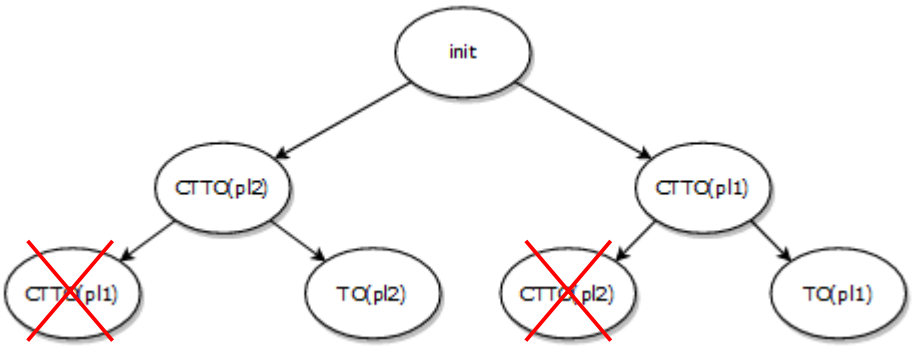
עבור כל מטוס נבחרת פעולת ההמשך לפי מצב האינדיקטורים הנוכחיים שלו. כלומר, לפי כל תת מצב בעיה נסיק שורה של תת מצבים עוקבים אשר ירכיב את מצב הבעיה העוקב הכולל.

כדי לקבוע מי מהמועמדים תקין נבדוק:

1. אם פעולת ההמשך מצריכה את תפיסת המרחב האווירי (a=1) והוא כבר תפוס, נפסול את המועמד.
2. אם פעולת ההמשך מצריכה תפיסת מסלול (li=1) וכל המסלולים כבר תפוסים, נפסול את המועמד.
3. במידה ומועמד נפסל או לא נבחר, המטוס תקוע במצבו בצומת הבעיה הנוכחי ויוכל לנסות להתקדם לפעולה הבאה שלו בצומת הבעיה העוקב שיתקבל.

נציין כי למעשה תוארה פה דרך אחת לפסילת מסלולים בגרף הבעיה הנבנה דינאמית.
בתת פרק 3.6 נתאר דרך נוספת לפסילת מסלולים בעזרת אילוצים זמניים.

באיור 13 מטה ניתן לראות הדגמה להתחלתו של גרף הנבנה דינאמית עבור בעיה הכוללת שני מטוסים ומסלול יחיד. ניתן לראות כי תחילה ניתן ליישר את pl1 או pl2 על המסלול לצורך המראה. וכי מהרגע שאחד מהם התיישר המסלול היחיד בבעיה תפוס ולכן השני לא יכול להתיישר גם. כלומר, צומת זה נפסל.



איור 14 - דוגמה להתחלת גרף הנבנה דינאמית עבור הבעיה המתוארת לעיל, כולל פסילת בנים לא חוקיים.

3.4. הגדרת קלט הבעיה

לפני שנתאר את מימוש אלגוריתם התכנון, נתאר בתת פרק זה כיצד הגדרנו ממשק הקולט את נתוני המשתמש אשר מזין אותם לאלגוריתם התכנון.

קובץ קונפיגורציה חוקי מכיל את הבאים (לפי הסדר משמאל לימין):
נציין כי כל הזמנים בתת סעיפים 1-7 מוגדרים בדקות.

1. מזהה המטוס.
2. הזמן המינימלי (יחסית לתחילת הבעיה) בו המטוס חייב להתחיל להתיישר על מסלול.
3. הזמן המקסימלי (יחסית לתחילת הבעיה) בו המטוס חייב להתחיל להתיישר על מסלול.
4. הזמן שלוקח למטוס לבצע את משימתו באוויר.
5. הזמן המקסימלי בו המטוס יכול להיות באוויר.
6. הזמן המקסימלי (יחסית לתחילת הבעיה) בו המטוס חייב לנחות ולפנות את מסלול הנחיתה.
7. סטטוס המציין האם בתחילת הבעיה המטוס נמצא על הקרקע (1) או באוויר (5).

כמו כן קובץ הקונפיגורציה מכיל את מספר המטוסים, מספר המסלולים וחסם עליון לזמן מציאת פתרון הנמדד בשניות. זאת כדי שהמשתמש יוכל לשלוט בזמן תהליך התכנון ולבצע שינויים בהתאם.

דוגמה לקובץ קונפיגורציה חוקי:

number_of_planes = 3
number_of_lanes = 2
max_run_time = 3

plane id	start day min	start day max	mission duration	max fuel	end day	status
plane0	0	10	40	90	200	1
plane1	00	00	10	20	00	5
plane2	00	00	20	31	00	5

נציין כי שדה הסטטוס מאפשר למשתמש לפתור בעיית זמן אמת בה חלק מהמטוסים נמצאים כבר באוויר. מעשית ניתן לבצע תכנון מחדש (re-planning) ולמצוא פתרון חדש בהתאם לאילוצים חדשים שנוספו לבעיה. לדוגמה: תקלה במטוס מסוים שחייב לנחות בדחיפות. את חומרת התקלה ניתן למדל באמצעות שדה מספר 5 (max fuel) על-ידי קביעת זמן קטן יותר עבור תקלה חמורה יותר.

בנוסף, ניתן לראות כי כאשר בתחילת הבעיה המטוס נמצא באוויר הוזן המספר 00 בחלק מהפרמטרים. הדבר מציין כי במקרה זה ערך הפרמטר אינו רלוונטי לפתרון הבעיה.

3.5. אלגוריתם חיפוש ראשוני

את אלגוריתם החיפוש הראשוני ביצענו באמצעות מימוש של אלגוריתם חיפוש BFS.
ראשית, גילינו כי עבור מספר מטוסים גדול האלגוריתם לא התכנס וזאת מפני שבפועל מספר צמתי הגרף נתון על-ידי num_of_planes[11]. נציין שזהו חסם עליון כי לכל מטוס לכל היותר 11 שלבים בסדר היום שלו.
כלומר, קיבלנו הצפה של צמתים שללא הכוונה מסוימת בתהליך החיפוש לא אפשרה להגיע לצומת המטרה בזמן סביר.
לכן, שילבנו לחיפוש זה שתי היוריסטיקות בסיסיות ובכך הפכנו אותו לאלגוריתם A*.

שתי ההיוריסטיקות ששילבנו הן:

1. הכוונת החיפוש בגרף כך שכל המטוסים יבצעו את סדר היום שלהם אחד אחרי השני. כלומר, המטוס הראשון מתחיל ומסיים את הבעיה ולאחר מכן השני מתחיל ומסיים, וכן הלאה.
אלגוריתם זה התכנס בסיבוכיות זמן מצוינת עבור כל מספר מטוסים נתון.

2. הכוונת החיפוש בגרף כך שהרצת המטוסים תהיה מקבילית ככל הניתן. כלומר, הכוונה כך שכל המטוסים יהיו באותו השלב בסדר היום שלהם ככל הניתן (מכיוון שיש כמות מסלולים מוגבלת ונתיב אוויר בודד הדבר לא אפשרי תמיד).
הבעייתיות בהיוריסטיקה זו היא שכבר עבור כמות מטוסים נמוכה יחסית ועם מספר מסלולים מתאים, האלגוריתם לא מתכנס בזמן סביר. (עבור 5 מטוסים ו3 מסלולים זמן ההרצה כבר נהיה בסביבות ה- 25 דקות).

נציין כי שתי ההיוריסטיקות הנ"ל סיפקו לנו אפשרויות הרצה בסיסיות לבדיקת הנכונות הראשונית של אלגוריתם התכנון.

3.6. שילוב אילוצים זמניים

בניית האילוצים הזמניים הסתמכה על הגדרת הבעיה שתוארה בתת פרק 3.3. דבר שעזר לנו להגדיר קשר כרונולוגי זמני בין פעולות של אותו המטוס ובין פעולות של מטוסים שונים.

חשוב להבין שגם בהינתן מטוס בודד בעל מסלול פתרון יחיד המתאר פשוט את סדר היום שלו, ללא שילוב של אילוצים זמניים נקבל פתרון סופי הקובע שכל פעולות המטוס יתבצעו באותו זמן. כלומר, שכל סדר יום המטוס יתבצע בשבריר שנייה, דבר שכמובן אינו אפשרי בפועל. את הבעיה הנ״ל ניתן לפתור על-ידי שילוב אילוצים זמניים.

כמובן שיש לשלבם הן בין פעולות אותו מטוס והן בין מטוסים שונים וזאת לצורך סנכרון נכון של משאבי הבעיה. כלומר, לצורך סנכרון נכון של המסלולים והמרחב האווירי בבעיה.

נציין כי בעזרת האילוצים הזמניים ניתן לפסול כיווני פתרון לא אפשריים ובכך להקטין את גרף הבעיה ואת המצבים האפשריים במהלך חיפוש צומת המטרה בגרף. נראה זאת בתת פרק 3.6.2.

3.6.1. אילוצים זמניים בין פעולות של אותו המטוס

פעולות של אותו המטוס הוגדרו עם מרווח זמני מינימלי אפסילון בניהן, כאשר אפסילון נבחר להיות גודל קטן מאוד, 0.001 דקות. בהגדרה זו בעצם אילצנו את הפעולות השייכות לאותו מטוס להתבצע בסדר עוקב כפי שנדרש מהן. לכן, סדר הפעולות ההגיוני ישמר בפתרון הסופי של הבעיה. בפועל, מכיוון שהאפסילון הנבחר קטן מאוד פעולות שיכולות להתבצע מיד אחת לאחר השנייה אפקטיבית עבור המשתמש יתבצעו במקביל. ואילו פעולות שלוקחות יותר זמן, כדוגמת המראה או נחיתה, ישלבו אילוץ זמני נוסף המתאר זאת ובכך אפקטיבית עבור המשתמש לא יתבצעו במקביל.

אילוץ זמני בין פעולות של אותו מטוס נראה מהצורה הבאה:

$$ETO(pl1) - STO(pl1) \geq 10$$

$$SM(pl1) - ETO(pl1) \geq 0.001$$

כלומר, משך תהליך ההמראה של מטוס צריך להמשך לפחות 10 דקות וזמן התחלת משימתו באוויר חייב להיות עוקב לזמן סיום המראתו.

נזכיר כי האילוצים הזמניים הינם אילוצים של בעיית אופטימיזציה למציאת מינימום זמן כולל לפתרון הבעיה. לכן, בפועל זמן ההמראה שיבחר על-ידי כלי האופטימיזציה יהיה 10 דקות וזמן התחלת משימת המטוס באוויר יקבע להיות אפסילון זמן לאחר סיום ההמראה.

3.6.2. אילוצים זמניים בין פעולות בין מטוסים שונים

סדר הפעולות בין מטוסים שונים מגדיר את ניהול משאבי הבעיה. כלומר, את ניהול מסלולי ההמראה/נחיתה והמרחב האווירי.

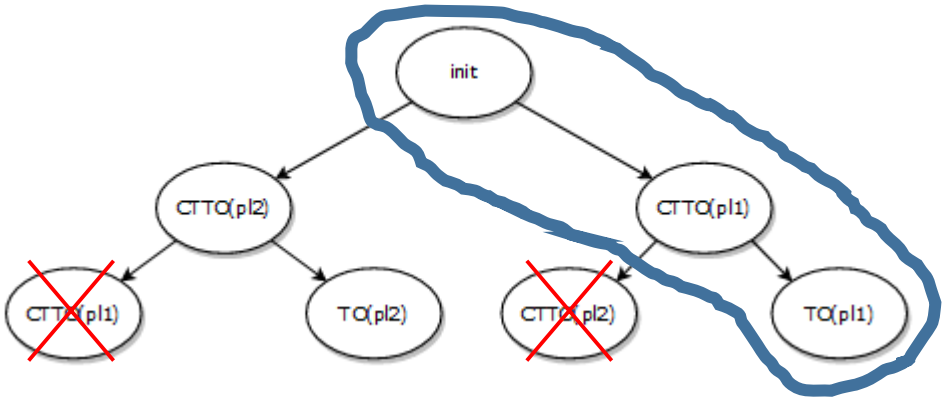
בפועל ניהול משאבי הבעיה מתבצע בעזרת האינדיקטורים והפרמטרים שהוגדרו שתת פרק 3.3.

ניזכר באיור 14 מתת פרק זה, תחת האילוץ הזמני הבא:

$$TO(pl1) - TO(pl2) > 0.001$$

כלומר, לפי אילוצי הבעיה מטוס 1 חייב להמריא לפני מטוס 2.

בפועל, לפי אילוץ זה יבחר המסלול הבא בגרף הבעיה:



איור 15 - הדגמה לבחירת מסלול בגרף הבעיה עקב האילוץ הזמני המתואר לעיל.

נציין כי המסלול המקביל אינו חוקי לפי אילוצי הבעיה ולכן לא יכול להיבחר על-ידי האלגוריתם.

כלומר, קיבלנו אילוץ זמני אשר מקטין לנו את מספר המצבים האפשריים בבעיה ובכך מקל על מציאת צומת המטרה.

3.6.3. מימוש גרף STN בעזרת כלי אופטימיזציה מתאים

לבסוף, לאחר שהאילוצים שתוארו בתת הפרקים הקודמים הוזנו לכלי האופטימיזציה שלנו, נפתרת בעיית אופטימיזציה למציאת מינימום זמן כולל לפתרון הבעיה.

הפתרון הסופי מוצג כגרף STN. גרף זה מציג את פתרון הבעיה הסופי תוך הצגת סדר כרונולוגי נכון של הפעולות שצריכות להתבצע כדי להגיע מצומת ההתחלה לצומת המטרה בגרף הבעיה.

נציין כי לגרף ה- STN תפקיד נוסף והוא להנגיש בצורה גרפית את פתרון הבעיה לאדם אשר יוכל ליישם אותו בפועל בסופו של דבר.

בפועל, גרף ה- STN נוצר מחדש עבור כל שלב בתהליך החיפוש בגרף. כעת, כפי שתואר בתת פרק 2.5.3 בסקר הספרות הגרף הדינאמי מוחזק בתור המכיל את צמתי הגרף. זאת כאשר בכל שלב של האלגוריתם שולפים את הצומת מראש התור, מסיקים את בניו בעזרת פונקציית ה successor ומכניסים אותם לתור בסדר הנקבע לפי ההיוריסטיקה הנבחרת.
נקבל למעשה בעיית אופטימיזציה הגדלה דינאמית עם גרף הבעיה. כלומר, עם הוצאת כל צומת מראש התור מוסיפים את האילוצים הנלווים לו ואת זמן הפעולה שהוא מייצג כפרמטר שנרצה למזער בבעיית האופטימיזציה הנוכחית.

לדוגמה, אם מדובר בבעיה בעלת שני מטוסים ומסלול אחד בה מטוס pl1 בדיוק סיים להמריא ותור המצבים נמצא במצב הבא:

CTTO(pl2)
TO(pl2)
. . .

נקבל כי הצומת בראש התור, CTTO(pl2), יוסיף את האילוץ:

$$CTTO(pl2) - ETO(pl1) > 0.001$$

כאשר CTTO(pl2) הוא פרמטר הנוסף לבעיית האופטימיזציה, אותו יש למזער כמובן.

לאחר מכן, בעיית האופטימיזציה תיפתר מחדש בתוספת האילוץ החדש וזמן הפעולה כפרמטר אותו יש למזער ונחזור על התהליך המוצג שוב. כלומר, נוציא את הצומת הבא מראש התור, נוסיף את אילוציו ופרמטר הזמן החדש אותו יש למזער לבעיית האופטימיזציה, נפתור אותה ונכניס את בניו לתור בסדר הנקבע לפי ההיוריסטיקה הנבחרת.
נציין כי במידה והגענו לשלב בו אין פתרון לבעיית האופטימיזציה, נחזור צעד לאחור בגרף.
כלומר, נסיר את הצומת שהוסיף את האילוץ שהוביל אותנו למבוי סתום. כמו כן נסיר גם את האילוץ וכמובן שלא נביא את בניו של הצומת, שכן הגענו למסלול פתרון לא חוקי שאין טעם להמשיך ולחפש בו את צומת המטרה.

3.7. הגדרת היוריסטיקה מותאמת לבעיה

בתכנון היוריסטיקה המותאמת לבעיה התמקדנו בצוואר הבקבוק של הבעיה הנמצא במשאב המרחב האווירי. זהו צוואר הבקבוק מפני שברגע נתון וללא תלות במספר המסלולים בבעיה רק מטוס אחד לכל היותר יכול לתפוס את המרחב האווירי. במילים אחרות, צוואר הבקבוק של הבעיה נמצא בתהליכי ההמראות/נחיתות ואם נדע לנהל את סדרם כראוי נוכל להתכנס לפתרון חוקי וטוב ובסיבוכיות זמן טובה ומספקת.

מטרת ההיוריסטיקה היא לשערך את סדר הרצת המטוסים שיוביל למציאת פתרון חוקי תוך התכנסות מהירה.

אנו למעשה מנסים לשערך את סדר ההמראות והנחיתות של כל המטוסים. חשוב לציין שמובטח שזמן נחיתה של כל מטוס גדול מזמן המראה שלו, אך לא מובטח כי זמן הנחיתה של מטוס X גדול מזמן ההמראה של מטוס Y.

לשם מימוש ההיוריסטיקה המתוארת נגדיר את שני הפרמטרים הבאים:

לשם פשטות בעת תיאור הפרמטרים נתייחס רק למקרה ההמראה, מקרה הנחיתה סימטרי.

כמו כן, לצורך ההגדרות הבאות נזכיר את עצם הגדרת הפרמטרים בקובץ הקונפיגורציה בתת פרק 3.4.

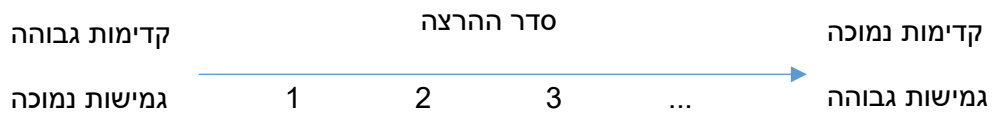
1. <u>קדימות</u>: נתינת עדיפות למטוס בעל זמן להמראה פוטנציאלי מוקדם יותר (סביר יותר שייצא לפני מטוס בעל זמן להמראה פוטנציאלי מאוחר יותר).

   זהו למעשה הפתרון שאנו שואפים אליו כי הוא יביא למינימום זמן המראת כל מטוס. מפני שכל מטוס מאלץ טווח זמנים להמראה, מצאנו שהסתמכות רק על פרמטר זה לא מובילה להתכנסות מהירה בהרבה מקרים. לכן הגדרנו גם את פרמטר הגמישות.

2. <u>גמישות</u>: נתינת עדיפות למטוס בעל גמישות נמוכה יותר. כלומר, מטוס בעל מרווח זמני המראה קטן יותר כך שיותר קשה לתמרן בקביעת זמן ההמראה שלו.

על-ידי משקול שני הפרמטרים הללו אנו קובעים את סדר הרצת המטוסים במקרה ההמראה.

בפועל מתקבל ה- trade off הבא בין הפרמטרים:



איור 16 - הדגמת ה trade off בין פרמטר הקדימות לפרמטר הגמישות.

כלומר, בהינתן מטוס בעל קדימות וגמישות גבוהות נצטרך לבצע החלטה בהתאם לשאר נתוני הבעיה היכן למקמו בסדר ההמראה.

נעשה זאת גם במקרה הנחיתה ונקבל את סדר הרצת המטוסים באותו אופן. (במקרה הנחיתה פרמטר הגמישות שקול לכמה דלק יש למטוס כי הוא מציין כמה אפשר לתמרן עם זמן הנחיתה שלו בהתאם לדלק).

כך אנו מריצים את האלגוריתם בתנאי התחלה שונים ומנסים למצוא את תנאי ההתחלה שיגרום להתכנסות מהירה של האלגוריתם. בפועל תנאי ההתחלה שאנו מנסים לשערך הוא סדר המראות ונחיתות כלל המטוסים, כך שיביא את הבעיה להכנסות מהירה.

3.8. Learning Heuristic – למידת היוריסטיקה חכמה בעזרת כלים של מערכת לומדת

בפועל, למצוא פונקציה שבהינתן קובץ קלט מסוים תדע להוציא כפלט את סדר המראות ונחיתות המטוסים זו משימה לא קלה. ולכן, עבור היוריסטיקה זו מימשנו על-ידי רשת נוירונים מערכת לומדת שתפקידה הוא ללמוד את פונקציית סדר המראות ונחיתות המטוסים.
הלמידה הינה למידה מודרכת, כלומר יצרנו סט אימון המכיל לכל דוגמה תיוג של סדר המראות ונחיתות נכון. נציין שהתיוגים נקבעו על סמך מדדי הקדימות והגמישות שהוגדרו בתת הפרק הקודם.
בנוסף, נדגיש כי ישנה אפשרות שבזמן תהליך החיפוש האלגוריתם יבחר בדרך פתרון הכוללת סדר הרצה שונה ממה שקבעה רשת הנוירונים המאומנת. תפקיד הרשת הוא לתת כיוון לאלגוריתם החיפוש ולא לקבוע את דרך החיפוש. וזאת כדי שיוכל להתכנס לפתרון חוקי בזמן קצר.

3.8.1. הגדרת קלט רשת הנוירונים

נגדיר את מטריצת המאפיינים המהווה את הקלט לרשת הנוירונים. בהתאם לנאמר בתת פרק 3.4 מטריצה זו נגזרת מקובץ הקלט שמזין המשתמש לאלגוריתם.
כעת, מפני שאנו צריכים להתייחס להמראות ונחיתות בנפרד נציג את מאפייני הקלט באופן הבא:

חסם מקסימלי לביצוע	מדד גמישות	מדד קדימות
...	...	...

כאשר:

- מדד הקדימות:
  - במקרה ההמראה זהו הערך start day min.
  - במקרה הנחיתה זהו הערך mission duration.
- מדד הגמישות:
  - במקרה ההמראה זהו הערך start day max – start day min.
  - במקרה הנחיתה זהו הערך max fuel – mission duration.
- חסם מקסימלי לביצוע:
  - במקרה ההמראה זהו הערך end day.
  - במקרה הנחיתה זהו הערך max fuel + 20. (זהו החסם מפני שהערך max fuel מייצג את הזמן המקסימלי בו המטוס חייב להתחיל בתהליכי הנחיתה ולאחריו פינוי המסלול. שני תהליכים שלפי הנחות הפרויקט שהוצגו בתת פרק 3.3 אורכים 20 דקות.

בנוסף, נשרשר לסוף המטריצה שורה נוספת המייצגת את המאפיינים של מספר המסלולים בבעיה.
נדגיש כי ממד המטריצה הוא: 3 X (1 + מספר ההמראות + מספר הנחיתות)

3.8.2. מבנה רשת הנוירונים

הרשת הלומדת מורכבת 4 שכבות ליניאריות (fully connected):

1. שכבה מקדימה שתפקידה להמיר את ממד מאפיין מספר המסלולים ממספר טבעי לווקטור מממד 3 (כדי שנוכל לשרשר אותו למטריצת המאפיינים).
   self.num_of_lanes_transform = nn.Linear(1, features_in, bias=True)
2. self.linear_1 = nn.Linear(features_in, hidden, bias=True)
3. self.linear_2 = nn.Linear(hidden, hidden, bias=True)
4. self.linear_3 = nn.Linear(hidden, 1, bias=True)

כאשר:

- features_in = 3.
- hidden = 10.

שכבות 1,2,3 אלו הן השכבות הלומדות העיקריות, כאשר בין כל שכבה לשכבה יש מעבר בפונקציית softmax המשמשת כשכבת אקטיבציה.

3.8.3. שיטת האימון

לאימון רשת הנוירונים בחרנו בשיטת ה- cross validation. כפי שכתבתנו בפרק סקר הספרות, שיטה זו הינה שיטה לאימון רשת בהינתן סט אימון קטן יחסית.

באימון הרשת עשינו שימוש ב 25 דוגמאות שחולקו ל- 5 קבוצות אימון (בגודל 5 דוגמאות כל אחת). כמובן שעשינו שימוש בשיטה זו במטרה למקסם את ביצועי הרשת תחת סט האימון הנתון.

כמו כן, הרשת אומנה למשך 30,000 איטרציות ותחת הבחירות הבאות:

1. שימוש במדד שגיאה ריבועית.
2. אימון בשיטת SGD עם גודל צעד 0.0025.

לשם השלמות נציין כי לצורך האימון הוספנו שדה נוסף לקובץ הקלט שהוגדר בתת פרק 3.4 והוא שדה התיוג של הדוגמה.
מדובר סך-הכול במספור ידני של סדר ההמראות נחיתות בדוגמה וזאת בהתאם למאפיינים שהוגדרו בה.

להלן קובץ קלט כזה לדוגמה:

number_of_planes = 3
number_of_lanes = 2
max_run_time = 3

plane id	start day min	start day max	mission duration	max fuel	end day	status	label
plane0	0	10	40	90	200	1	3 4
plane1	00	00	10	20	00	5	1
plane2	00	00	20	31	00	5	2

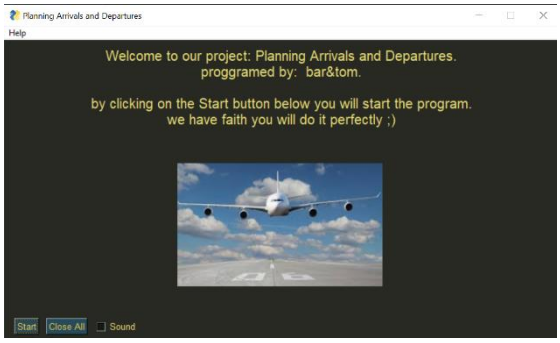
ניתן לראות בסוף את מנגנון התיוג שהוסף. כאשר, עבור מטוס הנמצא בתחילת הבעיה על הקרקע שני תיוגים, להמראה ולנחיתה בהתאם.

3.9. GUI- Graphical User Interface

מטרת ה GUI היא לספק ממשק נוח עבור המשתמש דרכו הוא יוכל להזין קלט לאלגוריתם בצורה נוחה וידידותית. למעשה, תפקיד הממשק הוא להוות תחליף לשיטת הזנת קלט דרך שורות קוד וקבצי קונפיגורציה שנועדו למטרות פיתוח ו- debug אשר פחות נוחה באופן טבעי עבור המשתמש.

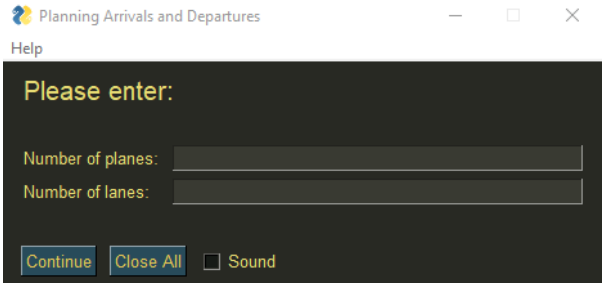
לממשק שלנו ארבעה מסכים עיקריים:

1. מסך פתיחה אשר מכיל הודעת פתיחה, כפתור השולט על שמע במהלך ההרצה ולשונית about אשר מספקת מידע על הפרויקט, המנחה ועל זכויות היוצרים.
נציין כי פונקציית השליטה על השמע ולשונית ה- about יופיעו מאתה בכל מסכי הממשק.



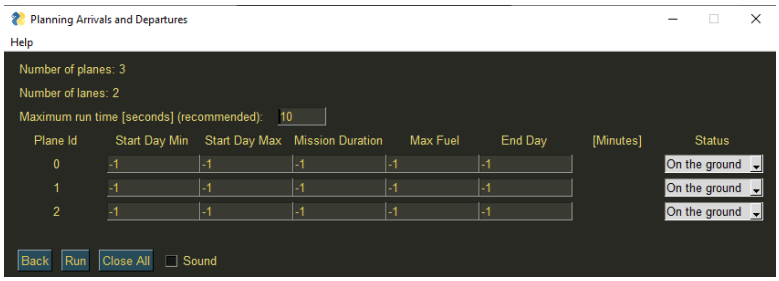
איור 17 - מסך פתיחה של ממשק ה- GUI.

2. מסך ההגדרה הראשוני של הבעיה בו המשתמש יגדיר את כמות המסלולים, כמות המטוסים וזמן ההרצה הכולל שהוא מאפשר לזמן החיפוש (זמן הרצה דיפולטי הינו 10 שניות).



איור 18 - מסך ההגדרה הראשוני של ממשק ה- GUI.

3. מסך ההגדרה השני של הבעיה בו המשתמש ישלים את הגדרת הבעיה תוך הזנת הנתונים המתאימים לכלל המטוסים בבעיה. במסך זה לשונית נוספת המספקת הוראות כיצד יש להשלים את הגדרת הבעיה. בנוסף נציין כי כל הפרמטרים במסך זה מוגדרים גם בתת פרק 3.4.



איור 19 - מסך ההגדרה השני של ממשק ה- GUI.

4. מסך תוצאת האלגוריתם. נציג מסך זה בפרק התוצאות.

נציין כי בכל שלב שירצה יוכל המשתמש לעבור בין המסכים השונים על מנת לשנות ולהחליף נתונים במידת הצורך.

4. תוצאות ומסקנות

על מנת להעריך את תוצאות האלגוריתם יצרנו 25 קבצי הרצה בעלי פתרון.
בפרק זה נציג שתי דוגמאות בהן האלגוריתם הצליח למצוא פתרון בהכוונת רשת הנוירונים ודוגמה אחת בה לא.
כמובן שבדוגמה בה רשת הנוירונים נכשלה נציג את סדר ההרצה שכן מוביל לפתרון כמו גם את הפתרון עצמו שהיה אמור להתקבל.
כמו כן, בפרק הנספחים נציג שתי דוגמאות לקובץ log מלא כפלט מהאלגוריתם. מטרת קובץ זה היא לצורכי debug ופיתוחים עתידיים.

4.1. דוגמאות בהן רשת הנוירונים הצליחה

נזכיר כי קובץ קלט חוקי נראה מהצורה הבאה:

number_of_planes = 3
number_of_lanes = 2
max_run_time = 3

plane id	start day min	start day max	mission duration	max fuel	end day	status	label
plane0	0	10	40	90	200	1	3 4
plane1	00	00	10	20	00	5	1
plane2	00	00	20	31	00	5	2

1. <u>קובץ קלט:</u>

```
number_of_planes = 3
number_of_lanes = 4
max_run_time = 3
plane0 0 1 200 300 320   1   4 6
plane1 0 1 30  40  75    1   1 2
plane2 0 1 120 150 215   1   3 5
```

איור 20 - דוגמה לקובץ קלט (1).

<u>פתרון האלגוריתם:</u>

Planning Arrivals and Departures

solution :

Operation	Plane Id	Lane Id	Start Time [Relatively minute]	Duration [minutes]
Aligning	0	0	0	10
Aligning	1	1	0	10
Aligning	2	3	0	10
Take Off	1	1	10	10
Mission	1		20	30
Landing	1	2	50	10
Evacuation	1	2	60	10
Take Off	2	3	60	10
Take Off	0	0	70	10
Mission	2		70	120
Mission	0		80	200
Landing	2	1	190	10
Evacuation	2	1	200	10
Landing	0	2	280	10
Evacuation	0	2	290	10

Re-Planning

איור 21 - דוגמה לפתרון האלגוריתם (1).

גרף STN מלא:

```
t_sctto_1 :  0.0        :        parents:  []                                  :        childs:  ['t_ectto_1']
t_sctto_2 :  0.0        :        parents:  []                                  :        childs:  ['t_ectto_2']
t_sctto_0 :  0.0        :        parents:  []                                  :        childs:  ['t_ectto_0']
t_ectto_1 :  10.0       :        parents:  ['t_sctto_1']                       :        childs:  ['t_sto_1']
t_ectto_2 :  10.0       :        parents:  ['t_sctto_2']                       :        childs:  ['t_sto_2']
t_ectto_0 :  10.0       :        parents:  ['t_sctto_0']                       :        childs:  ['t_sto_0']
t_sto_1   :  10.001     :        parents:  ['t_ectto_1']                       :        childs:  ['t_eto_1']
t_eto_1   :  20.001     :        parents:  ['t_sto_1']                         :        childs:  ['t_sl_2', 't_sm_1', 't_sl_1']
t_sm_1    :  20.002     :        parents:  ['t_eto_1']                         :        childs:  ['t_em_1']
t_em_1    :  50.002     :        parents:  ['t_sm_1']                          :        childs:  ['t_sl_1']
t_sl_1    :  50.003     :        parents:  ['t_eto_1', 't_em_1']               :        childs:  ['t_el_1']
t_el_1    :  60.003     :        parents:  ['t_sl_1']                          :        childs:  ['t_st_1', 't_sto_2']
t_st_1    :  60.004     :        parents:  ['t_el_1']                          :        childs:  ['t_et_1']
t_sto_2   :  60.004     :        parents:  ['t_ectto_2', 't_el_1']             :        childs:  ['t_eto_2']
t_et_1    :  70.004     :        parents:  ['t_st_1']                          :        childs:  ['t_sl_0']
t_eto_2   :  70.004     :        parents:  ['t_sto_2']                         :        childs:  ['t_sm_2', 't_sto_0']
t_sto_0   :  70.005     :        parents:  ['t_ectto_0', 't_eto_2']            :        childs:  ['t_eto_0']
t_sm_2    :  70.005     :        parents:  ['t_eto_2']                         :        childs:  ['t_em_2']
t_eto_0   :  80.005     :        parents:  ['t_sto_0']                         :        childs:  ['t_sl_2', 't_sm_0']
t_sm_0    :  80.006     :        parents:  ['t_eto_0']                         :        childs:  ['t_em_0']
t_em_2    :  190.005    :        parents:  ['t_sm_2']                          :        childs:  ['t_sl_2']
t_sl_2    :  190.006    :        parents:  ['t_em_2', 't_eto_0', 't_eto_1']    :        childs:  ['t_el_2']
t_el_2    :  200.006    :        parents:  ['t_sl_2']                          :        childs:  ['t_sl_0', 't_st_2']
t_st_2    :  200.007    :        parents:  ['t_el_2']                          :        childs:  ['t_et_2']
t_et_2    :  210.007    :        parents:  ['t_st_2']                          :        childs:  []
t_em_0    :  280.006    :        parents:  ['t_sm_0']                          :        childs:  ['t_sl_0']
t_sl_0    :  280.007    :        parents:  ['t_el_2', 't_et_1', 't_em_0']      :        childs:  ['t_el_0']
t_el_0    :  290.007    :        parents:  ['t_sl_0']                          :        childs:  ['t_st_0']
t_st_0    :  290.008    :        parents:  ['t_el_0']                          :        childs:  ['t_et_0']
t_et_0    :  300.008    :        parents:  ['t_st_0']                          :        childs:  []
```

איור 22 - דוגמה לפלט האלגוריתם כגרף STN מלא (1).

ניתוח תוצאת האלגוריתם:

באיור 21 ניתן לראות את פתרון האלגוריתם כאשר הוא מפורט וכולל זמני התחלת כל פעולה (זמן יחסי בדקות), משך זמן כל פעולה (בדקות) ומסלול המשויך לכל פעולה במידת הצורך.
ניתן לראות כי סדר המראות/נחיתות המטוסים שנקבע על ידי האלגוריתם לאחר הכוונת רשת הנוירונים זהה לסדר האמיתי שנקבע בתיוג הדוגמה. כלומר, ההיוריסטיקה שנלמדה אכן הצליחה במקרה זה.
כמו כן, בנוסף לפתרון האלגוריתם צירפנו את פתרון גרף ה- STN המלא שהתקבל בו מתואר מסלול הפתרון מצומת ההתחלה לצומת המטרה, תוך התייחסות עבור כל צומת מי הוריו. כלומר, התייחסות עבור כל צומת מה הפעולות שחייבות להתבצע לפניו.
למעשה כל צומת בגרף הפתרון הוא פעולה שיש לבצע בזמן מסוים כך שבסופו של דבר מסלול הפתרון הוא סדר פעולות שיש לבצע בזמנים מסוימים.
כמו כן, לכל צומת במסלול הפתרון מתואר גם מי ילדיו. כלומר, מה הפעולות שיכולות להתבצע אחריו אשר אחת מהן באמת נבחרה, המשיכה את מסלול הפתרון ואת גרף ה- STN.

2. <u>קובץ קלט:</u>

```
number_of_planes = 4
number_of_lanes = 2
max_run_time = 3
plane0 0 100 40 50 200   1   5 8
plane1 0 10 20 30 90     1   1 4
plane2 10 33 29 50 140   1   3 6
plane3 0 22 30 110 130   1   2 7
```

איור 23 - דוגמה לקובץ קלט (2).

<u>פתרון האלגוריתם:</u>

Planning Arrivals and Departures

solution :

Operation	Plane Id	Lane Id	Start Time [Relatively minute]	Duration [minutes]
Aligning	1	1	0	10
Aligning	3	0	0	10
Take Off	1	1	10	10
Mission	1		20	20
Aligning	2	1	20	10
Take Off	3	0	20	10
Take Off	2	1	30	10
Mission	3		30	30
Landing	1	0	40	10
Mission	2		40	29
Evacuation	1	0	50	10
Aligning	0	0	60	10
Landing	2	1	69	10
Take Off	0	0	79	10
Evacuation	2	1	79	10
Mission	0		89	40
Landing	3	1	89	10
Evacuation	3	1	99	10
Landing	0	0	129	10
Evacuation	0	0	139	10

Re-Planning

איור 24 - דוגמה לפתרון האלגוריתם (2).

גרף STN מלא:

```
t_sctto_1 :  0.0        :   parents:  []                                  :   childs:  ['t_ectto_1']
t_sctto_3 :  0.0        :   parents:  []                                  :   childs:  ['t_ectto_3']
t_ectto_1 :  10.0       :   parents:  ['t_sctto_1']                       :   childs:  ['t_sto_1']
t_ectto_3 :  10.0       :   parents:  ['t_sctto_3']                       :   childs:  ['t_sto_3']
t_sto_1   :  10.001     :   parents:  ['t_ectto_1']                       :   childs:  ['t_eto_1']
t_eto_1   :  20.001     :   parents:  ['t_sto_1']                         :   childs:  ['t_sm_1', 't_sto_3', 't_sctto_2']
t_sto_3   :  20.002     :   parents:  ['t_eto_1', 't_ectto_3']            :   childs:  ['t_eto_3']
t_sctto_2 :  20.002     :   parents:  ['t_eto_1']                         :   childs:  ['t_ectto_2']
t_sm_1    :  20.002     :   parents:  ['t_eto_1']                         :   childs:  ['t_em_1']
t_eto_3   :  30.002     :   parents:  ['t_sto_3']                         :   childs:  ['t_sm_3', 't_sto_2', 't_sl_1']
t_ectto_2 :  30.002     :   parents:  ['t_sctto_2']                       :   childs:  ['t_sto_2']
t_sto_2   :  30.003     :   parents:  ['t_eto_3', 't_ectto_2']            :   childs:  ['t_eto_2']
t_sm_3    :  30.003     :   parents:  ['t_eto_3']                         :   childs:  ['t_em_3']
t_em_1    :  40.002     :   parents:  ['t_sm_1']                          :   childs:  ['t_sl_1']
t_eto_2   :  40.003     :   parents:  ['t_sto_2']                         :   childs:  ['t_sl_1', 't_sm_2', 't_sl_2']
t_sl_1    :  40.004     :   parents:  ['t_eto_2', 't_em_1', 't_eto_3']    :   childs:  ['t_el_1']
t_sm_2    :  40.004     :   parents:  ['t_eto_2']                         :   childs:  ['t_em_2']
t_el_1    :  50.004     :   parents:  ['t_sl_1']                          :   childs:  ['t_sl_2', 't_st_1']
t_st_1    :  50.005     :   parents:  ['t_el_1']                          :   childs:  ['t_et_1']
t_em_3    :  60.003     :   parents:  ['t_sm_3']                          :   childs:  ['t_sl_3']
t_et_1    :  60.005     :   parents:  ['t_st_1']                          :   childs:  ['t_sctto_0']
t_sctto_0 :  60.006     :   parents:  ['t_et_1']                          :   childs:  ['t_ectto_0']
t_em_2    :  69.004     :   parents:  ['t_sm_2']                          :   childs:  ['t_sl_2']
t_sl_2    :  69.005     :   parents:  ['t_el_1', 't_eto_2', 't_em_2']     :   childs:  ['t_el_2']
t_ectto_0 :  70.006     :   parents:  ['t_sctto_0']                       :   childs:  ['t_sto_0']
t_el_2    :  79.005     :   parents:  ['t_sl_2']                          :   childs:  ['t_st_2', 't_sto_0']
t_st_2    :  79.006     :   parents:  ['t_el_2']                          :   childs:  ['t_et_2']
t_sto_0   :  79.006     :   parents:  ['t_ectto_0', 't_el_2']             :   childs:  ['t_eto_0']
t_et_2    :  89.006     :   parents:  ['t_st_2']                          :   childs:  ['t_sl_3']
t_eto_0   :  89.006     :   parents:  ['t_sto_0']                         :   childs:  ['t_sl_3', 't_sl_0', 't_sm_0']
t_sl_3    :  89.007     :   parents:  ['t_em_3', 't_eto_0', 't_et_2']     :   childs:  ['t_el_3']
t_sm_0    :  89.007     :   parents:  ['t_eto_0']                         :   childs:  ['t_em_0']
t_el_3    :  99.007     :   parents:  ['t_sl_3']                          :   childs:  ['t_sl_0', 't_st_3']
t_st_3    :  99.008     :   parents:  ['t_el_3']                          :   childs:  ['t_et_3']
t_et_3    :  109.008    :   parents:  ['t_st_3']                          :   childs:  []
t_em_0    :  129.007    :   parents:  ['t_sm_0']                          :   childs:  ['t_sl_0']
t_sl_0    :  129.008    :   parents:  ['t_el_3', 't_eto_0', 't_em_0']     :   childs:  ['t_el_0']
t_el_0    :  139.008    :   parents:  ['t_sl_0']                          :   childs:  ['t_st_0']
t_st_0    :  139.009    :   parents:  ['t_el_0']                          :   childs:  ['t_et_0']
t_et_0    :  149.009    :   parents:  ['t_st_0']                          :   childs:  []
```

איור 25 - דוגמה לפלט האלגוריתם כגרף STN מלא (2).

ניתוח תוצאת האלגוריתם:

סדר המראות/נחיתות המטוסים שנקבע על ידי האלגוריתם לאחר הכוונת רשת הנוירונים:

1. מטוס 1 ממריא.
2. מטוס 3 ממריא.
3. מטוס 2 ממריא.
4. מטוס 1 נוחת.
5. מטוס 2 נוחת.
6. מטוס 0 ממריא.
7. מטוס 3 נוחת.
8. מטוס 0 נוחת.

באיור 24 ניתן לראות את פתרון האלגוריתם כאשר הוא מפורט וכולל זמני התחלת כל פעולה (זמן יחסי בדקות), משך זמן כל פעולה (בדקות) ומסלול המשויך לכל פעולה במידת הצורך.
ניתן לראות שסדר זה תואם לסדר האמיתי כפי שנקבע בתיוג הדוגמה פרט לשלבים 5,6 שהתחלפו. בסופו של דבר האלגוריתם הצליח להתכנס לפתרון חוקי ובסיבוכיות זמן טובה.
כלומר, בסופו של דבר ההחלפה הנ"ל לא הייתה משמעותית וההיוריסטיקה שנלמדה אכן הצליחה במקרה זה.
כמו כן, בנוסף לפתרון האלגוריתם צירפנו את פתרון גרף ה- STN המלא שהתקבל בו מתואר מסלול הפתרון מצומת ההתחלה לצומת המטרה, תוך התייחסות עבור כל צומת מי הוריו. כלומר, התייחסות עבור כל צומת מה הפעולות שחייבות להתבצע לפניו.
למעשה כל צומת בגרף הפתרון הוא פעולה שיש לבצע בזמן מסוים כך שבסופו של דבר מסלול הפתרון הוא סדר פעולות שיש לבצע בזמנים מסוימים.
כמו כן, לכל צומת במסלול הפתרון מתואר גם מי ילדיו. כלומר, מה הפעולות שיכולות להתבצע אחריו אשר אחת מהן באמת נבחרה, המשיכה את מסלול הפתרון ואת גרף ה- STN.

4.2. דוגמה בה רשת הנוירונים נכשלה

1. <u>קובץ קלט:</u>

```
number_of_planes = 4
number_of_lanes = 2
max_run_time = 3
plane0 0 10 41  84  130   1   1 7
plane1 0 10 42  60  100   1   2 5
plane2 0 40 43  73  120   1   3 6
plane3 0 70 144 150 250   1   4 8
```

איור 26 - דוגמה לקובץ קלט (3).

2. <u>סדר ההרצה:</u>

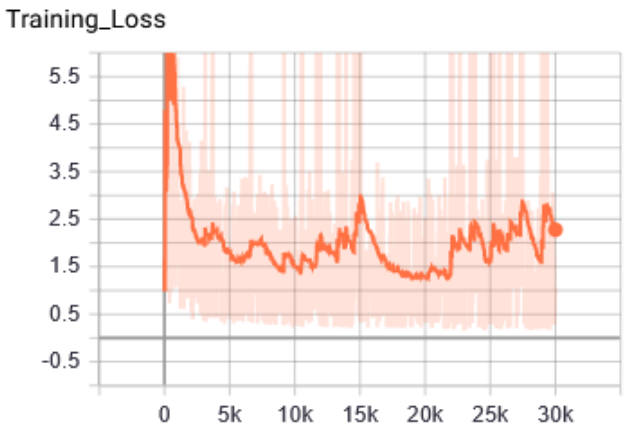
בהכוונת רשת הנוירונים סדר ההרצה שהתקבל הוא:

1. מטוס 1 ממריא.
2. מטוס 0 ממריא.
3. מטוס 2 ממריא.
4. מטוס 1 נוחת.
5. מטוס 2 נוחת.
6. מטוס 3 ממריא.
7. מטוס 0 נוחת.
8. מטוס 3 נוחת.

ניתן לראות כי סדר זה אינו תואם את הסדר שציפינו לקבל לפי התיוג בקובץ הקלט וכי בניגוד למקרה בתת הפרק הקודם הפעם השגיאה ביחס לתיוג בקובץ הקלט משמעותית יותר.
לכן האלגוריתם לא הצליח להתכנס ולמצוא פתרון בזמן שהוקצב לו.

לשם שלמות נוכיח כי בהינתן הכוונה לפי סדר ההרצה הנכון (המוצג בשדה התיוג בקובץ הקלט) קיים פתרון. הפתרון המלא מוצג בפרק הנספחים.

4.3. תוצאות אימון רשת הנוירונים



איור 27 - שגיאת האימון.

איור 28 - שגיאת הבוחן.

ניתן לראות כי ישנו תהליך למידה וכי הן שגיאת האימון והן שגיאת הבוחן קטנות.
נציין שהצגנו את גרפי השגיאות הנ"ל בתצוגה "חלקה" ולא "מורעשת" כדי שיהיה קל להבחין בתהליך הלימוד בצורה כללית.
נשים לב כי הן שגיאת האימון והן שגיאת הבוחן הגיעו בממוצע לערך 2. לפי מדד השגיאה הריבועית שהוגדר ערך זה מציין כי הוחלף סדר של 2 פעולות צמודות. (כל פעולה נמצאת במרחק 1 ממיקומה האמיתי לפי התיוג – סך-הכול מתקבלת השגיאה 2).
לכן, בסך-הכול נוכל להגיד שתהליך הלימוד שהתקבל הוא טוב, אם כי בהחלט ניתן לשפרו בהמשך.

4.4. סיכום התוצאות

נסכם ונאמר כי בסופו של דבר בכל 25 קבצי הדוגמאות שהרצנו ותחת היוריסטיקה אשר מכווינה את תהליך החיפוש נכונה התקבלו מסלולי פתרון חוקיים ותת-אופטימליים. כלומר, אופטימליים ביחס לכל מסלול שנבחר.
נדגיש כי בהינתן זמן ריצה אינסופי יתקבלו מסלולי פתרון חוקיים ותת-אופטימליים גם ללא הכוונת ההיוריסטיקה.

בכל הנוגע לסיבוכיות זמן מציאת מסלול הפתרון הדבר מתכתב עם ביצועי רשת הנוירונים.
בשיטת אימון ה- cross validation קיבלנו כי ההיוריסטיקה שנלמדה אכן הצליחה ב 15 מתוך 25 הדוגמאות. זאת כאשר הצלחה הוגדרה במקרה בו האלגוריתם הצליח להתכנס לפתרון בזמן שהוקצב לו (עד 3 שניות). מדובר ב- 60 אחוזי הצלחה.

5. רשימת מקורות

1. https://cw.fel.cvut.cz/old/_media/courses/a4m33pah/01-intro.pdf
2. https://medium.com/omarelgabrys-blog/path-finding-algorithms-f65a8902eb40
3. https://medium.com/@nicholas.w.swift/easy-a-star-pathfinding-7e6689c7f7b2
4. https://medium.com/@geekrodion/linear-programming-introduction-e0547f3db30d
5. https://www.youtube.com/watch?v=aircAruvnKk
6. https://cs231n.github.io/convolutional-networks/
7. https://towardsdatascience.com/how-to-do-deep-learning-on-graphs-with-graph-convolutional-networks-7d2250723780
8. https://en.wikipedia.org

6. נספחים

6.1. דוגמאות לקובץ log מלא כפלט מהאלגוריתם

קובץ ה- log מכיל:

1. זמן כולל לפתרון הבעיה. (הזמן בו המטוס האחרון סיים לנחות ולפנות את מסלול הנחיתה).
2. סדר הרצת המראות ונחיתות המטוסים כפי שהתקבל על-ידי האלגוריתם.
3. מסלולי ההמראה והנחיתה של המטוסים.
4. גרף STN מלא של פתרון הבעיה.
5. האילוצים הזמניים בבעיה כפי שנקבע במסלול הפתרון הנבחר.
6. זמן ההרצה של האלגוריתם עד למציאת הפתרון.

דוגמה לקובץ log עבור דוגמה 1 בתת פרק 4.1:

```
==========================================================
solution found :)
max time: 300.0079999999999
exec order (as we target):
['1_to', '1_l', '2_to', '0_to', '2_l', '0_l']
exec order (by the algorithm):
['1_to', '1_l', '2_to', '0_to', '2_l', '0_l']
==========================================================
==========================================================
final solution:
==========================================================
plane 0 : [take off lane id, landing lane id] = [0, 2]
plane 1 : [take off lane id, landing lane id] = [1, 2]
plane 2 : [take off lane id, landing lane id] = [3, 1]
==========================================================
t_sctto_1 : 0.0 : parents: [] : childs: ['t_ectto_1']
t_sctto_2 : 0.0 : parents: [] : childs: ['t_ectto_2']
t_sctto_0 : 0.0 : parents: [] : childs: ['t_ectto_0']
t_ectto_1 : 10.0 : parents: ['t_sctto_1'] : childs: ['t_sto_1']
```

t_ectto_2 : 10.0 : parents: ['t_sctto_2'] : childs: ['t_sto_2']

t_ectto_0 : 10.0 : parents: ['t_sctto_0'] : childs: ['t_sto_0']

t_sto_1 : 10.001 : parents: ['t_ectto_1'] : childs: ['t_eto_1']

t_eto_1 : 20.001 : parents: ['t_sto_1'] : childs: ['t_sm_1', 't_sl_2', 't_sl_1']

t_sm_1 : 20.002 : parents: ['t_eto_1'] : childs: ['t_em_1']

t_em_1 : 50.002 : parents: ['t_sm_1'] : childs: ['t_sl_1']

t_sl_1 : 50.003 : parents: ['t_em_1', 't_eto_1'] : childs: ['t_el_1']

t_el_1 : 60.003 : parents: ['t_sl_1'] : childs: ['t_sto_2', 't_st_1']

t_st_1 : 60.004 : parents: ['t_el_1'] : childs: ['t_et_1']

t_sto_2 : 60.004 : parents: ['t_ectto_2', 't_el_1'] : childs: ['t_eto_2']

t_et_1 : 70.004 : parents: ['t_st_1'] : childs: ['t_sl_0']

t_eto_2 : 70.004 : parents: ['t_sto_2'] : childs: ['t_sm_2', 't_sto_0']

t_sto_0 : 70.005 : parents: ['t_ectto_0', 't_eto_2'] : childs: ['t_eto_0']

t_sm_2 : 70.005 : parents: ['t_eto_2'] : childs: ['t_em_2']

t_eto_0 : 80.005 : parents: ['t_sto_0'] : childs: ['t_sl_2', 't_sm_0']

t_sm_0 : 80.006 : parents: ['t_eto_0'] : childs: ['t_em_0']

t_em_2 : 190.005 : parents: ['t_sm_2'] : childs: ['t_sl_2']

t_sl_2 : 190.006 : parents: ['t_eto_0', 't_eto_1', 't_em_2'] : childs: ['t_el_2']

t_el_2 : 200.006 : parents: ['t_sl_2'] : childs: ['t_sl_0', 't_st_2']

t_st_2 : 200.007 : parents: ['t_el_2'] : childs: ['t_et_2']

t_et_2 : 210.007 : parents: ['t_st_2'] : childs: []

t_em_0 : 280.006 : parents: ['t_sm_0'] : childs: ['t_sl_0']

t_sl_0 : 280.007 : parents: ['t_el_2', 't_et_1', 't_em_0'] : childs: ['t_el_0']

t_el_0 : 290.007 : parents: ['t_sl_0'] : childs: ['t_st_0']

t_st_0 : 290.008 : parents: ['t_el_0'] : childs: ['t_et_0']

t_et_0 : 300.008 : parents: ['t_st_0'] : childs: []

==========

t_eto_2 - t_sto_2 >= 10

t_sm_1 - t_eto_1 >= 0.001

t_el_2 - t_sl_2 <= 10.01

t_et_2 - t_st_2 >= 10

t_sm_0 - t_eto_0 <= 0.01

t_et_0 - t_st_0 >= 10

t_et_2 <= 215

t_em_1 - t_sm_1 >= 30

t_st_2 - t_el_2 <= 0.01

t_sctto_2 <= 1

t_eto_0 - t_sto_0 <= 10.01

t_et_2 - t_st_2 <= 10.01

t_ectto_0 - t_sctto_0 >= 10

t_et_0 <= 320

t_ectto_1 - t_sctto_1 <= 10.01

t_et_0 - t_st_0 <= 10.01

t_el_1 - t_sl_1 <= 10.01

t_sl_2 - t_eto_0 >= 0.001

t_sl_0 - t_el_2 >= 0.001

t_et_1 - t_st_1 >= 10

t_sto_0 - t_ectto_0 >= 0.001

t_eto_2 - t_sto_2 <= 10.01

t_sl_1 - t_sm_1 <= 40

t_el_0 - t_sl_0 >= 10

t_ectto_1 - t_sctto_1 >= 10

t_sl_1 - t_em_1 >= 0.001

t_ectto_2 - t_sctto_2 <= 10.01

t_sctto_0 <= 1

t_em_2 - t_sm_2 >= 120

t_sm_2 - t_eto_2 <= 0.01

t_et_1 - t_st_1 <= 10.01

t_sctto_2 >= 0

t_et_1 <= 75

t_ectto_0 - t_sctto_0 <= 10.01

t_sl_0 - t_et_1 >= 0.001

t_sctto_1 >= 0

t_sto_2 - t_ectto_2 >= 0.001

t_sl_2 - t_eto_1 >= 0.001

t_sto_2 - t_el_1 >= 0.001

t_eto_0 - t_sto_0 >= 10

t_sm_2 - t_eto_2 >= 0.001

t_el_0 - t_sl_0 <= 10.01

t_sctto_1 <= 1

t_st_1 - t_el_1 <= 0.01

t_eto_1 - t_sto_1 <= 10.01

t_st_2 - t_el_2 >= 0.001

t_sl_1 - t_eto_1 >= 0.001

t_sto_0 - t_eto_2 >= 0.001

t_st_1 - t_el_1 >= 0.001

t_em_0 - t_sm_0 >= 200

t_el_2 - t_sl_2 >= 10

t_sm_0 - t_eto_0 >= 0.001

t_sl_0 - t_sm_0 <= 300

t_st_0 - t_el_0 >= 0.001

t_sl_2 - t_em_2 >= 0.001

t_st_0 - t_el_0 <= 0.01

t_ectto_2 - t_sctto_2 >= 10

t_sl_0 - t_em_0 >= 0.001

t_sto_1 - t_ectto_1 >= 0.001

t_sl_2 - t_sm_2 <= 150

t_sctto_0 >= 0

t_sm_1 - t_eto_1 <= 0.01

t_eto_1 - t_sto_1 >= 10

t_el_1 - t_sl_1 >= 10

==========

==========
==========

total run time: 0.030989699999999676

==========

דוגמה לקובץ log עבור דוגמה 1 בתת פרק 4.2:

```
==========================================================
solution found :)
max time: 214.007
exec order (as we target):
['0_to', '1_to', '2_to', '3_to', '1_l', '2_l', '0_l', '3_l']
exec order (by the algorithm):
['0_to', '1_to', '2_to', '3_to', '1_l', '2_l', '0_l', '3_l']
==========================================================
==========================================================
final solution:
==========================================================
plane 0 : [take off lane id, landing lane id] = [1, 1]
plane 1 : [take off lane id, landing lane id] = [0, 1]
plane 2 : [take off lane id, landing lane id] = [1, 0]
plane 3 : [take off lane id, landing lane id] = [0, 0]
==========================================================
t_sctto_0 : 0.0 : parents: [] : childs: ['t_ectto_0']
t_sctto_1 : 0.0 : parents: [] : childs: ['t_ectto_1']
t_ectto_0 : 10.0 : parents: ['t_sctto_0'] : childs: ['t_sto_0']
t_ectto_1 : 10.0 : parents: ['t_sctto_1'] : childs: ['t_sto_1']
t_sto_0 : 10.001 : parents: ['t_ectto_0'] : childs: ['t_eto_0']
t_eto_0 : 20.001 : parents: ['t_sto_0'] : childs: ['t_sm_0',
't_sctto_2', 't_sto_1']
t_sto_1 : 20.002 : parents: ['t_ectto_1', 't_eto_0'] : childs: ['t_eto_1']
t_sctto_2 : 20.002 : parents: ['t_eto_0'] : childs: ['t_ectto_2']
t_sm_0 : 20.002 : parents: ['t_eto_0'] : childs: ['t_em_0']
t_eto_1 : 30.002 : parents: ['t_sto_1'] : childs: ['t_sm_1',
't_sto_2', 't_sctto_3']
t_ectto_2 : 30.002 : parents: ['t_sctto_2'] : childs: ['t_sto_2']
```

t_sto_2 : 30.003 : parents: ['t_ectto_2', 't_eto_1'] : childs: ['t_eto_2']

t_sctto_3 : 30.003 : parents: ['t_eto_1'] : childs: ['t_ectto_3']

t_sm_1 : 30.003 : parents: ['t_eto_1'] : childs: ['t_em_1']

t_eto_2 : 40.003 : parents: ['t_sto_2'] : childs: ['t_sto_3', 't_sl_1', 't_sm_2']

t_ectto_3 : 40.003 : parents: ['t_sctto_3'] : childs: ['t_sto_3']

t_sto_3 : 40.004 : parents: ['t_eto_2', 't_ectto_3'] : childs: ['t_eto_3']

t_sm_2 : 40.004 : parents: ['t_eto_2'] : childs: ['t_em_2']

t_eto_3 : 50.004 : parents: ['t_sto_3'] : childs: ['t_sm_3', 't_sl_2', 't_sl_1']

t_sm_3 : 50.005 : parents: ['t_eto_3'] : childs: ['t_em_3']

t_em_0 : 61.002 : parents: ['t_sm_0'] : childs: ['t_sl_0']

t_em_1 : 72.003 : parents: ['t_sm_1'] : childs: ['t_sl_1']

t_sl_1 : 72.004 : parents: ['t_em_1', 't_eto_2', 't_eto_3'] : childs: ['t_el_1']

t_el_1 : 82.004 : parents: ['t_sl_1'] : childs: ['t_st_1', 't_sl_2']

t_st_1 : 82.005 : parents: ['t_el_1'] : childs: ['t_et_1']

t_em_2 : 83.004 : parents: ['t_sm_2'] : childs: ['t_sl_2']

t_sl_2 : 83.005 : parents: ['t_eto_3', 't_em_2', 't_el_1'] : childs: ['t_el_2']

t_et_1 : 92.005 : parents: ['t_st_1'] : childs: ['t_sl_0']

t_el_2 : 93.005 : parents: ['t_sl_2'] : childs: ['t_st_2', 't_sl_0']

t_st_2 : 93.006 : parents: ['t_el_2'] : childs: ['t_et_2']

t_sl_0 : 93.006 : parents: ['t_el_2', 't_em_0', 't_et_1'] : childs: ['t_el_0']

t_et_2 : 103.006 : parents: ['t_st_2'] : childs: ['t_sl_3']

t_el_0 : 103.006 : parents: ['t_sl_0'] : childs: ['t_st_0', 't_sl_3']

t_st_0 : 103.007 : parents: ['t_el_0'] : childs: ['t_et_0']

t_et_0 : 113.007 : parents: ['t_st_0'] : childs: []

t_em_3 : 194.005 : parents: ['t_sm_3'] : childs: ['t_sl_3']

t_sl_3 : 194.006 : parents: ['t_et_2', 't_el_0', 't_em_3'] : childs: ['t_el_3']

t_el_3 : 204.006 : parents: ['t_sl_3'] : childs: ['t_st_3']

t_st_3 : 204.007 : parents: ['t_el_3'] : childs: ['t_et_3']

t_et_3 : 214.007 : parents: ['t_st_3'] : childs: []

==========

t_ectto_3 - t_sctto_3 >= 10

t_sl_3 - t_sm_3 <= 150

t_sctto_0 <= 10

t_el_0 - t_sl_0 >= 10

t_st_3 - t_el_3 <= 0.01

t_sto_3 - t_eto_2 >= 0.001

t_ectto_0 - t_sctto_0 >= 10

t_sctto_2 <= 40

t_eto_0 - t_sto_0 >= 10

t_sl_2 - t_sm_2 <= 73

t_eto_3 - t_sto_3 >= 10

t_el_3 - t_sl_3 <= 10.01

t_sm_3 - t_eto_3 <= 0.01

t_eto_0 - t_sto_0 <= 10.01

t_et_3 - t_st_3 >= 10

t_ectto_2 - t_sctto_2 <= 10.01

t_sm_1 - t_eto_1 >= 0.001

t_sm_0 - t_eto_0 <= 0.01

t_sctto_1 <= 10

t_et_3 - t_st_3 <= 10.01

t_ectto_1 - t_sctto_1 >= 10

t_sl_0 - t_sm_0 <= 84

t_sl_1 - t_em_1 >= 0.001

t_et_2 <= 120

t_el_2 - t_sl_2 >= 10

t_el_3 - t_sl_3 >= 10

t_sl_3 - t_et_2 >= 0.001

t_sm_3 - t_eto_3 >= 0.001

t_sctto_3 >= 0

t_sl_1 - t_eto_2 >= 0.001

t_eto_1 - t_sto_1 <= 10.01

t_et_1 - t_st_1 >= 10

t_et_2 - t_st_2 <= 10.01

t_st_0 - t_el_0 <= 0.01

t_ectto_3 - t_sctto_3 <= 10.01

t_et_2 - t_st_2 >= 10

t_sm_1 - t_eto_1 <= 0.01

t_et_1 <= 100

t_st_1 - t_el_1 >= 0.001

t_eto_2 - t_sto_2 >= 10

t_sl_1 - t_sm_1 <= 60

t_st_2 - t_el_2 >= 0.001

t_st_0 - t_el_0 >= 0.001

t_sto_1 - t_ectto_1 >= 0.001

t_sctto_3 <= 70

t_sl_0 - t_el_2 >= 0.001

t_el_0 - t_sl_0 <= 10.01

t_sctto_2 >= 0

t_sctto_0 >= 0

t_el_1 - t_sl_1 <= 10.01

t_sl_0 - t_em_0 >= 0.001

t_em_3 - t_sm_3 >= 144

t_et_0 <= 130

t_ectto_1 - t_sctto_1 <= 10.01

t_st_3 - t_el_3 >= 0.001

t_el_2 - t_sl_2 <= 10.01

t_sm_2 - t_eto_2 <= 0.01

t_et_0 - t_st_0 <= 10.01

t_el_1 - t_sl_1 >= 10

t_sl_2 - t_eto_3 >= 0.001

t_sto_2 - t_ectto_2 >= 0.001

t_sto_0 - t_ectto_0 >= 0.001

t_em_0 - t_sm_0 >= 41

t_st_2 - t_el_2 <= 0.01

t_sctto_1 >= 0

t_eto_1 - t_sto_1 >= 10

t_sl_1 - t_eto_3 >= 0.001

t_sto_3 - t_ectto_3 >= 0.001

t_sm_2 - t_eto_2 >= 0.001

t_sl_2 - t_em_2 >= 0.001

t_sl_0 - t_et_1 >= 0.001

t_sl_2 - t_el_1 >= 0.001

t_ectto_0 - t_sctto_0 <= 10.01

t_eto_3 - t_sto_3 <= 10.01

t_sl_3 - t_el_0 >= 0.001

t_et_3 <= 250

t_sto_2 - t_eto_1 >= 0.001

t_et_1 - t_st_1 <= 10.01

t_eto_2 - t_sto_2 <= 10.01

t_sm_0 - t_eto_0 >= 0.001

t_sl_3 - t_em_3 >= 0.001

t_sctto_2 - t_eto_0 >= 0.001

t_em_1 - t_sm_1 >= 42

t_et_0 - t_st_0 >= 10

t_sto_1 - t_eto_0 >= 0.001

t_em_2 - t_sm_2 >= 43

t_ectto_2 - t_sctto_2 >= 10

t_sctto_3 - t_eto_1 >= 0.001

t_st_1 - t_el_1 <= 0.01

==========

==========

total run time: 0.054914499999999755

==========