**המחלקה להנדסת תוכנה**

**פרויקט גמר – ה'תשפ"ד**

**אלגוריתמי תזמון עבור חישוב מקבילי בסביבה הטרוגנית**

**Task Scheduling for Parallel Computation in Heterogeneous Systems**

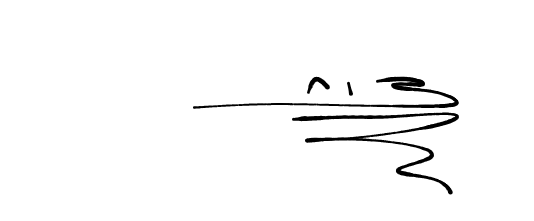
**מאת**

**אליה אטלן**

**318757200**

**אביב זבולוני**

**211313333**



**מנחה אקדמי/ת: ד"ר צור לוריא אישור: תאריך: 2024\07\17**

מערכות ניהול הפרויקט:

|  |  |  |
| --- | --- | --- |
| # | מערכת | מיקום |
| 1 | מאגר קוד | [Repositories (github.com)](https://github.com/orgs/Task-Scheduling-Team-Azrieli/repositories) |
| 2 | קישור ליומן | [יומן פגישות](https://docs.google.com/document/d/18pG3YXsfY8yLas0scpnb4-WNK5lNeO9h-yugJFtvGOg/edit?usp=sharing) |
| 3 | קישור לסרטון דוח אלפא | [דוח\_אלפה.mp4](https://drive.google.com/file/d/1WMnKhALSdBLPtn6So8Evf9wjV-cNyTh6/view?usp=sharing) |

מידע נוסף (מחקו את המיותר)

|  |  |
| --- | --- |
| סוג הפרויקט | 1. מחקרי ממרצה במכללה 2. תעשייתי חברת hi-tech |
| פרויקט ממשיך | פרויקט המשך של שיר גולה שכותרתו "אופטימיזציית המתזמן עבור מחשוב מקבילי בזמן אמת".  פרויקט זה בא להציע גישה שונה לפתרון אותה הבעיה ששיר ניסתה לפתור |

**תוכן עניינים**

[**נאום המעלית** 1](#_Toc172715091)

[**מבוא** 2](#_Toc172715092)

[**תיאור הבעיה** 6](#_Toc172715093)

[**דרישות ואפיון הבעיה** 7](#_Toc172715094)

[**הבעיה מבחינת הנדסת תוכנה** 7](#_Toc172715095)

[**סקירת עבודות דומות בספרות והשוואה** 8](#_Toc172715096)

[**תיאור הפתרון** 9](#_Toc172715097)

[**המשך הפרק תיאור הפתרון** 10](#_Toc172715098)

[**תוצאות ומסקנות** 11](#_Toc172715099)

[**נספחים** 16](#_Toc172715100)

[תכנון הפרויקט – ברזולוציה של שבועיים 19](#_Toc172715101)

# **נאום המעלית**

בעולם הטכנולוגי המודרני, מופיעים קשיים חדשים שדורשים פתרונות חדשניים. אחד מהם הוא הצורך לתזמן\לשבץ מספר רב של משימות התלויות זו בזו על מספר מעבדים מסוגים שונים, כאשר כל משימה מוכרחת לרוץ על מעבד מסוג ספציפי.

בעיה זו רלוונטית מתמיד לאור השימוש הנפוץ ברשתות נוירונים הדורש מעבדים ייחודיים לריצה אופטימאלית.

בימינו, התעשייה משתמשת באלגוריתמים פשוטים שמשבצים משימות למעבדים באופן שרירותי.

בפרויקט שלנו, חקרנו אלגוריתמים חדשניים לתזמון מעבדים בסביבה זו. באמצעות אלגוריתמים אלו, ניסנו למזער את זמן הריצה הכולל של המערכות.

# **מבוא**

מאז שנות ה 90, עולם המחשוב המבוזר צמח. עקב כך התפשטו רשתות מחשבים גדולות ומורכבות יותר, בנוסף לשימוש בארכיטקטורות client-server.

תהליך זה הביא לצורך גדול יותר בתזמון מעבדים, שכן המערכות המורכבות האלה דרשו תיאום וניהול יעיל של משאבי המערכת. על מנת למקסם את רווחיהם, חברות הענן שאפו (ושואפות) "לשבץ" כמה שיותר משימות בזמן הקצר ביותר (מזעור זמן ה idle של המעבדים).  
  
בנוסף להתפתחות הענן, התפתחות הלמידה העמוקה באמצעות רשתות נוירונים עמוקות דרש פיתוח מעבדים מסוג חדש, המעבדים החדשים הותאמו עבור עיבוד מרובה משימות (כגון עיבוד מקבילי של מידע בכמויות עצומות). התפתחות הגרפיקה הביאה להפצת המעבדים החדשים הללו שתמכו בעיבוד גרפי. עם הלמידה העמוקה, עלה הצורך בתזמון מעבדים מסוגים שונים המותאמים למשימות ספציפיות על מנת לתמוך במטלות שדורשות חישובים מורכבים, מקביליים ומהירים.

הפרויקט שלנו הוא גם פרויקט תעשייתי, בשיתוף עם חברת Mobileye והמנחה התעשייתי יוסי קריינין. למערכות הפועלות בזמן אמת כמו אלה המפותחות על ידי Mobileye (העוסקת בפיתוח מערכות נהיגה אוטונומית) עולה הדרישה לאלגוריתמים יעילים שיתזמנו משימות בזמן מינימלי, וישתמשו בחומרה ייעודית המתאימה למשימות ספציפיות.

במערכות כמו של Mobileye, שעוסקות בעיבוד תמונה וקבלת החלטות בזמן אמת, חוץ מדרישה למהירות ביצוע מרבית, ישנו גם צורך באמינות ובדיוק גבוה. אלגוריתמי תזמון ישפרו את יכולת המערכת לתפקד בזמן אמת.

כיום בתעשייה משתמשים באלגוריתם תזמון חמדן. אלגוריתם זה משבץ משימה מוכנה למעבד פנוי באופן שרירותי. בנוסף לכך החמדן חסכן במשאבים מכיוון שהוא לא מתחזק מבנה נתונים מלבד תור\רשימה של משימות מוכנות.

בעיית תזמון מעבדים היא בעיה ידועה ונפוצה בתעשייה שמתעסקת בניהול ובשיפור שימוש המעבדים במערכות תוכנה. הבעיה עוסקת בניסיון לתזמן משימות למערכות שמכילות מספר מעבדים, על מנת לייעל את המערכת על פי קריטריונים שונים לפי דרישות המערכת.  
  
עולם בעיות התזמון הוא מאוד רחב, וכל בעיה בעולם זה אפשר לתאר על ידי שלושת הקריטריונים הבאים:

**המעבדים:**

\* **1:** רק מעבד אחד, עליו רצים כל המשימות.

\* **P:** מספר מעבדים מאותו סוג, כאשר מספר המעבדים הוא P

\* **Q:** מספר מעבדים מסוגים שונים ( Q סוגי מעבדים)

\* **R:** מספר מעבדים מסוגים שונים, כאשר זמן הריצה של כל משימה  
 תלוי במעבד.

וכולי.

**אילוצים על תזמון המשימות:**

\* **prec:** תחילת ריצה של משימה יכולה להיות תלויה בהשלמת   
 משימה\משימות אחרות.

\* **due date:** על משימות מסוימות להסתיים עד זמן נתון.

\* **release time:** משימות מסוימות יכולות לרוץ רק לאחר שמאורע   
 מסוים קרה.  
  
 \* **processing time:** זמני הריצה של המשימות, יכולים להיות כולם   
 שווים, או חלק מקבוצה מסויימת.

וכולי, אנחנו נתמקד באילוצים מסוג prec, כאשר ה processing times   
 שייכים ל

**המטרה:**

**:**מזעור הזמן מתחילת המשימה הראשונה, ועד סיום המשימה   
 האחרונה *\** **:** מזעור סכום זמני הריצה אילו היה ברשותנו רק מעבד אחדאנחנו נתמקד ב **.**כל הווריאציות ומידע עליהן, נמצא ב[[1]](http://schedulingzoo.lip6.fr/)

בפרויקט זה אנו מתמקדים במצב בו ישנם מספר סוגים שונים של מעבדים, וכל תהליך צריך לרוץ על סוג מעבד מסוים. המשימות כולן חלק מאלגוריתם גדול ומסובך, ולכן ישנן תלויות רבות ביניהן. אנו רוצים למזער את זמן הריצה הכולל. בשפת התזמון, אנו מתמקדים ב-**Q|prec|Cmax**.

למרות שמדובר בבעיה חשובה ומרכזית למטרות תעשייתיות, מפתיע מאוד לגלות שיש מעט מאוד ספרות בנושא - ההסביר לכך הוא שיש כל כך הרבה ווריאציות של בעיות תזמון - ממש מאות, ומעט מאוד העמיקו בבעיה הספציפית שאנו מתמקדים בה.

**אלגוריתמי online ואלגוריתמי offline:**

בעת חישוב סדר המשימות כפי שהאלגוריתם מתאר, ניתן להוסיף אילוץ על זמן

ריצת המתזמן.

**אלגוריתם online:** דורש זמן חישוב מאוד מהיר כך שיהיה זניח לעומת זמן הריצה

הכולל של המשימות (על מנת שלא יפגע בביצועי המערכת),

מפני שהתזמון מחושב בזמן ריצת המשימות, בכל פעם שהן

צריכות להשתבץ למעבד, בנוסף אין מידע על זמן הריצה בפועל

של המשימות לפני הרצתן (ניתן לקרב ע"י ריצות קודמות

במקרים מסוימים) ולכן לא ניתן להסתמך על זמן זה לצורך התזמון.

**אלגוריתם offline:** מבצע חישוב מראש של התזמון שיכול לערוך זמן רב

(ביחס למשימות להרצה), בעל גישה למידע המלא על זמן הריצה בפועל והתלויות בין המשימות, ולבסוף מחזיר את רשימת סדר

הריצות

של המשימות על כל המעבדים.

# **תיאור הבעיה**

אנחנו ענינו על 3 שאלות מרכזיות במהלך הפרויקט:

**שאלה 1:**

האם קיים אלגוריתם קירוב יותר טוב מהחמדן לתזמון מעבדים בסביבה הטרוגנית (מעבדים מסוגים שונים)?

**שאלה 2:**

האם קיים\קיימים אלגוריתמים טובים יותר מהחמדן על קלטים\בעיות מהעולם האמיתי?

שאלה 3:

מהו האלגוריתם הכי טוב לחומרה הספציפית של Mobileye?  
  
בנוסף לסימולציה שנפתח בשלב השני (שאלה 2), נוסיף אילוצים על האלגוריתם כך שיתאים למתזמן של Mobileye (אשר הוטמע בחומרה) שבשימוש כיום.  
כיום Mobileye משתמשים במתזמן חמדן שעובד עם רשימות של משימות בעלי 2 רמות דחיפות, ומשבץ מהתור הדחוף למעבד זמין (אם התור הדחוף ריק, אז הוא מתזמן מהתור הפחות דחוף).

כך נוכל להשוות את האלגוריתם החמדן ש Mobileye מריצים כיום עם

האלגוריתמים שנפתח.

## **דרישות ואפיון הבעיה**

הדרישות של הלקוח (Mobileye) הן:

1. אלגוריתם שנותן תזמונים יותר טובים מאשר האלגוריתם החמדן הקלאסי שבשימוש כיום. אפילו אם זה רק משפר את המקרה הממוצע, ולא את המקרה הגרוע.
2. אלגוריתם שאפשר להטמיע בחומרה הקיימת.

התוצר ש Mobileye יקבלו יהיה מספר אלגוריתמים שונים, שנפתח בשפת Python בנוסף ל Pseudo-Code שממנו פותחו האלגוריתמים.

## **הבעיה מבחינת הנדסת תוכנה**

הפרויקט דורש:

1) מציאת חסמים על זמני ריצה של אלגוריתמי תזמון.

2) פיתוח אלגוריתמי תזמון חדשים.

3) מציאת היוריסטיקות על מנת לייעל את האלגוריתמים הקיימים.

4) פיתוח מערכת סימולציה על מנת להריץ ולבחון את האלגוריתמים השונים.

# **סקירת עבודות דומות בספרות והשוואה**

1. [The scheduling zoo](http://schedulingzoo.lip6.fr/)

זהו מאגר נרחב של קונפיגורציה מסויימת של בעיות תזמון, והמחקרים או העבודות המתאימים אליהן בספרות. מסד נתונים זה הוא עצום, שכן הוא כולל את כל הבעיות בתחום והוא עוזר להדגיש את רוחב התחום ואת מגוון הגישות שנחקרו. ומכאן הסיבה למחסור היחסי של ספרות המתייחסת ספציפית לבעייה שלנו תזמון משימות במערכות הטרוגניות (Q|prec|Cmax), שכן זה מדגיש את המשמעות של פרויקט זה.

1. [Svensson, Ola. "Hardness of precedence constrained scheduling on identical machines." SIAM Journal on Computing 40.5 (2011): 1258-1274.](https://theory.epfl.ch/osven/Ola%20Svensson_publications/SICOMP11b.pdf)

עבודה זו מתעמקת באתגרים של תזמון עם אילוצי קדימות. למרות שהעבודה מתמקדת במכונות זהות (P|prec|Cmax), היא מדגישה את הקושי בתזמון אפילו באילוצים פשוטים על המערכת - Svensson מראה במחקר שאפילו עם כמות אינסופית של מכונות זהות, ואפילו כשלמשימות יש זמן ריצה זניח, הבעיה עדיין 2-מקרבת. פרויקט זה מרחיב את הניתוח של Svensson לסביבות הטרוגניות, ששם מאפייני הביצועים המשתנים של מעבדים מסבכים עוד יותר את תזמון הקדימות. התובנות מהמחקר של Svensson עזרו לנו לנתח את המורכבות של בעיית התזמון שלנו למערכות הטרוגניות.

1. [Graham, Ronald L. "Bounds for certain multiprocessing anomalies." Bell system technical journal 45.9 (1966): 1563-1581.](http://compalg.inf.elte.hu/~tony/Kutatas/BinPacking/Graham-1966-BoundsforAnomalies.pdf)

בעבודתו, מציג Graham את המושג של תזמון משימות מאפס עם דוגמאות הרצה והגדרות מדויקות של מערכת עם קדימויות, אבל עם מעבדים זהים (P|prec|Cmax). Graham עזר לנו לנתח את הגבולות התיאורטיים לתזמון משימות במערכות הטרוגניות. הניתוח של אלגוריתמי תזמון חמדנים של Graham עזר לקבוע מידת בסיס לביצועים של היוריסטיקות התזמון שפיתחנו. פרויקט זה מתבסס על עקרונותיו של גרהם, שכן הם גם חלים על סביבות הטרוגניות.

1. [Pinedo, M. L. (2012). Scheduling (Vol. 29). New York: Springer.](https://web-static.stern.nyu.edu/om/faculty/pinedo/schedtheory/book4/overview-downloadable-material.pdf)

בספר, Pinedo מביא בחינה מקיפה של מודלים, אלגוריתמים והיוריסטיקות של תזמון. רבים מהם סייעו בפיתוח של אלגוריתמי התזמון בפרויקט זה, והיו בעלות ערב רב מכיוון שהיו בסיס טוב להתחיל ממנו.

בהשוואה לספרות הקיימת, פרויקט זה תורם לתחום באמצעות טיפול בפער במחקר הקשור למערכות הטרוגניות, באמצעות שילוב היסודות התיאורטיים של עבודתם של Graham ו Svensson עם ההיוריסטיקות והמודלים המעשיים שהוצגו על ידי Pinedo.

# **תיאור הפתרון**

הפתרון דורש לענות על שלושת השאלות המרכזיות בפרויקט:

**שאלה 1:**

ננתח זמני ריצה, נוכיח חסמים על ידי בניות (קונפיגורציות) שנמצא ובעזרת היוריסטיקות

ננסה למטב את החמדן ה"טיפש".  
  
**שאלה 2:**

ניקח קלט מהעולם האמיתי ש Mobileye תספק לנו, ונפתח אלגוריתמים שיכולים

לייעל את זמן הריצה של החמדן על הקלטים שנלקחו מהעולם האמיתי.

**שאלה 3:**

ניעזר באלגוריתמים שפיתחנו בחלק השני של הפרויקט (שאלה 2) על מנת לפתח אלגוריתם

שיתאים לחומרה הקיימת במערכות של Mobileye.   
  
נפתח סימולטור על מנת לבחון את האלגוריתם שלנו אל מול האלגוריתם החמדן הקיים

באמצעות מידע קיים של Mobileye. במידה ומצאנו אלגוריתם יותר טוב, נציע אותו ל Mobileye.

# **המשך הפרק תיאור הפתרון**

הפרויקט שלנו מסתמך מאוד על הסימולציה שיצרנו לתזמון המעבדים.

קודם כל רשמנו פסאודו-קוד שיסמלץ הרצת מעבדים בסביבה הטרוגנית (ראו נספח [1]).

(בשביל דוגמת הרצה ראו נספח [5])

לאחר מכן היינו צריכים לחשוב איך נסמלץ מעבדים ומשימות, לשם כך יצרנו את המחלקות Task, Processor אשר מייצגות משימות ומעבדים, וכל המידע שדרוש על מנת להריץ את הפסאודו-קוד   
(ראו נספח [2])

**איך השגנו דאטא?**

הפרויקט הוא בשיתוף עם Mobileye, ולכן נציג מ Mobileye נתן לנו גישה להמון דאטא.

עד שהדאטא הגיע, עבדנו עם קבצים json, ולכן כשהדאטא הגיע רשמנו קוד במחלקה ששמה Parser, שהמטרה שלה היא לנתח את הקבצים הללו ולהפוך אותם לפורמט ה json שהשתמשנו בו.

(דוגמא ל input בנספחים למטה ראו נספח [3])

לאחר מכן, מכיוון שהפרויקט עצמו דורש לבנות אלגוריתמים חדשים, יצרנו מחלקה ששמה Algorithm כאשר כל אלגוריתם שנממש ירש ממחלקה זו.

למחלקה הזו יש פונקציית decide, ובעצם זה הלב של המתזמן והיא מה שקובעת איך המשימות תשובצנה. (ראו נספח [4])

לבסוף, כתבנו את המחלקה Sim שקודם כל קוראת את הדאטא, ולאחר מכן מריצה את מימוש הפסאודו-קוד.

# **תוצאות ומסקנות**

* ~~מציאת חסמים עליונים ותחתונים לאלגוריתם החמדן הבסיסי בסביבה הטרוגנית~~
* ~~פיתוח פסאודו-קוד להרצת הסימולציות ע"מ לבחון את האלגוריתמים השונים~~
* ~~פיתוח ב Python את מערכת הסימולציה בהשראת הפסאודו-קוד~~
* ~~הוספת אילוסטרציה גרפית כדי שנוכל לנתח את זמני הריצה ולממש את היוריסטיקות~~
* ~~דיון ומחקר על היוריסטיקות\אלגוריתמים שכדאי לבדוק~~
* ~~מימוש האלגוריתמים~~
* ~~הרצת האלגוריתמים על כל ה Data ובדיקת ממוצע זמני הסיום לצורך השוואה.~~
* ~~דיון וחשיבה על ההיוריסטיקה המבטיחה critical path~~
* ~~מימוש critical path והרצתו~~
* הצעת ודיון על יוריסטיקות\אלגוריתמים נוספים
* הצגת התוצאות עד כה לנציג מובילאיי
* הצעת אלגוריתם אופטימלי (מתוך אלו שחקרנו) עבור החומרה הקיימת במערכת מובילאיי
* במידה וה crititical path (שהוא אלגוריתם offline) משפר בהרבה את זמן הריצה נשקול לפתח מתזמן שיתזמן לפי critical path ו clustering ע"מ לאפשר את הרצת התזמון בזמן אמת (online)

**בהתייחס לשאלת המחקר הראשונה שלנו:**

האם קיים חסם תחתון לאלגוריתם חמדן בסביבה הטרגונית?

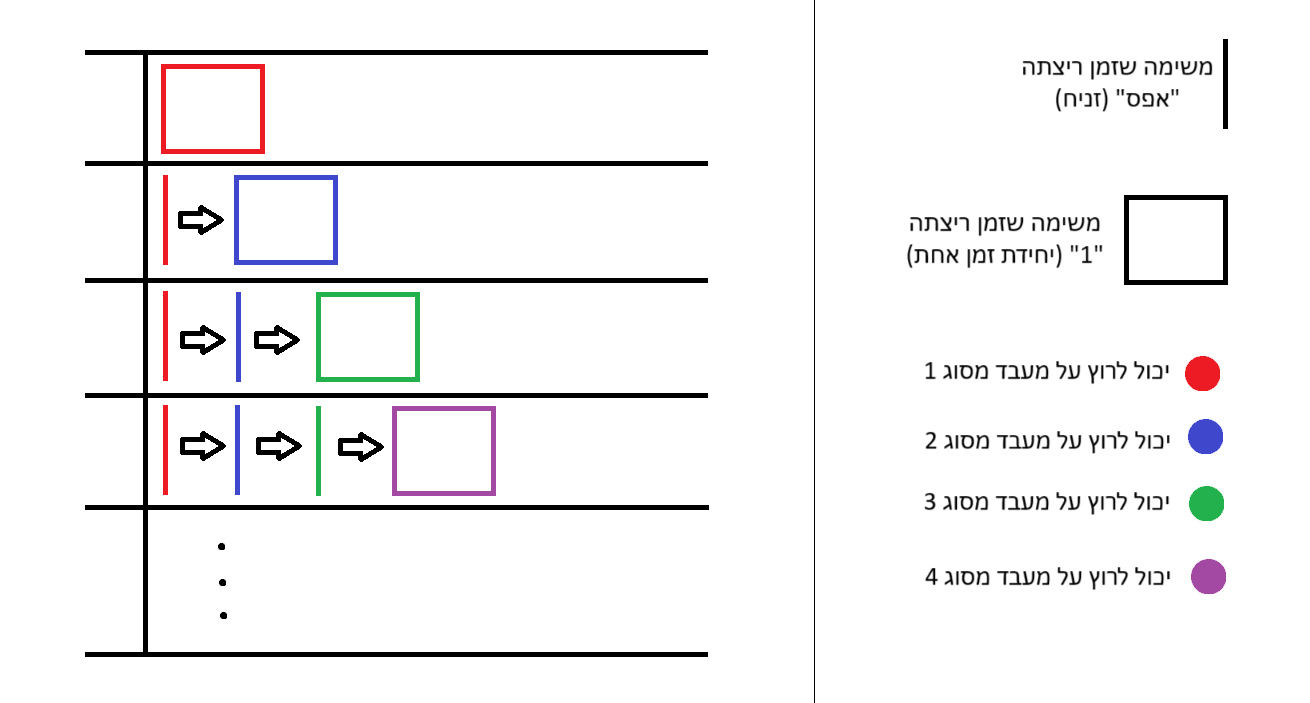
מדוע שנפנה בכלל לאלגוריתם חמדן שמביא רק קירוב לפתרון הבעיה?

אפילו בעיית התזמון על שני מעבדים זהים עם משימות בעלי זמני ריצה שונים היא NP קשה (זו בעיית Partition). מכאן שניתן לבנות רדוקציה מווריאציה זו לכל ווריאציה אחרת של הבעיה, וכך גילינו שגם הבעיה שלנו היא NP קשה.

לכן, לא ניסינו למצוא אלגוריתם אופטימלי לבעיה. במקום, ניסינו להתקרב כמה שיותר לפתרון האופטימלי בדומה לגישה לפתרון בעיות NP-קשות אחרות.

הוכח [[2]](https://theory.epfl.ch/osven/Ola%20Svensson_publications/SICOMP11b.pdf) שעבור בעיית תזמון מעבדים בסביבה הומוגנית (כאשר כל המעבדים מאותו סוג), לא ניתן לשפר את הקירוב של החמדן [[3]](http://compalg.inf.elte.hu/~tony/Kutatas/BinPacking/Graham-1966-BoundsforAnomalies.pdf) (2-מקרב) בזמן פולינומי.

בעזרת בניית דוגמה בה החמדן הוא לכל היותר P מקרב, מצאנו חסם תחתון עבור אלגוריתם חמדן בסביבה הטרוגנית:



כפי שניתן לראות בשרטוט בכל שורה ישנן משימות כשאר כל משימה תלויה במשימה שלמאלה, צבע המשימה מצביע על סוג המעבד היחיד שיכול לבצע אותה, משימה שהיא פס זמן הריצה שלה הוא 0 (או קטן כרצוננו לצרכים מעשיים) וריבוע הוא משימה שזמן ריצתה הוא 1, אלגוריתם חמדן עלול להריץ את המשימות לפי סדר השורות וכך עבור P סוגי מעבדים, כלומר P יחידות זמן, בעוד אלגוריתם אופטימאלי יריץ הכל ביחידת זמן אחת בלבד (ע"י הרצת כלל המשימות שזמנן 0 לפי סדר העמודות, וריצה במקביל של כלל המשימות בעלות זמן 1).

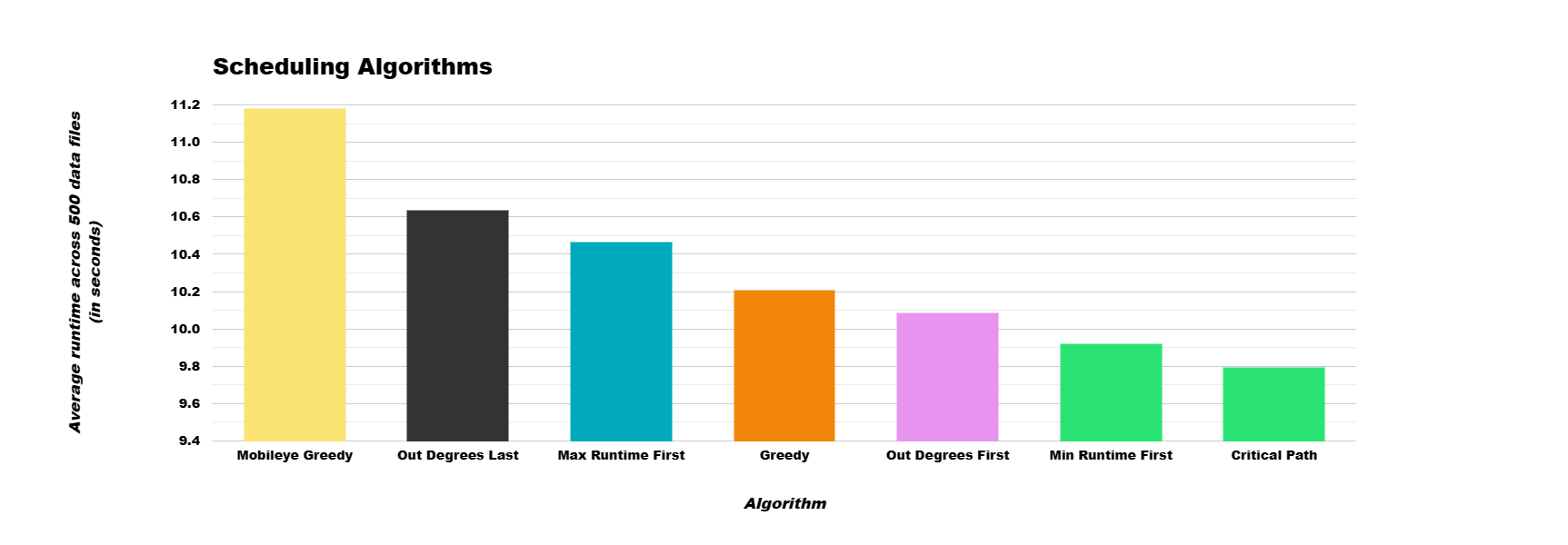
**כעת נתמקד בתוצאות שאלת המחקר השנייה:**

האם קיימים אלגוריתמים יעילים יותר מהחמדן עבוד בעיות תזמון מהעולם האמיתי?

את נושא זה חקרנו לעומק באמצעות העזרה של Mobileye, אשר סיפקו לנו דוגמאות מהעולם האמיתי.

על מנת לענות על שאלה זו, פיתחנו סימולטור (ב python) שמסמלץ סביבה הטרוגנית כללית על מנת לבחון אלגוריתמים שונים ולהשוות ביניהם. לאחר מכן, פיתחנו אלגוריתמי תזמון על ידי שימוש בהיוריסטיקות שונות על מנת לנסות לייעל את החמדן כאשר זמני הריצה והתלויות בין המשימות ידועים מראש.

כפי שניתן לראות בגרף, האלגוריתם החמדן של Mobileye הוא בעל הביצועים הנמוכים ביותר ביחס להיוריסטיקות, שכן הוא אלגוריתם online, ונתון לאילוצי חומרה של המעבדים של Mobileye בעוד שההיוריסטיקות שלנו הן בגדר offline, זמני הריצה של המשימות ידועים מראש, ואין אף אילוץ על החומרה.

השוואת אלגוריתמים לפי זמן ריצה כולל:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Critical Path | Min Runtime First | Out Degrees First | Greedy | Max Runtime First | Out Degrees Last | Mobileye Greedy |  |
| ~95.9% | ~97.2% | ~98.8% | 100% | ~102.5% | ~104.2% | ~109.5% | % Runtime From Greedy |
| ~4.2% | ~2.9% | ~1.2% | 0% | ~2.4%- | ~4%- | ~8.7%- | % Faster |
| 2.5% | 3.5% | 2.1% | 0% | 2.6% | 3.2% | DC | Standard Deviation |

להלן השוואת האלגוריתמים ביחס לאלגוריתם החמדן המסורתי:

**ולבסוף שאלת המחקר השלישית:**

האם ניתן למצוא אלגוריתם online שמשפר את זמני הריצה?

ניסנו להתאים את היוריסטיקות שבחרנו בשאלה הקודמת לחומרה של Mobileye, שבה אין תור קדימויות אלא רק 2 תורים בלבד "דחוף" ו "לא דחוף" כאשר כל המשימות הדחופות סיימו לרוץ, המתזמן משבץ את הלא דחופות.

בחלק זה של הפרוייקט המרנו את ההיוריסטיקות שהביאו סדר מלא, להיוריסטיקות שרק "צובעות" משימות ב"דחוף"\ "לא דחוף" לפי ערך סף מסוים (כמובן שנדרשנו להוסיף יכולות לסימולטור שלנו ע"מ לתמוך בסימולציה כזו.(

הצביעה התבצעה על מנת קביעת ערכי סף עפ"י חציונים, כל היוריסטיקה נבדקה על עד 7 ערכי סף שונים.

ולהלן התוצאות (חלקן, כיוון שלא הצלחנו להסיק דבר על התוצאות המלאות בשל השונות הגדולה(:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Out Degrees First | Greedy | Min Runtime First | Max Runtime First | Out Degrees Last |  |
| 2 | DC | 4791.6 | 4791.6 | 2 | Threshold |
| 98.8% | 100% | 101.2% | 101.3% | 104.2% | Runtime % From Greedy |
|  | 0% |  |  |  | % Faster |
| 2.16% | 0% | 2.3% | 2.2% | 3.3% | Stadnard Deviation |

במהלך העבודה על הפרויקט נתקלנו בקשיים מסוגים שונים.

ניסינו במשך זמן לא מבוטל להוכיח שאלגוריתם חמדן הוא גרוע יותר מP מקרב בעזרת בניות שונות, ללא הצלחה.

בנוסף הסימולטור שפיתחנו נעשה מורכב יותר ויותר ככל שהתקדמנו בשאלות המחקר, ונדרשנו להוסיף יכולות לקוד הקיים, מבלי לפגוע ביכולות הקיימות.

הדבר הביא אותנו לעצב את הקוד מחדש בכל הוספה על מנת לשמור על הביצועים, היכולות הקיימות, הקריאות של הקוד וכמובן תכנות נכון לפי עקרונות הOOP.

קושי נוסף שהתמודדנו עימו הוא בדיקת הסימולטור שלנו.

כמובן שבדקנו את הסימולטור עבור מקרים פשוטים אך עבור DAG-ים עם מאות משימות הדבר נהפך קשה עד בלתי אפשרי.

בנוסף בכל יכולת שהוספנו לסימולטור צצו באופן טבעי בעיות בקוד החדש והקיים, נעזרנו בתכנון נכון על מנת למזער את הבאגים ולמצוא את שורש הבעיות ולתקנן.

נוסיף שנדרשנו להתמודד גם עם כמויות גדולות של מידע שדרש ניתוח, בחרנו להציג ולנתח את המידע בקובץ אקסל על מנת להקל עלינו את הניתוח ולהסיק מסקנות מכל המידע שנאסף מהסימולציות.

בנימת סיום, אמנם תוצאותינו לא מצביעים על אלגוריתם online שמסוגל לרוץ על החומרה של Mobileye ולגרום באופן גורף לשיפור את זמני הריצה במקרה הממוצע, אך הפעלת Clustering על ריצה של DAG וניסיון להתאימו לDAG שכבר ניתחנו את האלגוריתם המהיר ביותר עבורו עשוי לשפר את זמן הריצה הממוצע.

כיוון מבטיח נוסף הוא שימוש בCritical Time, כלומר לתזמן לפי משימות שעל הנתיב הקריטי.

אלגוריתם זה על אף היותו אלגוריתם offline הביא לשיפור ניכר (4.2% במקרה הממוצע ועד כ10% במקרים מסוימים), לכן מעניין לנסות להופכו לאלגוריתם Online, ע"י ניבוי הDAG הבא לפי הריצה של הDAG הקודם וניצול מידע הקשור לחלקים קבועים שקיימים בכל DAG.

כמובן שיש לקחת בחשבון את זמן חישוב הCritical Time שאמנם ירוץ בין שני DAGs אך עדיין עלול לקחת זמן על אף היותו מהיר יחסית (זמן הריצה של האלגוריתם שלנו הוא O(n) עבור n משימות בDAG). בכל מקרה שינוי כזה ידרוש שינוי קל בחומרת המתזמן, לעבור מארכיטקטורת דחוף\לא דחוף לארכיטקטורה שתומכת בהרצת משימות לפי סדר ידוע מראש.

פרק של בדיקות??

# **נספחים**

[1] פסאודו-קוד לסימולציה:

1. מכניסים את כל המשימות המוכנות לתור: ready\_tasks

עוברים על כל המעבדים וכל ה ready\_tasks עד שניתן לשבץ משימה למעבד פנוי

מעדכנים שהמעבד "עסוק"

מכניסים את המשימה ל current\_tasks

מעדכנים שה end\_time של המשימה הוא current\_time + duration

מוחקים את המשימה מ ready\_tasks

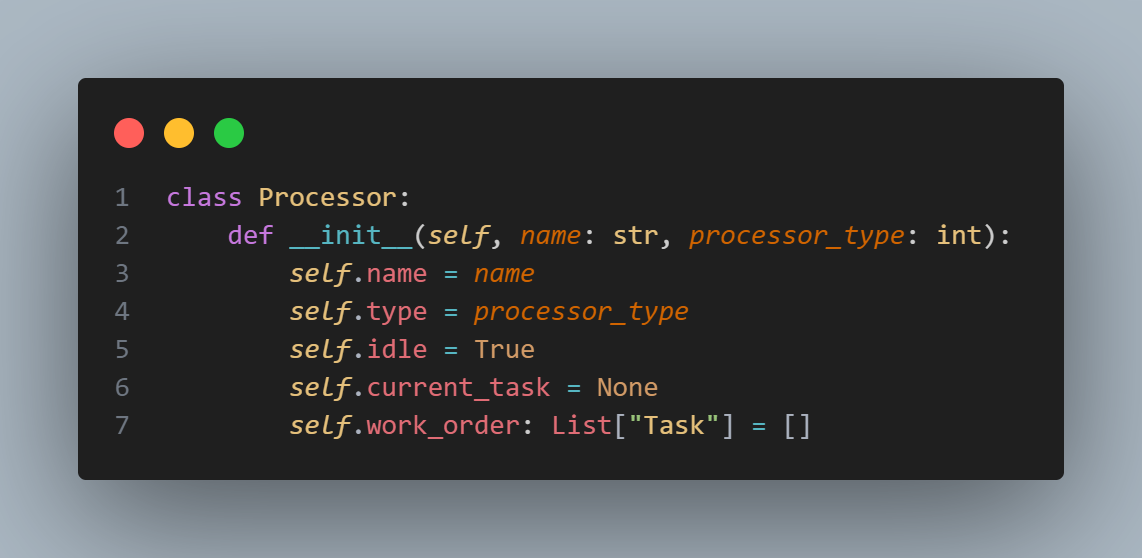
1. כל עוד יש משימות שרצות:

task = current\_tasks.pop()

current\_time = task.end\_time

task.update\_in\_degrees()

לבסוף, בצע את (1)

[2] תכונות Task, Processor



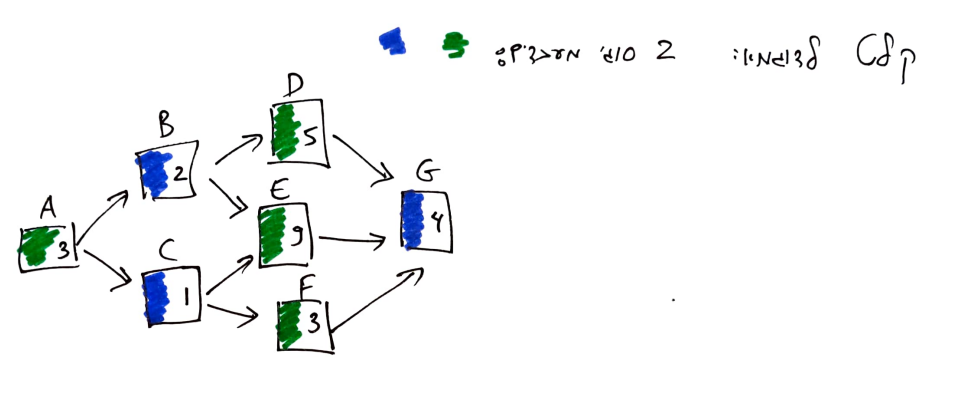


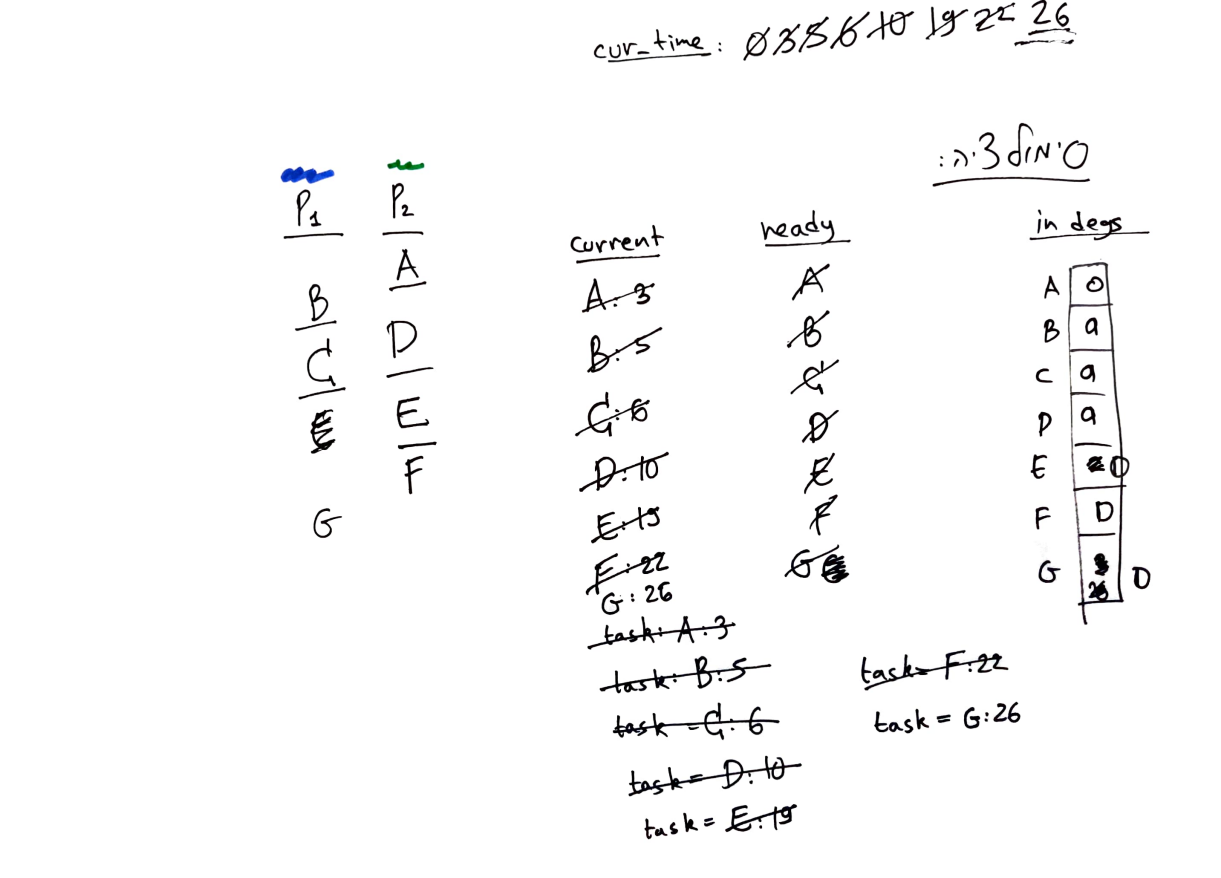
[3] דוגמא ל input:

[4] המחלקה Algorithm והפונקצייה decide:

האלגוריתם שבשימוש כיום Greedy:

[5] דוגמת הרצה:





## תכנון הפרויקט – ברזולוציה של שבועיים

|  |  |
| --- | --- |
| 23.04 | פגישת היכרות עם הנציג מ Mobileye והצגת הסימולציה והתוצאות עד כה |
| 30.4 | פגישת המשך עם צור ודיון על בחירת האלגוריתם האופטימלי עבור Mobileye |