**המחלקה להנדסת תוכנה**

**פרויקט גמר – ה'תשפ"ד**

**אלגוריתמי תזמון עבור חישוב מקבילי בסביבה הטרוגנית**

**Task Scheduling For Parallel Computation In Heterogenous Systems**

**מאת**

**אליה אטלן**

**318757200**

**אביב זבולוני**

**211313333**

**מנחה אקדמי/ת: ד"ר צור לוריא אישור: תאריך:**

מערכות ניהול הפרויקט:

|  |  |  |  |
| --- | --- | --- | --- |
| # |  | מערכת | מיקום |
| 1 |  | מאגר קוד | [Task-Scheduling-Team-Azrieli (github.com)](https://github.com/Task-Scheduling-Team-Azrieli) |
| 2 |  | יומן | [View 1 · Meeting Appointments (github.com)](https://github.com/orgs/Task-Scheduling-Team-Azrieli/projects/1/views/1) |

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 |

**מבוא**

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 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/)

נוכל לחשוב על המשימות כאילו הן מסודרות בתוך גרף מכוון ללא מעגלים (DAG) כאשר כל צומת היא משימה, וכל קשת מסמלת את התלות בין המשימה שהקשת יוצאת ממנה, למשימה אליה היא נכנסת. תלות זו מתבטאת בכך שמשימה שתלויה במשימות אחרות צריכה לחכות לסוף הריצה של כלל המשימות שהיא תלויה בהן.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 קרייניו. בנוסף למערכות ענן, 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. האלגוריתם החמדן הוא P + 1 מקרב בווריאציית הבעיה שלנו, כאשר P הוא מספר סוגי המעבדים. בנוסף לכך החמדן חסכן במשאבים מכיוון שהוא לא מתחזק מבנה נתונים מלבד תור\רשימה של משימות מוכנות.

**תיאור הבעיה**

אנחנו ננסה לתת מענה ל 3 שאלות מרכזיות במהלך הפרויקט:  
  
**שאלה 1:**

האם קיים אלגוריתם קירוב יותר טוב מהחמדן לתזמון מעבדים בסביבה הטרוגנית (מעבדים מסוגים שונים)? ננסה להתקרב לאלגוריתם האופטימלי (התיאורטי) כמה שיותר.  
  
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, ולכן גם הבעיה שלנו היא NP קשה.  
  
לכן לא ננסה למצוא אלגוריתם אופטימלי לבעיה. במקום, ננסה להתקרב כמה שיותר לפתרון האופטימלי.  
  
בנוסף לכך, 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. ננסה להשליך על הבעיה שלנו בסביבה הטרוגנית.

האם האלגוריתם החמדן הוא 2-מקרב גם במקרה שלנו? או אולי יותר גרוע? ואם כן, האם קיים אלגוריתם קירוב יעיל יותר?

**שאלה 2:**

האם קיים\קיימים אלגוריתמים טובים יותר מהחמדן על קלטים\בעיות מהעולם האמיתי?  
  
על מנת לענות על שאלה זו, נצטרך לפתח אלגוריתמי תזמון על ידי שימוש בהיוריסטיקות שונות על מנת לנסות לייעל את החמדן כאשר זמני

הריצה והתלויות בין המשימות ידועים מראש.  
  
את נושא זה נוכל לחקור יותר לעומק באמצעות העזרה של Mobileye, אשר יספקו לנו דוגמאות מהעולם האמיתי. בנוסף לכך, נפתח סימולטור שיסמלץ סביבה הטרוגנית כללית על מנת לבחון אלגוריתמים שונים ולהשוות ביניהם.

**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.

לאחר שנענה על שאלות 2,3 נוכל להציע ל Mobileye את השיפור בתוכנה (אם קיים).

## **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.

**תיאור הפתרון**

הפתרון דורש לענות על שלושת השאלות המרכזיות בפרויקט:

**שאלה 1:**

ננתח זמני ריצה, נוכיח חסמים על ידי בניות (קונפיגורציות) שנמציא 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.

**על הלקוח** (רלוונטי לפרויקטים מסוגים 2, 3, ו 4)

Mobileye טכנולוגיות ראיה בע"מ, היא חברה העוסקת בפיתוח טכנולוגית נהיגה אוטונומית וייצור מערכות סיוע מתקדמות לנהג

(ADAS), המבוססות על טכנולוגיות עיבוד תמונה, שבבי מחשב, ותוכנה.

במרץ 2017 נמכרה Mobileye לאינטל. המשרדים הראשיים שלMobileye ומרכז הפיתוח ממוקמים בירושלים, ומשרדי המכירות

פרוסים ברחבי העולם - ירושלים, תל אביב, חיפה, פתח תקווה, ניו יורק, דיסלדורף בגרמניה, שנגחאי בסין וטוקיו שביפן. באוקטובר

2022 הונפקה החברה Wall Street לפי שווי של 16.7 מיליארד דולר.

סקירת עבודות דומות בספרות והשוואה (רלוונטי לפרויקט מסוג 1)

1. [The scheduling zoo](http://schedulingzoo.lip6.fr/)
2. [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)
3. [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)