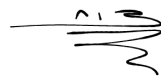


המחלקה להנדסת תוכנה
פרויקט גמר – ה'תשפ"ד
אלגוריתמי תזמון עבור חישוב מקבילי בסביבה הטרוגנית
Task Scheduling for Parallel Computation in Heterogeneous Systems

מאת
אליה אטלן
318757200
אביב זבולוני
211313333



מנחה אקדמי/ת: ד"ר צור לוריא אישור:

תאריך: 25\07\2024

מערכות ניהול הפרויקט:

#	מערכת	מיקום
1	מאגר קוד	Repositories (github.com)
2	קישור ליומן	יומן פגישות
3	קישור לסרטון	דוח סופי .mp4

מידע נוסף:

סוג הפרויקט	1. מחקרי ממצא 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

1.....	נאום המעלית
2.....	מבוא
6.....	תיאור הבעיה
7.....	דרישות ואפיון הבעיה
7.....	הבעיה מבחינת הנדסת תוכנה
8.....	סקירת עבודות דומות בספרות והשוואה
9.....	תיאור הפתרון
10.....	המשך הפרק תיאור הפתרון
11.....	תוצאות ומסקנות
16.....	Abstract
17.....	נספחים

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 אופטימלית.

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.

וכו'.

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 מסוימת.

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

שייכים ל R^+

המטרה

C_{max} : מזעור הזמן מתחילת המשימה הראשונה, ועד סיום המשימה האחרונה

$\sum_{j=0}^P C_j$ * : מזעור סכום זמני הריצה אילו היה ברשותנו רק מעבד אחד

אנחנו נתמקד ב C_{max} .

כל הווריאציות ומידע עליהן, נמצא ב [1]

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 - ההסבר לכך הוא שיש כל כך הרבה וריאציות 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

במהלך הפרויקט ענינו על 3 שאלות מרכזיות:

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 קליטים\בעיות מהעולם האמיתי?

שאלה 3

מהו האלגוריתם הכי טוב לחומרה הספציפית של Mobileye?

דרישות ואפיון הבעיה

הדרישות של הלקוח (Mobileye)

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

התוצר שהגשנו ל-Mobileye כולל מספר אלגוריתמים שונים, שפותחו בשפת Python, קוד הסימולטור, תוצאות ההרצות ותובנות מההרצות בעקבות ניתוח נתוני ההרצה.

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.
- (5) שימוש בעקרונות פיתוח נכונים: OOP, פתיחות לשינויים ושדרוגים.

סקירת עבודות דומות בספרות והשוואה

(1) [The scheduling zoo](#)

זהו מאגר נרחב של קונפיגורציות מסוימות 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 ($P|prec|C_{max}$), עובדה זו מדגישה את המשמעות של פרויקט זה.

(2) [Svensson, Ola. "Hardness of precedence constrained scheduling on identical machines." SIAM Journal on Computing 40.5 \(2011\): 1258-1274.](#)

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

(3) [Graham, Ronald L. "Bounds for certain multiprocessing anomalies." Bell system technical journal 45.9 \(1966\): 1563-1581.](#)

בעבודתו, מציג Graham את המושג של תזמון משימות מאפס עם דוגמאות הרצה והגדרות מדויקות של מערכת עם קדימויות עם מעבדים זהים ($P|prec|C_{max}$). עבודתו של Graham עזרה 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.

(4) [Pinedo, M. L. \(2012\). Scheduling \(Vol. 29\). New York: Springer](#)

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

ניתחנו זמני ריצה, הוכחנו חסמים על ידי בניית (קונפיגורציות) שמצאנו ובעזרת ההיוריסטיקות ניסינו למטב את החמדן ה"טיפש".

שאלה 2

קיבלנו קלט מהעולם האמיתי ש Mobileye סיפקו לנו, פיתחנו אלגוריתמים שעשויים לייעל את זמן הריצה של החמדן על הקלטים שקיבלנו.

שאלה 3

נעזרנו באלגוריתמים שפיתחנו בחלק השני של הפרויקט (שאלה 2) על מנת לפתח אלגוריתם שיתאים לחומרה הקיימת במערכות של Mobileye.

לאחר מכן, פיתחנו 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.

ראשית פיתחנו פסאודו-קוד עבור סימולטור שיסמלץ הרצת מעבדים בסביבה הטרוגנית.

לאחר מכן היינו צריכים לחשוב איך נסמלץ מעבדים ומשימות, לשם כך יצרנו את המחלקות Task, Processor אשר מייצגות משימות ומעבדים, וכל המידע הדרוש על מנת להריץ את הפסאודו-קוד

איך השגנו דאטא?

הפרויקט הוא בשיתוף עם Mobileye, וכך נציג מ Mobileye סיפק לנו גישה להמון דאטא. עד שקיבלנו גישה לדאטא, עבדנו עם קבצים json. כשהדאטא הגיע פיתחנו את המחלקה Parser, שתפקידה לנתח את הקבצים הללו ולהפוך אותם לפורמט ה json שהשתמשנו בו עד כה.

בעקבות הדרישה לפיתוח אלגוריתמים חדשים, יצרנו מחלקה אבסטרקטית ששמה Algorithm כאשר כל אלגוריתם שנממש ירש ממנה. למחלקה הזו יש פונקציית decide, זהו לב המתזמן. פונקציה זו היא שקובעת איך המשימות תשובצנה.

לבסוף, פיתחנו את המחלקה Sim שקוראת את הדאטא, ולאחר מכן מריצה את הסימולציה כותבת את התוצאות לקובץ excel, ומציג אילוסטרציה לריצה בעזרת המחלקה TimeLineIllustration (ראו נספח 2).

תוצאות ומסקנות

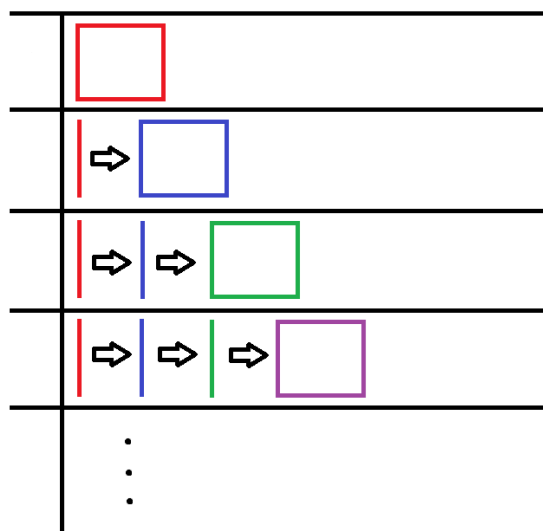
האם קיים חסם תחתון לאלגוריתם חמדן בסביבה הטרוגנית?

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, ולכן גם הבעיה שלנו היא NP קשה.

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

הוכח [2] שעבור בעיית תזמון מעבדים בסביבה הומוגנית (כאשר כל המעבדים מאותו סוג), לא ניתן לשפר את הקירוב של החמדן [3] (2-מקרב) בזמן פולינומי. בעזרת בניית דוגמה בה החמדן הוא לכל היותר P מקרב, מצאנו חסם תחתון עבור אלגוריתם חמדן בסביבה הטרוגנית:



משימה שזמן ריצתה
"אפס" (זניח)

משימה שזמן ריצתה
"1" (יחידת זמן אחת)

- יכול לרוץ על מעבד מסוג 1
- יכול לרוץ על מעבד מסוג 2
- יכול לרוץ על מעבד מסוג 3
- יכול לרוץ על מעבד מסוג 4

כפי שניתן לראות בשרטוט בכל שורה ישנן משימות כאשר כל משימה התלויה במשימה שלשמאלה, 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 עבור P סוגי מעבדים, כלומר P יחידות זמן, בעוד אלגוריתם אופטימאלי יריץ הכל ביחידת זמן אחת בלבד (ע"י הרצת כלל המשימות שזמןן 0 לפי סדר העמודות, וריצה במקביל של כלל המשימות בעלות זמן 1).

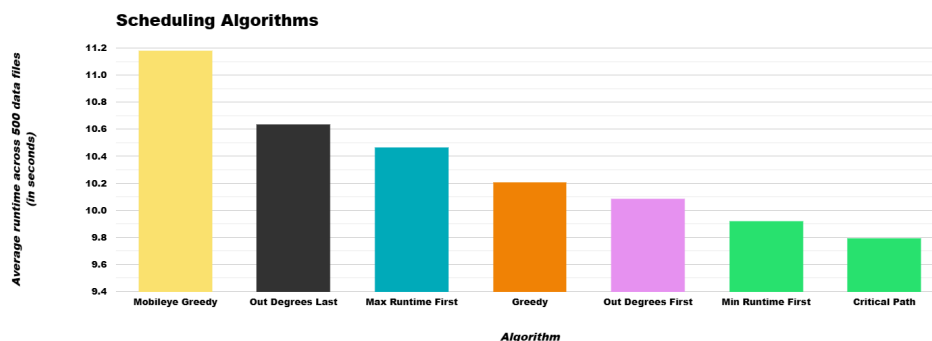
האם קיימים אלגוריתמים יעילים יותר מהחמדן עבוד בעיות תזמון מהעולם האמיתי?

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:



להלן השוואת האלגוריתמים ביחס לאלגוריתם החמדם המסורתי:

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

האם ניתן למצוא אלגוריתם online שמשפר את זמני הריצה?

ניסינו להתאים את היוריסטיקות שפיתחנו בשאלת המחקר 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 הפרויקט המרנו את ההיוריסטיקות שתזמנו בעזרת מיון כלל המשימות, 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 ידי קביעת ערכי סף לפי 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 Last	Max Runtime First	Min Runtime First	Greedy	Out Degrees First
Threshold	2	4791.6	4791.6	DC	2
Runtime % From Greedy	104.2%	101.3%	101.2%	100%	98.8%
Faster %	4%~	1.3~	1.2%~	0%	1.2%~
Standard Deviation	3.3%	2.2%	2.3%	0%	2.16%

במהלך העבודה על הפרויקט נתקלנו בקשיים מסוגים שונים.
ניסינו במשך זמן לא מבוטל להוכיח שאלגוריתם חמדן הוא גרוע יותר מ P 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 ה OOP.

קושי נוסף שהתמודדנו איתו הוא בדיקת הסימולטור שלנו.
כמובן שבדקנו את הסימולטור עבור מקרים פשוטים אך עבור DAG-ים עם מאות משימות הדבר הפך 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 ולתקן.

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 זמני הריצה במקרה הממוצע, אך הפעלת Clustering על ריצה של DAG וניסיון להתאימו ל DAG שכבר ניתחנו את האלגוריתם המהיר ביותר עבורו עשוי לשפר את זמן הריצה הממוצע.

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

אלגוריתם זה על אף היותו אלגוריתם offline הביא לשיפור ניכר (4.2% במקרה הממוצע ועד כ-10% במקרים מסוימים), לכן מעניין לנסות להמירו לאלגוריתם Online, ע"י ניבוי ה DAG הבא לפי הריצה של ה DAG הקודם וניצול מידע הקשור לחלקים קבועים הקיימים בכל DAG.

כמובן שיש לקחת בחשבון את זמן חישוב ה Critical 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 ב DAG).

בכל מקרה שינוי כזה ידרוש שינוי קל בחומרת המתזמן, לעבור מארכיטקטורת דחוף\לא דחוף לארכיטקטורה התומכת בהרצת משימות לפי סדר ידוע מראש.

Abstract

Mobileye is a leader in autonomous vehicle technology and advanced driver assistance systems, and they rely on complex algorithms that run simultaneously on multiple processors in a real-time framework. Because these algorithms are dynamic, their execution times cannot be determined ahead of time.

The scheduler, which is an integral part of the operating system, decides which jobs the processors perform at any given time. The greedy scheduler is often used for its efficiency and simplicity, but it often fails to account for future probabilities, increasing the run time

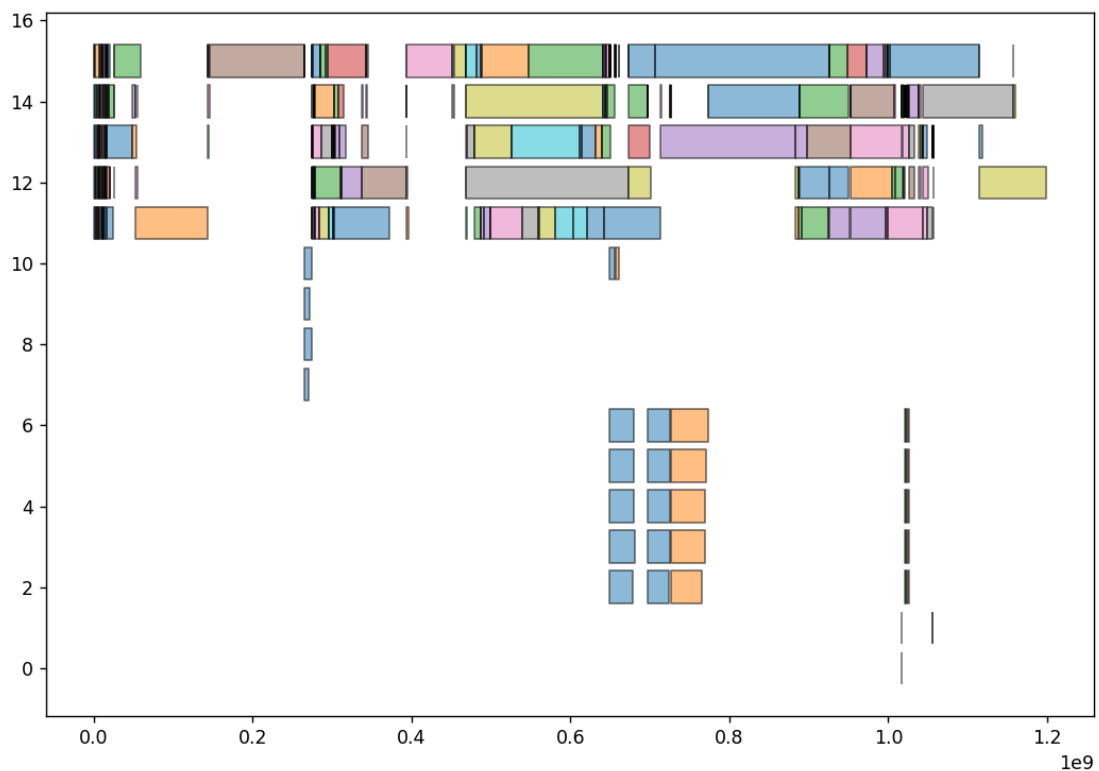
The goal of this program is to improve the execution time of algorithms that run tasks with different dependencies on multiple processors. The task runtime is not known ahead of time, and each task can only run on a specific type of processor. Our research and practical implementation investigate the use of different heuristics to effectively reduce the running time of the algorithm.

We used three main heuristics: Min-Runtime-First, Max-Out-Degrees, and Critical Time. Our analysis showed that the Critical Time heuristic performed better than others, reducing the run time significantly, while the Min-Runtime-First heuristic was the second best. These results show that productivity can be increased significantly by influencing the greedy planner to prioritize dependent tasks over a fixed urgency.

However, because Mobileye chooses to interrupt the scheduler's plan with their own priorities for different jobs, our attempts to match the heuristics to their hardware proved to be ineffective. As a result, our heuristics' total performance was below expectations on Mobileye's hardware.

נספחים

- [1] מאגר הקוד לסימולטור כולל תוצאות ההרצות: [repo](#)
[2] דוגמת הרצה מהסימולטור (אילוסטרציה של הסימולטור):



[3] דיאגרמת היחסים בין המחלקות והאובייקטים שפיתחנו עבור הסימולטור:

