גיל חנקין 315046805
גלעד עיני 034744920

דוח סיכום לקורס מימוש אלגוריתמים בתוכנה

INTRO

- התחלנו עם הבעיה של חלוקת משימות למכונות שמספרן תחום בין 2 ל 10 כשפונקציית המטרה היא מזעור מספר המכונות + המכונה הכבדה ביותר:

$$Minimize(|M| + Max(M))$$

$$where\ M\ is\ set\ of\ all\ Machines, and\ Max(M) is\ the\ heaviest\ machine(Sum\ of\ its\ job\ values)$$

- במהלך הסמסטר, בעיצומו של מימוש היוריסטיקה מספר 2 – חיפוש מקומי, הוחלט לשנות את הגדרת הבעיה לבעיית CMAX (קישור למאמר. קישור לבנצמארק). הבעיה היא עדיין חלוקת משימות למכונות אך מספר מכונות קבוע. פונקציית המטרה שונתה קלות:

$$Minimize(Max(M))$$

$$where\ M\ is\ set\ of\ all\ Machines, and\ Max(M) is\ the\ heaviest\ machine(Sum\ of\ its\ job\ values)$$

- הגשנו פרויקט שמכיל
  - Main.cpp - מכיל כל מיני פונקציות הקשורות בעיקר לקלטים (שליפה מקבצים, אתחול ידני וכל מיני מעטפות כדי לראות את הפלטים ...)
  - Class BNB (cpp + h)
  - Class LocalSearch (cpp + h)
  - Class Genetic (cpp + h)
  - Utils - מכיל פונקציות שמשמשות את הקלאס

BNB

- מטעמי תאימות לאחור, השארנו את מספר המכונות בתוך חישוב פונקציית המטרה והוספנו את אותו מספר המכונות לערכי פונקציית המטרה שלקחנו מבנצמארק.
- קלט נדרש לקלאס
  - רשימה של ערכי משימות
  - מספר מכונות

- מחשבים חסמים תחתונים גלובליים (לכולם הוספנו את מספר המכונות)
  - Perfect split
    - אם ניתן לחלק את המשימות בצורה מושלמת (בלי שארית חלוקה) הרי לא ניתן למצוא פתרון יותר טוב מזה
  - Pmax
    - זמן המכונה הכי כבדה הוא לפחות המשימה הכי כבדה
  - pigeonholePrinciple (שובך היונים)
    - אם יש n משימות ו m מכונות אז יש מכונה עם לפחות n/m המשימות הקלות.
  - לאחר חישוב שלושת החסמים, נבחר את המקסימלי שהוא האדוק ביותר.

- עבור כל משימה יוצרים NODE
  - מחשבים חסם תחתון לוקלי
    - ערך פונקציית המטרה הוא לפחות המכונה המקסימלית במצב הנוכחי של החלוקה של הNODE
    - שומרים את המקסימום (חסם תחתון לוקלי, חסם תחתון גלובלי) ב L (lower bound)
  - מחשבים חסם עליון לוקלי
    - משלימים את ההשמה החלקית הנוכחית ע"י LPT
    - מחשבים ערך פונקציית המטרה ושומרים ב U (upper bound)
  - בשלב זה ניתן לבדוק האם מצאנו פתרון יותר טוב מהפתרון המכהן
  - כעת יש בדיקה האם לקטום את תת העץ שהNODE הוא שורשו
    - האם החסם התחתון של ה NODE גדול או שווה לפתרון המכהן

- מצורף קובץ עם חמישה פלטים של BNB (BNBInputs)
- מסקנות:

במהלך מימוש ההיוריסטיקה של BNB הבנו את חשיבות אדיקות החסמים (תוכנה שהרצנו במשך יומיים ולדעתנו לא היתה קרובה לסיים).

הבנו את גודל העצים של משימות מהסוג הזה (עבור 30 משימות 10 מכונות גודל העץ המלא היה ~ 10^{30})

הבנו את החשיבות של האלגוריתם של החסם העליון, במקרה שלנו התחלנו עם משהו יותר פרמיטיבי וקיבלנו חסמים גרועים, ולאחר ששיניו ל LPTקיבלנו שיפורים משמעותיים בחסם העליון.

תחילה, לפני כל השיפורים שהכנסנו, BNB התקשה לסיים קלט באורך 30. בסופו של תהליך הוא פתר קלט באורך של 51 !

LOCAL SEARCH

- קלט למחלקה
  - רשימה של משימות
  - מספר מכונות
  - רשימה של אלגוריתמי הפתרונות ההתחלתיים
  - תוצאה אופטימלית מבנצמארק (השתמשנו בזה כחסם כשהרצנו מקבצי קלטים גדולים כדי לחסוך בזמן מפני שלעיתים האלגוריתם מצא את הפתרון האופטימלי והמשיך לרוץ כי הוא לא ידע שזה האופטימלי – פרמטר זה אינו נדרש והאלגוריתם יכול לעבוד בלעדיו)

- האלגוריתם הזה מחולק לשני שלבים:
  - מחשבים את כל הפתרונות ההתחלתיים שקיבלנו כפרמטר למחלקה:
    - LPT – כמו ב BNB
    - BESTFIT – משתמש בחסם תחתון לקבל ערך אופטמלי W לתא. ואז מנסה לחלק את המשימות כך שסכומם לא יעלה על W. כל מה שנותר לו ביד הוא יזרוק לתא הראשון.
    - SAMEMACHINE – השמה אשר המכונה הראשונה מכילה את כל המשימות
      - פתרון טרויאלי זה הציל את האלגוריתם כאשר השמה חמדנית (שני האלגורתמים הנ"ל) נתקעה
  - בדיקת שלושה חסמים תחתונים(אותם חסמים כמו באלגוריתם הקודם, BNB)
    - PMAX
    - שובך היונים
    - Perfect split
  - אם ערך פונקציית המטרה שווה לחסם התחתון המקסימאלי אין טעם להריץ את החלק השני של האלגוריתם, מכיוון שהפתרון הראשוני הינו אופטימאלי
  - אחרת עבור לשלב השני. השלב השני מכיל 5 צעדי חיפוש:
    - צעדי חיפוש(עם חשיבות לסדר):
      - הזז משימה 1 - צעד חיפוש זה הינו אופטימאלי (הצעד בודק את כל האפשרויות שניתן להגיע אליו באמצעות הזזה של משימה אחת ובוחר את הצעד הכי טוב- לא השיפור הראשון שנמצא!)
      - החלף משימה במשימה – צעד חיפוש זה הינו אופטימאלי
      - החלף משימה ב 2 משימות
      - הזז 2 משימות
      - החלף 2 משימות ב 2 משימות
  - הגדרה שיפור
    - אנחנו אומרים ששיפרנו אם אנחנו מוצאים מצב חדש עם ערך פונקציית מטרה נמוך יותר מהפתרון הקודם או ערך פונקציית מטרה זהה עם MSE קטן יותר.
    - Minimum Square Error – MSE מוגדר על ידי הנוסחא הבא:

$$\sqrt{\sum_{\forall m \in M} (\sum_{\forall j \in m} j)^2}$$

    - ערך פונקציית המטרה כמו ב INTRO.

- אם אנחנו לא מצליחים לשפר את פונקציית המטרה ולא מצליחים לעבור למצב עם פונקציית מטרה זהה ומכונות מאוזנות יותר האלגוריתם עוצר וההשמה האחרונה הינה ההשמה הטובה ביותר שהצלחנו למצוא.

- מצורף קובץ עם 123 פלטים של Local (LocalInputs)
- מסקנות:

במהלך מימוש ההיוריסטיקה של Local הבנו את חשיבות הפתרון ההתחלתי, התחלנו עם הפתרון ההתחלתי SAME MACHINE ולאחר מכן הוספנו את LPT ואת BESTFIT ששיפרו משמעותית את זמן הריצה ואת איכות הפתרון. למרבה ההפתעה, לאחר שהוספנו את שני האחרונים, הורדנו את SAMEMACHINE רק לגלות לאחר מכן שהפתרון הטריויאלי כן חשוב, והוא כן מצא פתרון אופטימאלי ש LPT ו BESTFIT הובילו את החיפוש המקומי למצב שהוא נתקע בלוקאל מקסימה.

התחלנו בהגדלת ראש ומימשנו צעדי חיפוש מאוד יקרים (החלפת 3 משימות ב 3 משימות אחרות), לאחר שעברנו לבעיה החדשה (Cmax), הבנו כאשר הקלט מכיל 25 מכונות, צעדים אלו גרמו לזמן ריצה מאוד מאוד ארוך.

חשיבות ה MSE: בגרסא הראשונית לא היו לנו את הכלים איך לצאת ממינימום לוקאלי, לאחר שיחה עם המרצה היא המליצה לנו להוסיף פונקציית עזר, MSE. פונקציה זו עזרה משמעותית להיחלץ ממינימום לוקאלי, גם אם צעדי לא הצליחו לשפר את פונקציית המטרה ה MSE הצליח "לאזן" את מצב המכונות.

Genetic

- קלט למחלקה
  - גודל האוכלוסיה
  - מספר הדורות
  - מספר המכונות
  - רשימת משימות
  - רמת דיבאג (כמה הדפסות – לא רלוונטי לאלגוריתם)
  - גנים מיוחדים (ערך בוליאני – מחליט אם ליצור גנים ראשוניים בדור הראשון מבוססי LPT ו BESTFIT)
    - החלטנו לא להשתמש בגנים אלו מכיוון שהם התקרבו לפתרון יותר מידי
  - אחוז המוטציות בכל דור
  - אחוז הביטים שישתנו במוטציה
  - פונקציית כשירות (יש 5 אופציות. נרחיב בהמשך)

- מחשבים חסמים תחתונים
  - PMAX
  - שובך היונים
  - Perfect split
    - לוקחים את המקסימלי

  - בכל שלב בריצה אם ערך פונקציית המטרה שווה לחסם התחתון המקסימאלי אפשר לעצור את האלגוריתם, מכיוון שהפתרון הינו אופטימאלי

- פונקציות הכשירות שמימשנו:
  - מרחק מערך פונקציית המטרה הכי גבוה בדור זה (ונרמול)– (לאחר בנצמארקינג משלנו – זו הפונקצייה בעלת התוצאות הכי טובות)
    - מכיוון שיש לנו פונקציה שצריך למזער, צעד זה גרם להיפוך האחוזים (גן עם ערך פונקציית מטרה נמוך, רחוק יותר מהערך הגבוה)

$$\frac{U - xi}{\sum_{j=1}^{\text{popSize}} U - xj}$$

  - מרחק מערך perfect split * c כאשר c=3 (ונרמול)

$$\frac{3ps - xi}{\sum_{j=1}^{\text{popSize}} 3ps - xj}$$

  - 1 חלקי ערך פונקציית המטרה של הגן (ונרמול)

$$\frac{1/tfi}{\sum_{j=1}^{\text{popSize}} 1/tfj}$$

- 1 חלקי שורש ערך פונקציית המטרה של הגן (ונרמול)

$$\frac{1/\sqrt{tfi}}{\sum_{j=1}^{\text{popSize}} 1/\sqrt{tfj}}$$

- 1 חלקי ריבוע ערך פונקציית המטרה של הגן (ונרמול)

$$\frac{1/tfi^2}{\sum_{j=1}^{\text{popSize}} 1/tfj^2}$$

- אנחנו מאתחלים את הדור הראשון שלנו בגנים(כרומוזומים) רנדומאלים לחלוטין.
  - מקבעים את הגן(כרומוזום) עם ערך פונקציית המטרה הכי טוב להיות ה Best gene found
- מתחילים את הריצה על מספר הדורות
  - יוצרים דור חדש
    - יוצרים מפת הסתברות(בעזרת פונקציית הכשירות שהועברה כפרמטר – הכי טובה הייתה מרחק מהמקסימלי – פונקציה מספר 1) עבור כל גן.
    - בוחרים X גנים למוטציה(מותר חזרות) ממפת ההסתברות (X הגיע כפרמטר – 0.05 היה הכי טוב שמצאנו). לכל גן שנבחר:
      - בוחרים Y אינדקסים שישתנו רנדומית (Y הועבר כפרמטר – 0.05 היה הכי טוב שמצאנו)
        - עבור כל אינקס שהוגרל: הגרל ערך בין 1-למספר המכונות – וזה הערך החדש.
    - בחר גודל אוכלוסיה פחות X להיות הורים לXO (מותר חזרות) בצורת זוגות ממפת ההסתברות
    - עבור כל זוג גנים שנבחרו ל XO:
      - הגרל שני אינקסים לחתכים ועושים XO לפי הגדרה. את שני הילדים שולחים ל NEXT GENERATION

- מצורף קובץ עם 6 פלטים עם רשימת משימות בגודל 2000(פעמיים), 3000, 3500, 4000, 10000 של Genetic (GeneticInputs)
- מסקנות:
  - בעיה ראשונה שנתקלנו בה זה היתה בעיית המינימיזציה, מכיוון שאנו מנסים לפתור בעיית מינימיזציה לא יכלנו להשתמש בדרך הטרואיאלית והפשוטה:

$$\frac{F(x)}{\sum F(y)}$$

מימשנו בסופו של דבר 5 פונקציות שונות.

- עקב ריבוי פרמרטרים באלגוריתם זה והמלצת המרצה עשינו באנצ'מארקינג משלנו ובחנו בו פרמטרים כגון:
  - גודל אוכלוסיה
  - מספר הדורות
  - אחוז המוטיציות בכל דור
  - אחוז השינוי בכל מוטציה
  - פונקציית כשירות
    - מצאנו שהפנקציית הכשירות הראשונה הינה הטובה ביותר
    - מספר דורות והאוכלוסיה הטובים ביותר היה 100,100
    - אחוז מוטציה ואחוז המוטציות בכל דור היה 5%,5%
- בסוף הכיול של הפרמטרים הרצנו משימות בגודל עצום ביחס ל Local & BNB והתרשמנו מאוד ש Genetic התקרב מאוד לאופטימום עד כדי טעות של חצי ממוצע המשימות, מלבד הקלט של ה 10000 משימות ושם הטעות הייתה מעט יותר גדולה. בגלל שהאלגוריתם הוא ליניארי, ניתן היה להריץ קלט בסדר גודל של 10000 משימות ולקבל תוצאה יחסית טובה בזמן קצר מאוד.