דו"ח עבודה "בית חכם"

האינטרנט של הדברים

JUNE 22, 2020

מאיה לביא | אור עזר
205781396| 205731524

אלגוריתמים:

ראשית תכנתנו את אלגוריתמי ה-DSA בהתאם לאלגוריתמים שהתבקשו במשימה.
להלן תיעוד הקוד ותוצאות הניסויים על 30 שכונות (כלל הניסויים המוצגים בדוח התבצעו על השכונות הללו – לפירוט על השכונות ראה נספח 1).

1. אלגוריתם 1: כל בית הוא סוכן

תיעוד קוד:

הוספנו משתני עזר אשר יעזרו לנו לשמור על הלוח זמנים המוצלח ביותר (מבין ה-10 איטרציות שנדרשו)

```
double bestGrade;//best grade of 10
double newGrade;//next grade we compare
double [] bestSched;//best shedule of 10
double[] randSched;//next schedule we compare
Map<PropertyWithData, Set<Integer>> bestSchedForAllProps;//map of prop schedules (best of 10)
```

הוספנו לולאה של 10 איטרציות בה מתקיימים חלקי קוד שנמצאים גם בקוד המקורי (מציאת השמה רנדומלית חדשה למכשירים, שיבוץ אותה בלוח הזמנים השכונתית ובדיקת התוצאות החדשות). לאחר מכן במידה והצלחנו לשפר את הציון באיטרציה הנוכחית בלולאה, נעדכן את כל ערכי ה-Best ונמשיך (בולם יאותחלו באיטרציה 0). ולבסוף נסיר את הלוח זמנים מלוחות הזמנים של השכונה.

```
for(int i=0; i<10; i++) {//10 random schedules for the props, pick the best one and save its' parameters
    randomSchedForAllProps.clean(); //**empty the previous schedule

    //for each property, pick a random set of ticks (satisfying all constraints)
    propToSubsetsMap.keySet().forEach(prop -> {
        Set<Integer> randSubset = pickRandomSubsetForProp(prop);
        randomSchedForAllProps.put(prop, randSubset);
    });
    randSched = helper.cloneArray(iterationPowerConsumption); //already with background load!
    randomSchedForAllProps.forEach((prop, ticks) -> {
        double powerCons = prop.getPowerConsumedInWork();
        ticks.forEach(tick -> randSched[tick] += powerCons);
    });
    allScheds.add(randSched);
    newGrade = calcImproveOptionGrade(randSched, allScheds); //the grade for this iteration

    if(i==0 || newGrade <= bestGrade) {//need to update bests (i==0 because in first iter we want to initialize)
        bestGrade = newGrade;
        bestSched = randSched;
        bestSchedForAllProps = randomSchedForAllProps;
    }

    allScheds.remove(randSched);
}
```

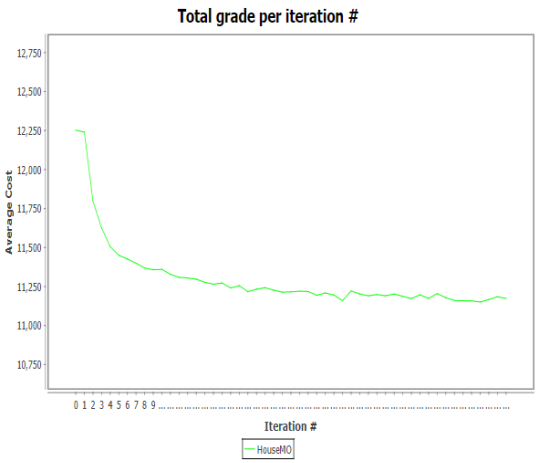
לבסוף הצבנו את הערכים המיטביים (Best) להיות אלו שיבוצעו והוספנו את הסתברות 0.7 לעבור לשיבוץ המשופר.

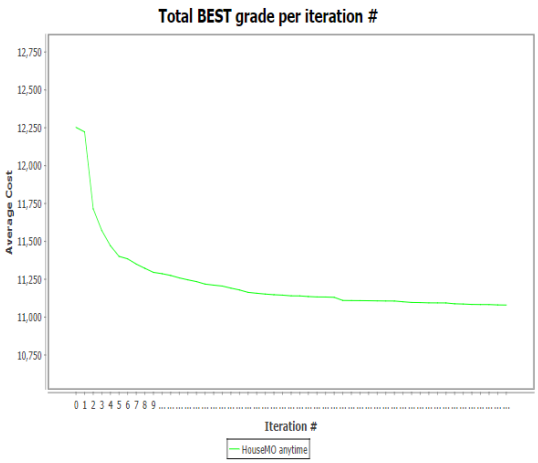
```
//return to original variables after finding best of 10
newGrade=bestGrade;
randSched = bestSched;
randomSchedForAllProps = bestSchedForAllProps;

//decide which of the 2 schedules to pick:
if (newGrade < prevGrade && flipCoin(0.7f)) { //pick the new schedule
    helper.totalPriceConsumption = newGrade;
    helper.ePeak = calculateEPeak(allScheds);
    randomSchedForAllProps.forEach((prop, ticks) ->
            updateTotals(prop,new ArrayList<>(ticks), propToSensorsToChargeMap.get(prop)));
}
else { //pick the previous schedule
    helper.totalPriceConsumption = prevGrade;
    allScheds.remove(randSched);
    allScheds.add(prevSched);
    helper.ePeak = calculateEPeak(allScheds);
    prevSchedForAllProps.forEach((prop, ticks) ->
            updateTotals(prop,new ArrayList<>(ticks), propToSensorsToChargeMap.get(prop)));
}
```

ביצועים:

ניתן לראות כי מגמת התוצאה היא ירידה, והעלות יורדת מתחת ל-11,250 לאחר 50 איטרציות.





2. אלגוריתם 2: כל בית הוא סוכן וכל מכשיר הוא סוכן גם כן

תיעוד קוד:

ראשית, ב-()improveSchedule שינינו את הלולאה כך שתבצע 100 איטרציות פנימיות אצל המכשירים:

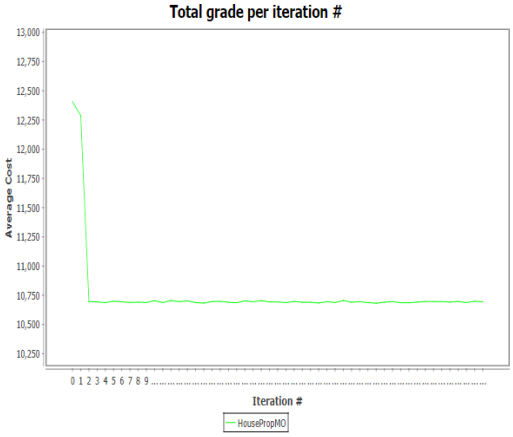
```
for(int i = 0; i < 100; i++){
    listOfPropsAgents.forEach( agent -> {//find schedule of ticks
        agent.chooseTicks();
    });
    listOfPropsAgents.forEach(agent -> {//change if improved sched + probability chose to change
        agent.changeTicks();
    });
}
```

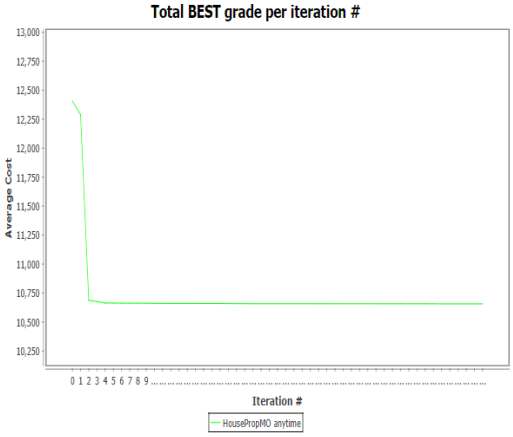
ב- PropAgent ב-()chooseTicks נוסף התנאי מעבר ללוח זמנים עם עלות שווה או קטנה בהסתברות 0.7:

```
if (newGrade <= oldGrade && flipCoin(0.7f)){
    this.toChange = true;
}
```

ביצועים:

ניתן לראות כי התוצאה יורדת ומתכנסת לאחר מעט מאוד איטרציות. העלות יורדת מתחת ל-10,750 לאחר 3 איטרציות ונשארת סביב ערך זה לאורך 50 האיטצריות.

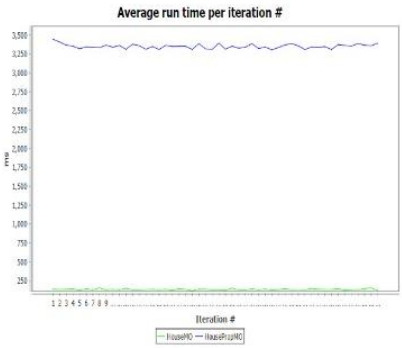




השוואת ביצועים:

ניתן לראות כי אלגוריתם 2 מניב תוצאות יותר נמוכות לאורך כל האיטרציות ומתכנס לערך לאחר מספר איטרציות ספורות בלבד. למרות זאת, זמן הריצה של אלגוריתם 2 ארוך (באופן יחסי אך עדיין מהיר מאוד) מאלגוריתם 1.

תוצאות אלה הן צפויות כיוון שאלגוריתם 2 מבצע 100 איטרציות של אלגוריתם DSA עבור **המכשירים לכל בית** ורק לאחר מכן מבצע אלגוריתם שכונתי. בכך, הוא מאפשר למכשירים 'לדבר' ביניהם ומאפשר שיפור לכל מכשיר. לכן, אלגוריתם 2 צפוי לצאת יותר זול מאלגוריתם 1 אשר לוקח את הלוח זמנים הטוב מבין 10 לוחות זמנים פיזיביליים אך רנדומליים עבור **כל המכשירים לכל בית** ולאחר מכן מנסה למזער את העלות השכונתית. כמובן שכל זה מסביר גם למה זמן הריצה שונה משמעותית ביניהם.

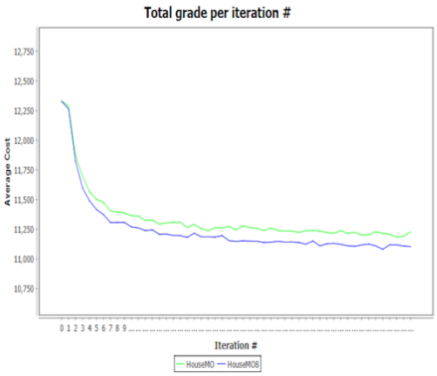


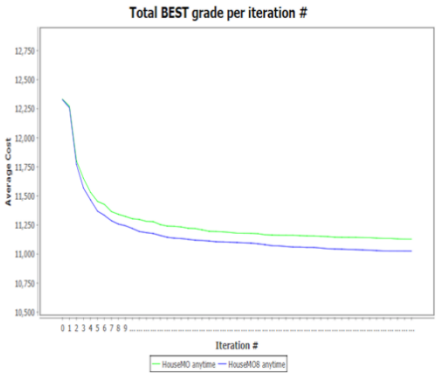
ניסויים לשיפור האלגוריתמים:

עבור 2 האלגוריתמים ביצענו מספר ניסויים- עבור שניהם, בדקנו האם הפתרון ישתפר אם נגדיל את הסתברות המעבר לפתרון אלטרנטיבי. בנוסף, בדקנו האם הגדלת האיטרציות הפנימיות (בין אם זה איטרציות של DSA או של חיפוש לוח זמנים רנדומלי). לאחר הבדיקות הבסיסיות הנ"ל, ניסינו עבור כל אלגוריתם לעודד exploration על ידי מעבר אקראי לפתרונות שונים בהסתברות נמוכה כלשהי. לא ידענו למה לצפות בשינוי ההסתברות כי כפי שנלמד בהרצאות, יתכן שאם כל הסוכנים יעברו אזי הפתרון יהפוך לגרוע יותר (אילוצים ימשיכו להתנגש), אך כיוון שלא הגדלנו בהרבה חשדנו שיתכן גם שיפור. שינוי האיטרציות יכול לשפר את הפתרון או להשאיר אותו כפי שהוא (כנראה לא לגרוע) ב-2 המקרים כיוון שדבר זה מאפשר למצוא פתרון עוד יותר טוב ממה שנמצא עד כה בכל שלב באלגוריתם (בין אם זה פתרון רנדומלי או לא). לא ידענו האם ה-exploration ישפר את הפתרון אך זה עלול לעודד פתרונות נוספים משופרים.

1. אלגוריתם 1: כל בית הוא סוכן

א. שינוי סיכוי מעבר לפתרון אלטרנטיבי:

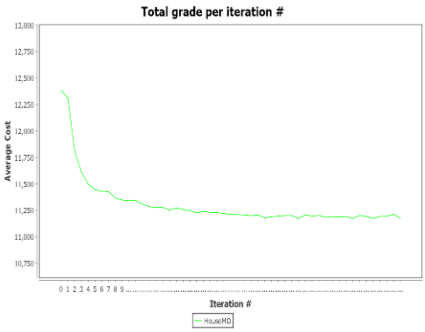




כפי שניתן לראות, העלות עבור האלגוריתם עם הסתברות מעבר 0.7 (קו ירוק) גבוה מהאלגוריתם עם הסתברות מעבר 0.8 (קו כחול). ההבדל אינו משמעותי ב50 איטרציות אך נראה כי המגמה ממשיכה ולכן שינוי זה משפר את הפתרון. (הקוד זהה לחלוטין חוץ משינוי ההסתברות)

ב. שינוי מספר החיפושים הרנדומליים עבור המכשירים:

במקרה זה ניתן לראות שהמגמות יצאו צמודות מאוד (נספח 2- גרפים והסבר על הקוד). נראה כי הגדלת מספר לוחות הזמנים הרנדומליים אינו משפר את הפתרון ולכן לא נמליץ על שינוי זה כיוון שהוא מאריך את זמן הריצה ללא תמורה ברורה.

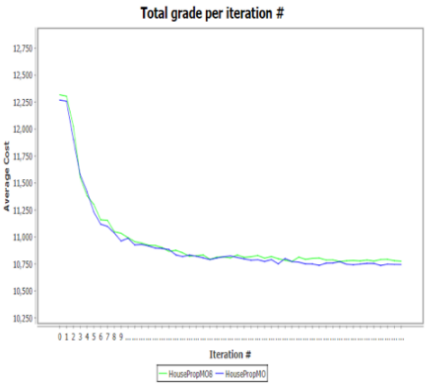


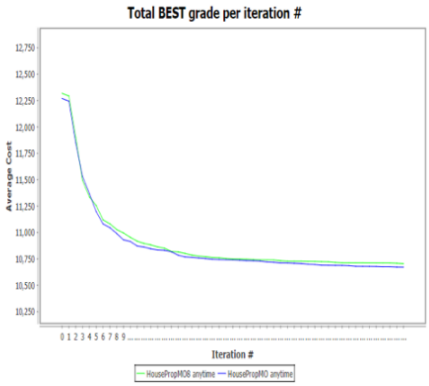
ג. עידוד exploration:

לאחר הוספת עידוד על ידי מעבר להשמה רנדומלית בהסתברות 0.1, קיבלנו תוצאות דומות מאוד לתוצאות המקוריות של האלגוריתם. נראה כי הערכים לא משתפרים משמעותית. בנוסף להסתברות זו ביצענו גם ניסויים עם הסתברויות נוספות (0.5, 0.15) בנוסף להגדלת מספר האיטרציות הפנימיות אך לא ראינו שיפור משמעותי באף ניסוי. (נספח 3 – הסבר על הקוד)

2. אלגוריתם 2: כל בית הוא סוכן וכל מכשיר הוא סוכן גם כן

א. שינוי סיכוי מעבר לפתרון אלטרנטיבי:





כפי שניתן לראות, העלויות אין משתנות באופן משמעותי לאחר 50 איטרציות של האלגוריתם. בכל זאת, נראה כי באיטרציות המאוחרות יותר, האלגוריתם עם הסתברות מעבר 0.7 (קו כחול) נמוך מהאלגוריתם עם הסתברות מעבר 0.8 (קו ירוק). לכן נחליט ששינוי זה אינו משמעותי ואינו משפר את הפתרון שממנו התחלנו. (הקוד זהה לחלוטין חוץ משינוי ההסתברות)

ב. שינוי מספר האיטרציות הפנימיות עבור המכשירים:

ערך הפתרון אינו משתנה משמעותית (נראה כי נשאר באזור 10,750) אך המגמה מתכנסת מאוחר יותר (נספח 4- גרפים והסבר קוד). מצד אחד, נעדיף אלגוריתם שמתכנס מהר, אך מצד שני ייתכן כי האלגוריתם עם יותר איטרציות ימשיך במגמת הירידה ולכן ישפר את הפתרון מעט מעבר למה שניתן לראות לאחר 50 איטרציות. לבסוף אין שיפור משמעותי בעלות הפתרון.

ג. עידוד exploration:

גם כאן, כפי שראינו באלגוריתם הראשון, הערך הסופי לא השתפר משמעותית לאורך הניסויים שלנו עם הסתברויות רנדומליזציה שונות ומספר איטרציות פנימיות שונות. ההסתברות שהניבה את התוצאות הטובות ביותר היא 0.15 , ללא שינוי במספר האיטרציות. תופעות שכן מצאנו הן שהסוכן הזול ביותר יצא זול יותר והסוכן היקר ביותר גם כן יצא זול יותר מאשר הסוכנים באלגוריתם המקורי, ונראה שזמן הריצה של האלגוריתם לרוב קצר יותר מהאלגוריתם המקורי. יתכן כי שימוש בו יקנה את התכונות האלו אך לא בהכרח ישפר את עלות הפתרון הסופי. (נספח 5- גרפים וקוד)

מסקנה סופית :

אנו ממליצים להשתמש באלגוריתם 2.ג בו מבוצע אלגוריתם DSA עבור המכשירים והבתים ובנוסף קיים עידוד לexploration.

נספחים :

נספח 1 פירוט הבעיות עליהן הרצנו את האלגוריתמים השונים:

חזרה לעמוד הראשי

בחרנו לקחת בעיות ברמת קושי מגוונות:

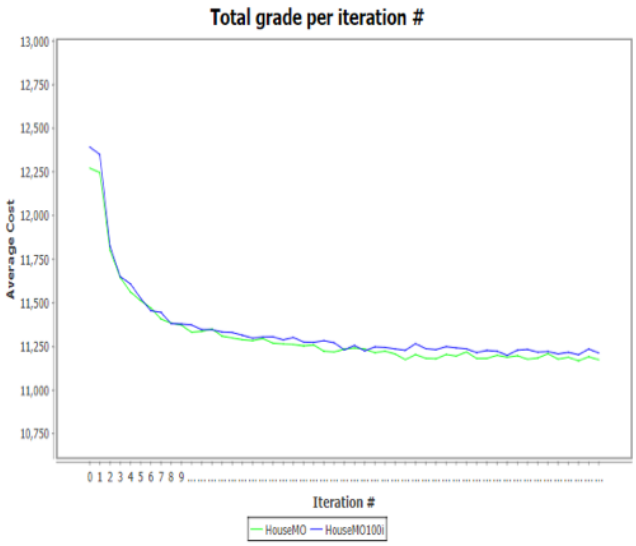
Small: 13,21,35
Medium: 37,40
Big: 112

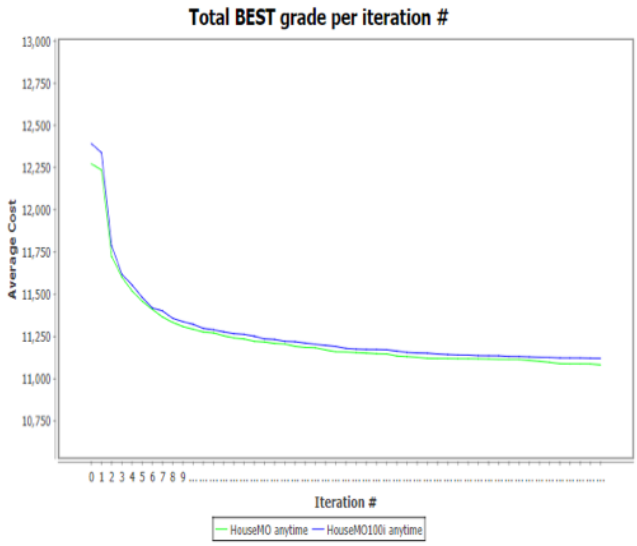
נספח House 2, שינוי מספר החיפושים הרנדומליים:

חזרה לעמוד הראשי

הסבר על הקוד:
הקוד זהה לחלוטין חוץ משינוי מספר החיפושים הרנדומליים.

גרפים :
HouseMO – האלגוריתם המקורי.
HouseMO100i – האלגוריתם המקורי עם שיפור מספר האיטרציות ל100.





נספח House 3, עם עידוד exploration:

חזרה לעמוד הראשי

הסבר על הקוד:
הוספנו תנאי מעבר להשמה רנדומלית בהסתברות 10% כך שגם אם ההשמה לא שיפרה את התוצאה, יהיה עוד exploration ודבר זה יאפשר להגיע לפתרונות חדשים עם האפשרות לשיפור התוצאה הסופית.

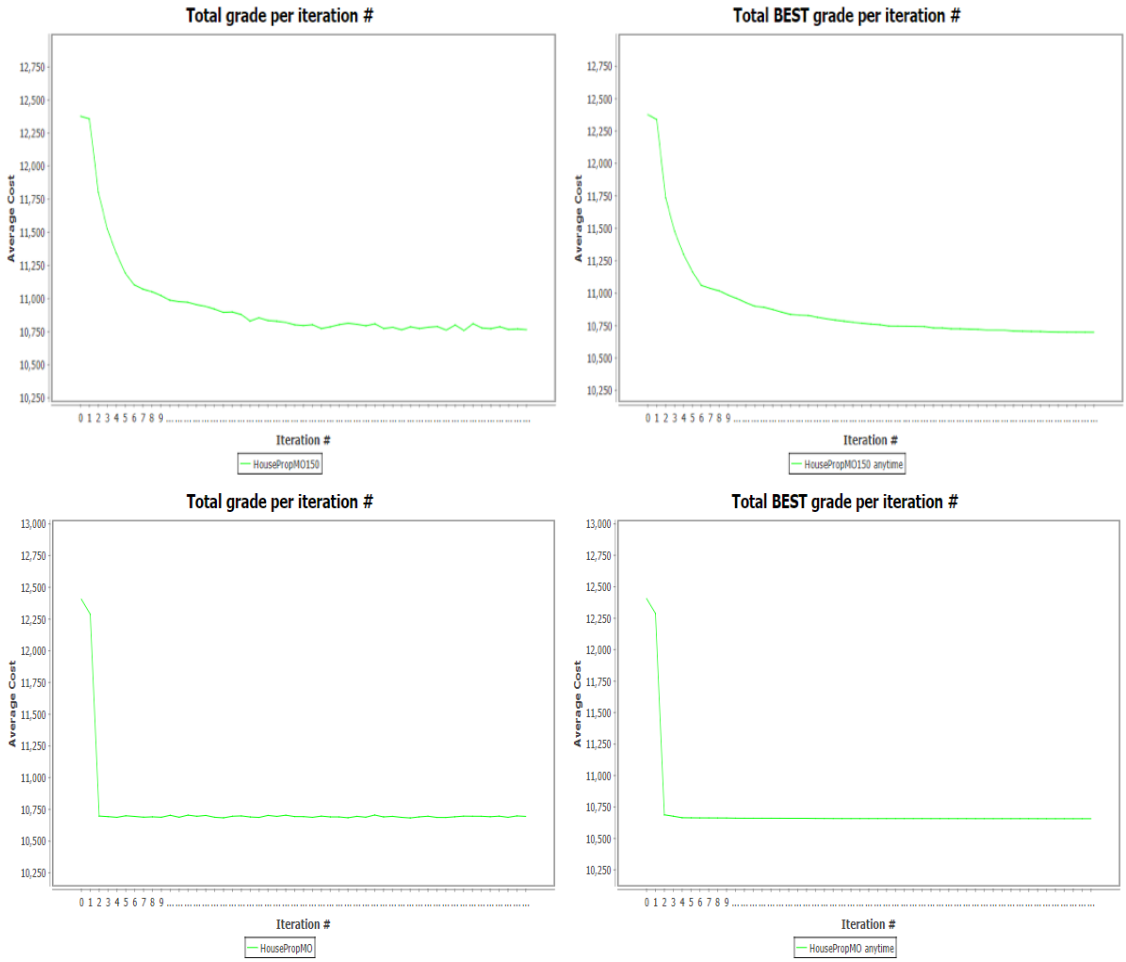
```
//decide which of the 2 schedules to pick:
if ((newGrade < prevGrade && flipCoin(0.7f))||flipCoin(0.1f)) { //pick the new schedule
    helper.totalPriceConsumption = newGrade;
    helper.ePeak = calculateEPeak(allScheds);
    bestSchedForAllProps.forEach((prop, ticks) ->
            updateTotals(prop,new ArrayList<>(ticks), propToSensorsToChargeMap.get(prop)));
}
```

נספח HouseProp 4 שינוי מספר האיטרציות הפנימיות:

חזרה לעמוד הראשי

הסבר על הקוד:
הקוד זהה לחלוטין חוץ משינוי מספר האיטרציות הפינמיות.

גרפים :
HousePropMO – האלגוריתם המקורי.
HousePropMO150 – האלגוריתם המקורי עם שיפור מספר האיטרציות ל150.



נספח 5 HouseProp עם עידוד exploration:

חזרה לעמוד הראשי

הסבר על הקוד:
הוספנו תנאי מעבר להשמה רנדומלית בהסתברות 15% כך שגם אם ההשמה לא שיפרה את התוצאה, יהיה עוד exploration ודבר זה יאפשר להגיע לפתרונות חדשים עם האפשרות לשיפור התוצאה הסופית.

```
if (flipCoin(0.15f)){
    this.toChange = true;
}
```

גרפים :
HousePropMO – האלגוריתם המקורי.
HousePropMOI2 – האלגוריתם המקורי עם עידוד exploration .
בגרף זמן הריצה נראה שהגרף הכחול לרוב מתחת לגרף הירוק למעט חריגים.