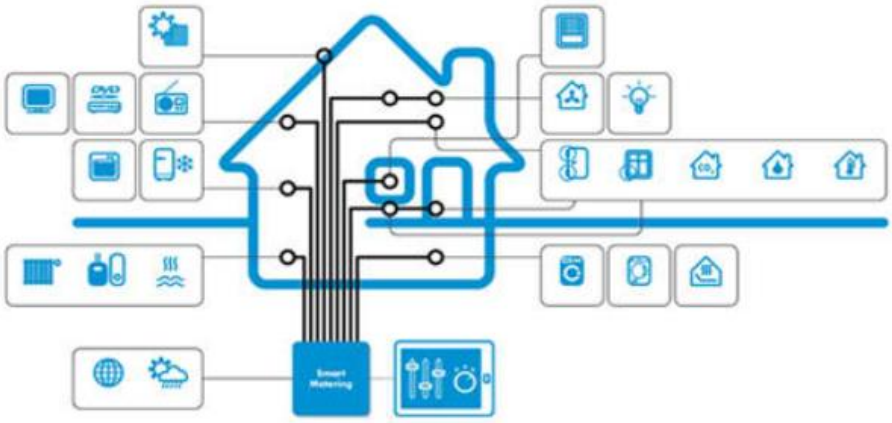
אוניברסיטת בן גוריון, קורס האינטרנט של הדברים

דוח תרגיל

"בית חכם"



אור נחשון אגש 311330500 | פלג נמדר 311124564

סמסטר ב, תשע"ב

1. מימוש אלגוריתם "כל בית סוכן" - House.java

בעבודה התבקשנו שכל סוכן (בית) ידגום 10 פתרונות רנדומליים אפשריים חדשים, ישווה בניהם וייקח את הפתרון הטובה ביותר. לשם כך הוספנו משתני עזר שיסייעו בחישובים ובשמירה על לוח הזמנים הטוב ביותר, להלן:

```
double bestGrade = 0; // best grade of 10 rand Schesds
double newGrade; // next grade we compare
double [] bestSched = null; // best shedule of 10
double [] randSched = null; // next scheduale we compre
Map<PropertyWithData, Set<Integer>> bestSchedProps =
        new HashMap<>(propToSubsetsMap.size()); // map of prop schedules
```

לקחנו את הקוד המקורי (שמוצא השמה רנדומלית חדשה למכשירים, משבץ אותה בלו"ז השכונתי ובודק את התוצאות החדשות) והוספנו לו לולאה שמבצעת זאת 10 פעמים. בסוף כל איטרציה שכזו בדקנו האם הציון השתפר, ובמידה וכן עדכנו את ערכי משתני העזר שצוינו לעיל והמשכנו לאיטרציה הבאה. לבסוף הסרנו את הלו"ז מלוחות הזמנים של השכונה.

```
for(int i=0; i<10; i++) {
    //10 random schedules for the props, chose the best one and save it's parameters
        randomSchedProps.clear(); //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);
        randomSchedProps.put(prop, randSubset);
    });
    randSched = helper.cloneArray(iterationPowerConsumption); //already with background load!
    randomSchedProps.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) {
            bestGrade = newGrade;
            bestSched = randSched;
            bestSchedProps = randomSchedProps;
    }

    allScheds.remove(randSched);
}
```

לבסוף בחנו האם ההשמה הטובה ביותר, מתוך העשר, טובה יותר מההשמה הנוכחית. במידה וכן הבית יעבור להשמה החדשה בהסתברות של 0.7; אחרת, הוא ישאר בהשמה הנוכחית שלו.

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

}
```

2. מימוש אלגוריתם "כל בית סוכן וכל מכשיר בבית סוכן" - HouseProp.java

התבקשנו לבצע 100 איטרציות פנימיות בתוך כל בית, כלומר 100 איטרציות בין המכשירים, ולכן הוספנו את הלולאה שבה כל מכשיר בוחר ומשנה השמות כפי שניתן לראות בצד.

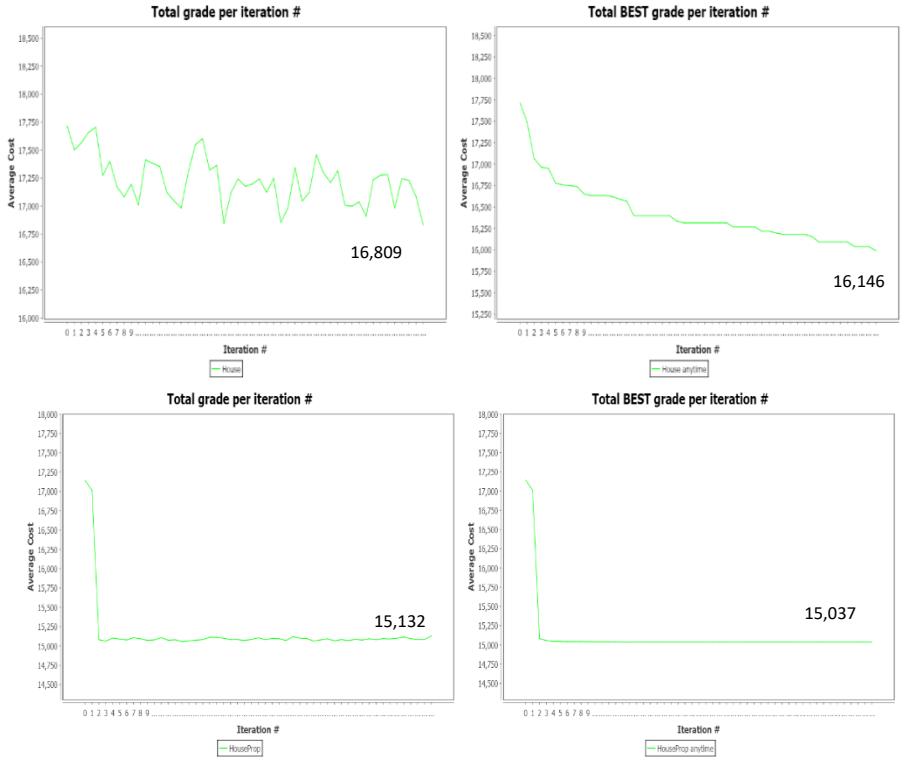
```
for(int i = 0; i < 100; i++){
    listOfPropsAgents.forEach( agent -> {
        agent.chooseTicks();
    });
    listOfPropsAgents.forEach(agent -> {
        agent.changeTicks();
    });
}
```

בנוסף בהוראות נדרשנו כי המכשירים הסוכנים יעברו להשמה החדשה בהסתברות 0.7 ולכן בסוכן מסוג PropAgent ערכנו את פונקציית chooseTicks() והוספנו את תנאי המעבר:

```
// decision: we compare both assignments using their grades
// you can use flipCoin(0.7f) as a boolean propability function
if (newGrade <= oldGrade && flipCoin(0.7f)){
    this.toChange = true;
}
```

3. השוואה בין שני המימושים ובחירת פתרון מועדף

ראשית נציין שבעקביות לאורך כל העבודה הקפדנו לבחון את כל האלגוריתמים על אותה קבוצת בעיות.



ניתן לראות כי שני האלגוריתמים לא Anytime ולא מונוטוניים אך עם זאת מגמת התוצאה היא ירידה. כפי שרואים בתוצאות **Prop עדיף** שכן המחיר הסופי שאליו הוא מתכנס נמוך יותר. Prop (אלגוריתם2) מניב ערך נמוך יותר בכל איטרציה ובאופן כללי הוא מתכנס מהר לאחר 2-3 איטרציות, לעומת House(אלגוריתם1) שאינו מתכנס. תוצאות אלו עמדו בציפיות שלנו מאחר שבאלגוריתם2 בכל בית המכשירים הם גם סוכנים ולכן כל מכשיר ובית יכול להגיע לתוצאה יותר טובה והעלות הגלובלית של כל השכונה קטן. זאת לעומת אלגוריתם 1 שבו אומנם נבחרת ההשמה הטובה ביותר מבין 10, אך עדיין זו השמה רנדומלית ולכן השיפור קטן יחסית. זמן הריצה של אלגוריתם 2 ארוך באופן יחסי (אך עדיין מהיר), ותוצאה זו צפויה שכן אלגוריתם זה מבצע 100 איטרציות DSA פנימיות, דבר שמוסיף לזמן הריצה יחסית לאלגוריתם1.

4. הצעות לשיפור האלגוריתמים

לאחר מימוש האלגוריתמים כפי שהתבקשנו, רצינו לשפר את הפתרון מעבר לכך ולכן בחנו מספר שינויים שסברנו שיכולים לשפר את האלגוריתמים:

א. הגדלת מספר האיטרציות ב-House- במצבו הנוכחי האלגוריתם עושה מעט איטרציות (50) ולא מתכנס ולכן בחנו מה קורה כאשר מגדילים את מספר האיטרציות ל-500. נמצא כי קיים שיפור אך גם במקרה זה האלגוריתם עדיין לא מתכנס ולכן החלטנו לבחון הגדלה למספר איטרציות גדול משמעותי של 2000 איטרציות. להלן התוצאות הסופיות:

IterationsNum	Best avg final grade	Total avg final grade
50 (מקור)	16,146	16,809
500	13,780	15,043
2000	13,487	15,013

ניתן לראות שהגדלת כמות האיטרציות ל-2000 אכן שיפרה את הפתרון אך לא בצורה משמעותית מדיי ביחס ל-500 המצדיקה את זמן הריצה הגדול משמעותית. מהתבוננות בתוצאות אנו מניחים שהגדלה למספר גדול יותר מ-2000 לא בהכרח תביא להתכנסות וזאת כנראה לאור העובדה שמדובר באלגוריתם לא מתוחכם מדיי ולכן לא המשכנו לבחון מספרים עולים. נמליץ להשתמש באלגוריתם זה עם מספר איטרציות 500 כך שהפתרון ישתפר אך עם זאת זמן הריצה יהיה סביר.

ב. הגדלת האיטרציות הפנימיות- אנו סבורים כי שינוי זה עשוי לשפר את הפתרון (או לכל הפחות להשאיר אותו כפי שהוא) כיוון שהוא מגדיל את הסיכוי לאתר פתרון טוב יותר.

House- הגדלנו את כמות לוחות הזמנים המוגרלים בכל איטרציה ל-20 (מצב נוכחי-10).

Prop- הגדלנו את כמות האיטרציות הפנימיות ל-150 (מצב נוכחי- 100).

להלן סיכום התוצאות שהתקבלו: (לצפייה בגרפים לחץ כאן)

	Best avg final grade		Total avg final grade	
	House	Prop	House	Prop
מקור	16,146	15,037	16,809	15,132
הצעה לשינוי	14,439	13,733	15,397	13,766
סה"כ שיפור	1,707	1,304	1,412	1,366

מהתוצאות ניתן לראות כי בשני האלגוריתמים יש שיפור ביחס לאלגוריתמי המקור שכן הפתרון הסופי טוב יותר. גם הפתרון המינימלי טוב יותר בשניהם, ולכן ניתן לומר שהשיפור שהוצע איפשר לאלגוריתמים למצוא פתרונות חדשים טובים יותר שקודם הם לא הגיעו אליהם במרחב החיפוש.

עם זאת חשוב לזכור שהגדלת כמות האיטרציות מגדילה את זמן הריצה. מכיוון שזמן הריצה עדיין סביר והשיפורים כן מטיבים עם הפתרון הגלובלי אנו בעד השיפורים שהוצעו.

ג. עידוד exploration- בצורה זאת אנחנו עשויים לעודד פתרונות נוספים משופרים. עשינו זאת ע"י שינוי הסיכוי למעבר להשמה טובה יותר מ-0.7 ל-0.75. לא ידענו למה לצפות בשינוי ההסתברות כי סיכוי גבוה מדיי מוביל לכך שיותר מדיי סוכנים מחליפים השמה בו זמנית ואז הפתרון הופך לגרוע יותר. כמו כן כפי שלמדנו בהרצאה הטווח שבו מתרחש כבר יותר מדיי exploration הוא סביב הנקודה 0.8, ולכן בחרנו p גבוה יותר מ-0.7 אך עם זאת עם סטייה מטה יחסית ל-0.8.

התקבל כי: (לצפייה בגרפים לחץ כאן)

	Best avg final grade		Total avg final grade	
	House	Prop	House	Prop
מקור	16,146	15,037	16,809	15,132
הצעה לשינוי	14413	13,692	15,625	13,750
סה"כ שיפור	1,733	1,345	1,184	1,382

מהתוצאות אנו רואים שגם הצעה זו משפרת את 2 תוצאות האלגוריתמים בדומה להצעה של הגדלת האיטרציות הפנימיות.

ד. עידוד exploitation- רצינו לבחון גם מה קורה כאשר דווקא אנחנו מקטינים את p להיות 0.6 ובכך אנחנו מפחיתים exploration ומגדילים את ה exploitation. התוצאות שהתקבלו הן: (לצפייה בגרפים לחץ כאן)

	Best avg final grade		Total avg final grade	
	House	Prop	House	Prop
מקור	16,146	15,037	16,809	15,132
הצעה לשינוי	15797	16,618	17054	16,646
סה"כ שיפור	349	-1,581	-245	-1,514

ראשית, לגבי אלגוריתם prop ניתן לראות שהוא לא משתפר. אנו סבורים כי הסיבה לכך היא כי הורדת ההסתברות למעבר להשמה גרמה להתכנסות למינימום מקומי (העובדה הזאת גם עומדת בקנה אחד עם מה שנלמד בכיתה שהסתברות נמוכה מדיי פוגעת בפתרון הגלובלי). לגבי House ניתן לראות כי במדד ה-anytime שמייצג את התוצאה הטובה ביותר ניתן לראות שיפור קטן, ובמדד התוצאה הסופית הממוצעת הוא אינו משתפר. אנו חושבים שהסיבה לכך שהתוצאות גבוליות היא בגלל שבאלגוריתם יש הרבה רנדומליות עם הגרלת ההשמות למכשירים ולכן מטבעו הוא עושה הרבה exploration ולכן העובדה שהורדנו את p לא פוגעת בו כל כך, והוא מצליח להתחמק ממינימום מקומי.

5. בחירת פתרון מועדף סופי

בכדי לאזן את צריכת החשמל של השכונה בצורה הטובה ביותר נמליץ להשתמש באלגוריתם **PropHouse** בשילוב עם השיפורים של הגדלת האיטרציות הפנימיות (הצעה ב) ועידוד exploration (הצעה ג).

6. נספחים

6.1. קבוצת הבעיות

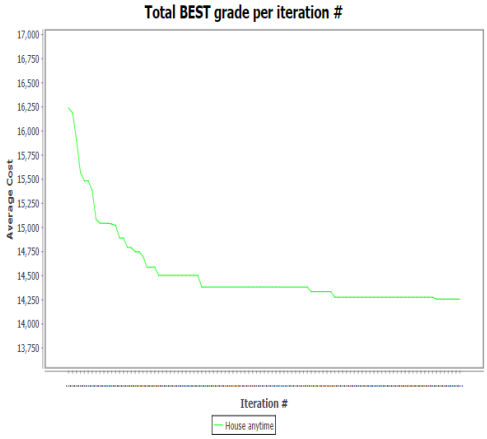
לאורך כל הבדיקות של האלגוריתמים הקפדנו בעקביות לבחון את כל האלגוריתמים על אותה קבוצת בעיות, להלן:

13-1-4, 21-1-3, 35-1-6, 40-1-5, 71-1-3, 112-1-2, 112-1-4

בחרנו אותן כך שיהיה מגוון מבחינת גדלים (SMALL, MEDIUM, BIG). כמו כן בחרנו 7 בעיות כדי שמצד אחד יהיה מדגם מייצג ונוכל לבחון מגמות ולא רק על סמך בעיה בודדת, ומצד שני לא בחרנו יותר מדיי בעיות כדי להקל על זמן הריצה.

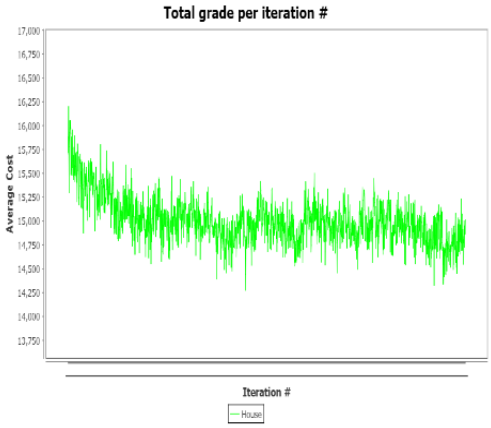
חזרה לדו"ח

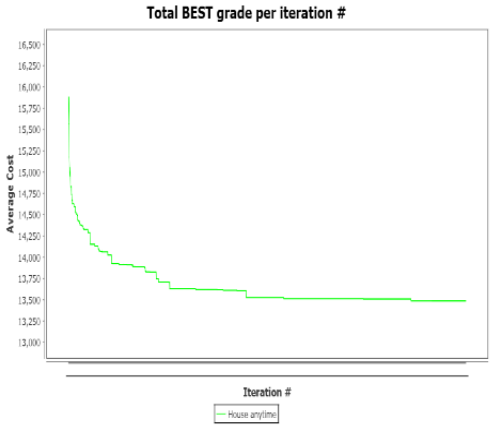
6.2. הגדלת מספר האיטרציות ל-500



חזרה לד"וח

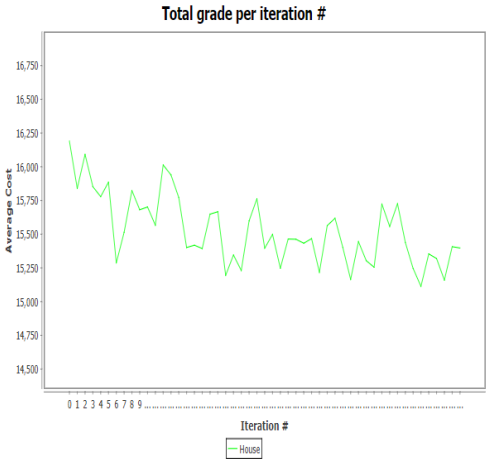
6.3. הגדלת מספר האיטרציות ל-2000

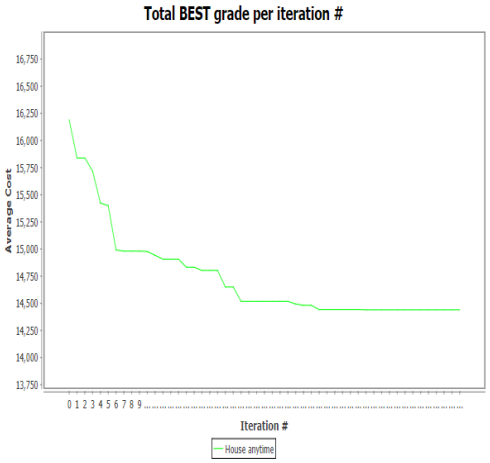


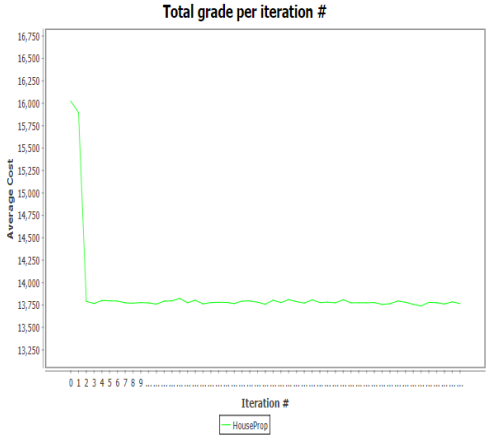


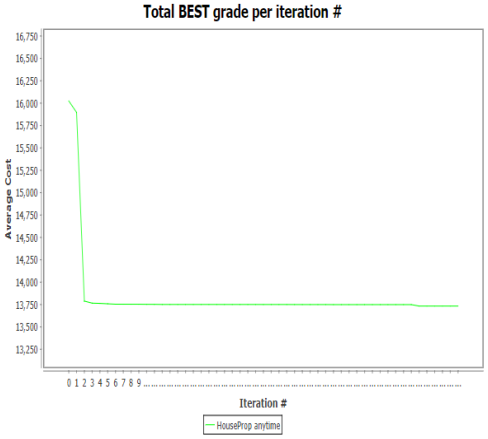
חזרה לדו"ח

6.4. הגדלת מספר האיטרציות הפנימיות



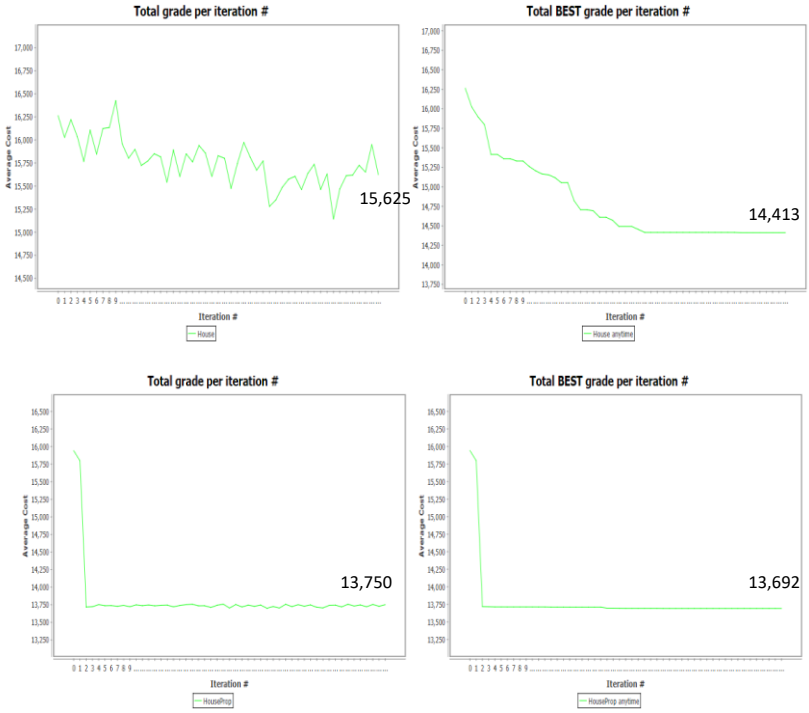






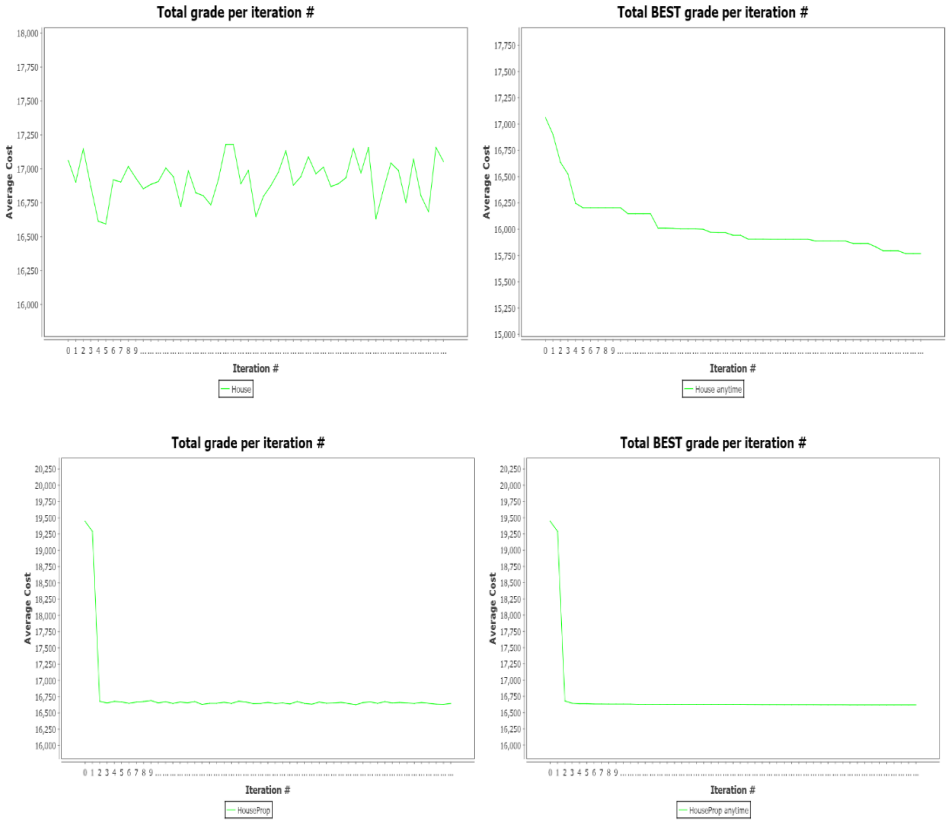
חזרה לדו"ח

6.5. עידוד exploration שינוי p=0.75



חזרה לדו"ח

6.6. עידוד exploitation



חזרה לדו"ח