פרוייקט בבינה מלאכותית - הרחבה

תומר לוי stlevy@t2.technion.ac.il

29 ביולי 2015

נושא: **הרחבת הפרוייקט:**

שימוש בחיפוש לוקלי למשחקי Tower Defense

מנחה: פרופ' שאול מרקוביץ

תוכן עניינים

1 מבוא

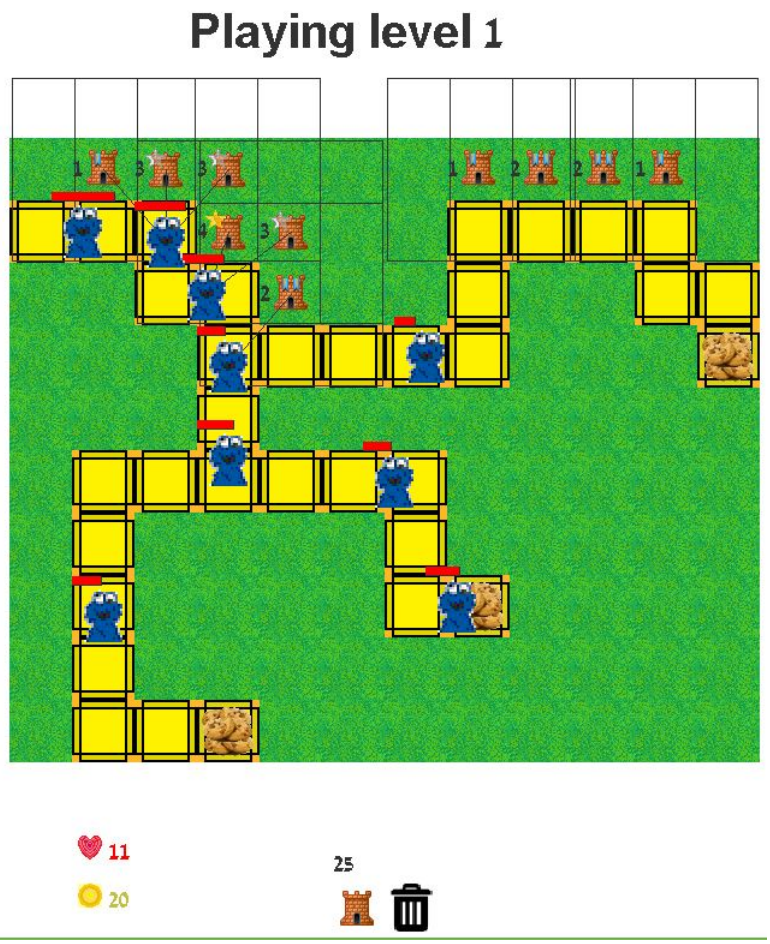
1.1 ז'אנר Tower Defense - תזכורת

Tower Defense (או TD) הוא ז'אנר משחקי מחשב מסוג אסטרטגיה בזמן אמת. מטרת המשחק היא לנסות לעצור את מעברן של דמויות עוינות מצד אחד של מפת המשחק למשנהו על ידי בניית מגדלים שירו בדמויות ... הריגת דמות אויב מביאה להרווחת כסף או נקודות, אשר בתורם משמשים את המשתמש בקניית מגדלים נוספים, או בשדרוג הקיימים. **הבחירה בסוג המגדלים ומיקומם במפה היא חלק מהאסטרטגיה החיונית במשחק. ... [[מתוך ויקיפדיה]]**

1.2 חוקי משחק ה TD בפרוייקט זה.

החוק שמודגש באדום הוא החוק היחיד שהשתנה מהפרוייקט הקודם. ובעצם מהווה את הרחבת הבעיה.

- השחקן:
  - השחקן מתחיל עם 20 נקודות חיים, ו 250 נקודות כסף. (בציור 1.1 : הלב האדום והמטבע הזהוב)
  - השחקן יכול לבנות מגדלים במחיר אחיד של 25 לאחד, ולמקם אותם בכל מקום שאין בו שביל או מגדל.
- המפה בגודל 12×10:
  - מורכבת מ־2 סוגי משבצות : שביל, ואדמה.
  - בסוף השביל יש מטרה אליה צועדות המפלצות.
  - **למפה יש נקודת כניסה אחת אבל נקודות יציאה מרובות.**
- המפלצות:
  - המפלצות הולכות על השביל הצהוב
  - בהגיעה לסוף המסלול מורידה חיים אחד לשחקן, ונעלמת מהמשחק.
- המגדלים:
  - המגדלים יורים במפלצות, ומורידים להם חיים.
  - רמת מגדל היא מספר המגדלים השכנים לו. (מדובר על שכנות־8 , כלומר גם אלכסונים)
  - אם מגדל כבר מכוון על מפלצת הוא ימשיך לירות עליה
  - אלגוריתם כיוון: הוא בוחר את המפלצת עם הכי מעט חיים (מיון ראשוני) שנמצאת כמה שיותר רחוק (מיון משני).
- מטרת השחקן היא למנוע מהמפלצות להגיע לנקודת הסיום ע"י קנית מגדלים
  - השחקן מפסיד אם החיים מגיעים לאפס.
  - השחקן מנצח אם הרג את כל המפלצות.



ציור 1.1 - דוגמא למשחק TD בזמן ריצה

לדוגמא בציור 1.1 אפשר לראות את משחק ה־TD.

1.3 הבעיה אותה אנחנו מנסים לפתור - subset selection

המטרה, היא לעבור את כל שלבי המשחק , כאשר כמה שפחות מפלצות הצליחו להגיע למשבצת המטרה.
במפה יש 120 משבצות, מתוכן בין 60 ל 110 משבצות שניתן לבנות בהן שבילים. בתחילת כל שלב, השחקן צריך לבנות 10 מגדלים.
השחקן צריך להחליט מה המיקום הטוב ביותר למקם את המגדלים, עם התחשבות באינטרציה עם השביל ובין המגדלים לבין עצמם.
יש כ $\binom{110}{10} - \binom{60}{10} \approx 5 \times 10^{13} - 7 \times 10^{10}$ אפשרויות. בהמשך במשחק, כל פעם שהשחקן מרוויח מספיק כסף, הוא יקנה מגדל אחד נוסף (פה מרחב החיפוש, כמובן, קטן מאוד)

2 תיאור הפתרון שנבחר לבעיה

כאמור, הבעיה היא מיקום k (בתחילת המשחק, 10) מגדלים בתוך כ־100 משבצות, כך שהאפקטיביות של המגדלים תהיה מקסימלית.

2.1 גרף המצבים - לא השתנה מהמשחק המקורי

- **צמתים:** כל צומת בגרף המצבים הוא k־יה של נקודות בלוח (x,y) פורמלית:

$$NODES = \left\{ (n,m)^k \mid n \in [0, gameWidth] \wedge m \in [0, gameHeight] \wedge legal\left((n,m)^k\right) \right\}$$

כאשר legal הוא פרדיקט שמוודא כי לא קיים אף זוג פעמיים ב־k־יה, ויש אדמה ריקה במפה בכל אחד מהזוגות:

$$legal\left((n,m)^k\right) = (\forall_{i,j}\,(n_i,m_i) \neq (n_j,m_j)) \wedge (\forall_i free\,(n_i,m_i))$$

כאשר free(n,m) הוא פרדיקט שמוודא שבקורדינטה n,m האדמה ריקה (כלומר אין שביל).

- **צומת התחלתי:** נבחר באופן אקראי k נקודות חוקיות למיקום מגדלים ללא חפיפות.
  - **צומת סופי:** אין, זוהי בעיית אופטימיזציה.
- **אופרטורים:**
  - כל אופרטור לוקח את אחת מ־k הנקודות, ומזיז אותה למיקום חוקי אחר במפה, בלי לשנות את שאר הנקודות.
  - נגדיר את קבוצת האופרטורים החוקיים מצומת לפי קבוצת ה־successors שלו, פורמלית :

$$\begin{aligned} succ\left(\{(n_0,m_0),(n_1,m_1)\ldots(n_{k-1},m_{k-1})\}\right) \quad = \quad & \{\{(n'_0,m'_0),(n'_1,m'_1)\ldots(n'_{k-1},m'_{k-1})\} \mid \\ & \exists i.\,(n_i,m_i) \neq (n'_i,m'_i) \\ & \wedge \forall j \neq i.\,(n_i,m_i) = (n'_i,m'_i) \\ & \wedge legal\left((n',m')^k\right)\} \end{aligned}$$

  - לפי הגדרה זו מקדם הסיעוף בחיפוש הראשוני במשחק הוא כמה מאות צמתים

2.2 אלגוריתם החיפוש - לא השתנה מהמשחק המקורי

2.2.1 תכונות אלגוריתם החיפוש

1. חיפוש לוקאלי
2. חיפוש אלומה - כלומר חסום זכרון
3. אלגוריתם anytime - כלומר מחזיר תוצאה בכל זמן שעוצרים אותו.
4. אלגוריתם אופטימיזציה - כלומר אין צומת מטרה, מחפשים את הצומת בעל הערך היוריסטי **הגדול ביותר**
5. $SAHC$- כלומר מתקדם דרך הגרדיאנט החזק ביותר

2.2.2 סיבות לבחירת האלגוריתם

כמו הרבה בעיות בתחום, השיקולים המרכזיים הם זמן וזיכרון, כאשר הגורם שמשפיע עליהם הוא מרחב המצבים המאוד גדול. ובכל זאת קיים מגוון רחב של אלגוריתמים שמטפלים במרחבים גדולים, להלן מספר סיבות לבחירה הספציפית שלי:

1. **חיפוש לוקאלי** :

   (א) מתוך המרחב הגדול, רוב המצבים שווים מעט.

   (ב) מצבים קרובים בגרף בדר"כ יהיו בעלי ערך יוריסטי קרוב, מאופי המשחק וההגדרה של הגרף.

   ולכן, מאופי המצב ההתחלתי (כפי שתואר ב 2.1) חיפוש לוקאלי יכול לחסוך מעבר על המון צמתים לא רלוונטים ובסבירות גבוהה להגיע לצמתים שהערך היוריסטי שלהם גבוה.

2. **חיפוש אלומה:**

   (א) בעצם שיפור ל SAHC , שעוזר להתחמק ממקסימות מקומיות

2.2.3 פסאודו קוד

```
beamSearch(root,width):
    best = [],current = [root]
    done = false
    while (not stopped) and (not done):
        done = true
        for node in current:
            if tryToAddToBeam(best,node,width)
              done = false
        current = []
        for node in best:
            for child in succ(node):
              tryToAddToBeam(current,child,width)
    best.sortDecending()
    return best[0]
```

ניתן לראות כי האלגוריתם משתמש בכמות זכרון חסומה ע"י $O(width)$
בכל שלב האלגוריתם מפתח את כל הבנים של האלומה הטובה ביותר עד כה, ובוחר את האלומה הטובה מבינהם,
ואז, הוא בוחר את הצמתים הטובים ביותר משתי האלומות ושם אותם באלומה הטובה ביותר.
אם האלגוריתם לא הצליח להכניס אף צומת חדש לאלומה הטובה ביותר, האלגוריתם הגיע למקסימום מקומי ועוצר.

```
tryToAddToBeam(beam,node,width):
    if beam.size() < width:
        current.add(node)
        return True
    beam.sortAscending()
    if Better(node,beam[0]):
        beam.remove(0)
        beam.add(node)
        return True
    return False
```

אלגוריתם הכנסה לאלומה :
אם האלומה לא מלאה הכנס,
אחרת, בדוק את האיבר עם הערך היוריסטי הנמוך ביותר, אם הוא נמוך מהערך היוריסטי של הצומת שרוצים להכניס תחליף ביניהם.

2.3 פיצ'רים והיוריסטיקות- האלמנט המרכזי שהשתנה מהפרוייקט המקורי

2.3.1 פיצ'ר השביל -נשארה זהה, אך נוספו חסרונות

מוטיבציה: בסופו של דבר, צריך שכמה שיותר מגדלים יוכלו לפגוע במפלצות לכמה שיותר זמן, כלומר, נעדיף שמגדל ירה 100% מהזמן שלו, מאשר מגדל שירה 50% מהזמן שלו, וזאת כיוון שאין משמעות במשחק ל"מנוחה".
לפי הגיון זה, ככל שמגדל יראה יותר שביל, הוא יהיה יותר אפקטיבי במשחק, כלומר, פיצ'ר זה ממקסם את את כמות השביל שרואים כל המגדלים במשחק.

הגדרה פורמלית : יהי $road\,(x,y)$ פרדיקט שמחזיר 1 אם הנקודה היא שביל ו 0 אם לא.
תהי $neighboringRoads\,(x,y)$ פונקציה שמחזירה לכל משבצת את כמות השכנים שמקיימים את $road$

$$neighboringRoads\,(x,y) = \sum_{-1\le i,j\le 1} road\,(x+i,y+j)$$

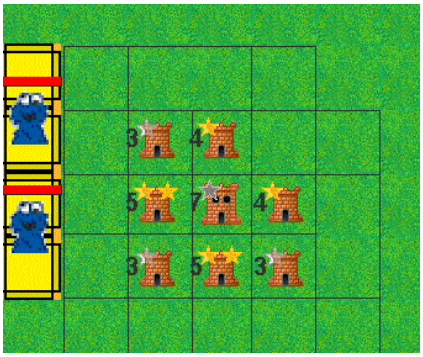
אנו מתעלמים מ $road\,(x,y)$ כיוון שאנו תמיד נקנה מגדל במקום חוקי, ולכן בהכרח $road\,(x,y)=0$.
לבסוף תהי $TOWERS$ קבוצת מיקומי המגדלים במצב נתון, נגדיר את פיצ'ר השביל f_{ROAD}:

$$f_{ROAD}\,(NODE) = \sum_{(x,y)\in TOWERS} neighboringRoads\,(x,y)$$

חסרונות : זהו פיצ'ר בעייתי כיוון שהמגדלים ברמה 0 חלשים, וקשה להם לפגוע בצורה משמעותית במפלצות, ולכן אף על פי שכמות הזמן שמפלצות נמצאות תחת אש היא אולי מקסימלית, המגדלים עשויים להכשל בלגרום נזק משמעותי.
חסרון חדש: אם מכין כל השבילים יש שבילים שפיצ'ר השביל שלהם מביא ערך נמוך (למשל שבילים ישרים או שצמודים לגבולות המפה) אזי כנראה שלא ישתלם לשים שם מגדלים ואז יהיו שבילים לא מכוסים והמפלצות יעברו בלא מפריע בשבילים אלו.

2.3.2 פיצ'ר הרמות

מוטיבציה: המגדלים ברמה 0 מאוד חלשים, וקשה להם להרוג את המפלצות, כאשר ההמגדלים עולים רמות, הם נהיים משמעותית חזקים יותר -כמה עשרות אחוזים לנזק לכל עליית רמה. הגיון זה מעדיף שהמגדלים יהיו כמה שיותר חזקים ולא בהכרח שיהיה כמה שיותר יעילים במבחינת מיקום.
בפועל, פיצ'ר זה ממקסם את הרמות של המגדלים במשחק.
שוייה להווצר בעיה של צבירי מגדלים שכולם ברמה גבוהה אבל לא רואים שביל כלל (לדוגמא, ציור 2.1).
לכן הפיצ'ר סופר רק את הרמות של מגדלים שרואים שביל כלשהו.



ציור 2.1 - המגדלים מסודרים בצורה אופטימלית מבחינת עליית רמות, אבל לא תורמים לשחקן כלל

הגדרה פורמלית : יהי $tower\,(x,y)$ פרדיקט שמסתפק אם במשבצת יש מגדל.
נגדיר פונקציה $level\,(x,y)$ שמחזירה את רמת המגדל במיקום מסוים.

$$level\,(x,y) = \sum_{-1\le i,j\le 1} tower\,(x+i,y+j) - tower\,(x,y)$$

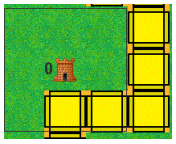
נשים לב כי אנו מתייחסים אל הפרדיקט כמחזיר 1 אם מסתפק ו 0 אחרת, וכי כמובן לא נחשיב את המגדל עצמו ברמות.
יהיה $seesRoad\,(x,y)$ פרדיקט שמסתפק אם המשבצת שכנה לשביל כלשהו.
בגלל התנאי הנוסף שהוספנו לפיצ'ר, נגדיר מושג "רמה אפקטיבית" שהיא רמת המגדל, אם ורק אם הוא רואה שביל $effectiveLevel\,(x,y)$:

$$effectiveLevel\,(x,y) = \begin{cases} level(x,y) & \text{if seesRoad}(x,y) \\ 0 & else \end{cases}$$

נגדיר את פיצ'ר הרמות f_{LEVEL}:

$$f_{LEVEL}(NODE) = \sum_{(x,y)\in TOWERS} effectiveLevel(x,y)$$

חסרונות : **אין התייחסות לטיב מיקום המגדל**, כלומר: אם היוריסטיקה יכולה לשפר מגדל בכמה משבצות שונות, היא לא תעדיף את המשבצת שרואה יותר שביל , לדוגמא, ציור 2.2
כמו בפיצ'ר הקודם, אין התייחסות לכיסוי המסלולים השונים במפה.



ציור 2.2 - הפיצ'ר יתן ערך זהה לכל מצב שבו נקנה מגדל ליד המגדל שבציור, אף על פי שבבירור יש מצבים עדיפים אסטרטגית (למשל מימין למגדל הקיים)

2.3.3 פיצ'ר השביל החדש

מוטיבציה: בפיצ'ר השביל הקודם (2.3.1) אנחנו לא מתחשבים בשאלה : האם השביל הוא חלק ממסלול אחד? חלק משלושה מסלולים? וכו'. כלומר מגדל שמכסה נקודה בשביל שהיא רק חלק ממסלול בודד, תשפיע כנראה פחות מנקודה שהיא חלק מכל המסלולים.
אנחנו מנסים למנוע מצב שבוא נבחר מיקום מגדלים שמכסה את מסלולים מסוימים בלי לכסות מסלולים אחרים כלל, ואנחנו עושים את זה ע"י נתינת "בונוס" (שהיא ספירה כפולה) של השבילים שהם חלק מכמה מסלולים.

הגדרה פורמלית : יהי $road(x,y,p)$ פרדיקט שמחזיר 1 אם הנקודה היא שביל וגם חלק ממסלול p ו־0 אם לא.
תהי $neighboringRoads(x,y,p)$ פונקציה שמחזירה לכל משבצת את כמות השכנים שמקיימים את $road$

$$neighboringRoads(x,y,p) = \sum_{-1\le i,j\le 1} road(x+i, y+j, p)$$

אנו מתעלמים מ $road(x,y,p)$ כיוון שאנו תמיד נקנה מגדל במקום חוקי, ולכן בהכרח $road(x,y,p) = 0$.
לבסוף תהי $TOWERS$ קבוצת מיקומי המגדלים במצב נתון, נגדיר את פיצ'ר השביל f_{ROAD}:

$$f_{ROAD}(NODE) = \sum_{p} \sum_{(x,y)\in TOWERS} neighboringRoads(x,y,p)$$

חסרונות : זהו פיצ'ר בעייתי כיוון שהמגדלים ברמה 0 חלשים, וקשה להם לפגוע בצורה משמעותית במפלצות, ולכן אף על פי שכמות הזמן שמפלצות נמצאות תחת אש היא אולי מקסימלית, המגדלים עשויים להכשל בלגרום נזק משמעותי.

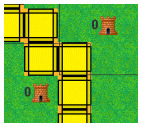
2.3.4 יוריסטיקה משוקללת

מוטיבציה: כל אחת מהיוריסטיקות הקודמות מתמקדת באלמנט אחר של בחירת מיקום למגדלים, ומתעלמת מהאלמנטים אחרים , המטרה היא למזער את החסרונות שקיימים בשתי היוריסטיקות, במחיר חישוב יוריסטיקה כבד יותר (כעת צריך לחשב בכל פעם את שתי היוריסטיקות)

הגדרה פורמלית: עבור פרמטר $0 \le \alpha \le 1$ היוריסטיקה היא:

$$h_{WEIGHTED}(NODE) = \alpha \times f_{ROAD}(NODE) + (1-\alpha) \times f_{LEVEL}(NODE)$$

חסרונות : היוריסטיקה כבדה יותר, היוריסטיקה לא מתייחסת ל"איכות" המגדלים אותה היא משפרת, כלומר היא לא תעדיף לשפר מגדל שרואה הרבה שביל, על פני לשפר מגדל שרואה מעט שביל (בהנחה וזה הדבר היחיד ששונה), לדוגמא, ציור 2.3.



ציור 2.3 - היוריסטיקה לא תעדיף לבנות מגדל מתחת למגדל השמאלי לעומת לבנות משמאל למגדל הימני (בשניהם המגדל החדש יראה 3 שבילים) אף על פי ששיפור המגדל השמאלי עדיף (כי הוא רואה יותר שביל ולכן אפקטיבי יותר)

2.3.5 יוריסטיקת המלבן המוכללת

מוטיבציה: המטרה של יוריסטיקה זו היא ליצור איזון בין מיקום מגדלים במקומות טובים,כיסוי כל השבילים ובין החזקה במגדלים ברמה גבוהה.

ביוריסטיקה זו, אנו מכפילים את: ערך פיצ'ר הרמות, ואת כמות השבילים שמכוסים ע"י המגדלים **לכל מסלול**.

אפשר לחשוב על ערך היוריסטיקה כמלבן, שאורכו הוא יוריסטיקה אחת, וגובהו הוא היוריסטיקה השניה. אם במצב משחק מסיום שטח המלבן הוא $a \times b$ ו $a > b$, בחיפוש הבא תהיה "שאיפה לריבוע", כלומר אם יהיו שני מצבי משחק, אחד שמאריך את a והשני שמאריך את b, תהיה העדפה להאריך את b (כיוון ש a גדול יותר) . כלומר בכל שלב החיפוש "ירבע" את המלבן.

בפועל, בשלבים במשחק שבהם יש יוריסטיקה שחוזקה בצורה משמעותית (למשל הרבה מגדלים בנקודות מצוינות, או מגדלים ברמה גבוהה) האלגוריתם ישאף "לרבע" את המלבן, ולכן יחזק את היוריסטיקה שתורמת פחות (את הצלע הקצרה), נשים לב כי העובדה שבשלב מסוים יוריסטיקה אחת חוזקה פחות, לא הופך אותה "בעיני האלגוריתם" לחשובה פחות, להפך החשיבות שלה עולה ככל שהפער גדל.

היתרון המשמעותי של היוריסטיקה הזו, זה שהיא מתחשבת בבחירות הקודמות.

כדי שהמספרים לא התפוצצו השתמשנו ב $\log$ והפכנו את המכפלה לסכום

הגדרה פורמלית: ערך היוריסטיקה הוא:

$$h_{PRODUCT}(NODE) = \log(h_{LEVEL}(NODE)) + \sum_{p} \log\left[\sum_{(x,y)\in TOWERS} neighboringRoads(x,y,p)\right]$$

חסרונות: לא מתחשבת במפלצות הנמצאות במקום מתקדם במפה ויכולים להרג בקלות, רק שאין מי שיהרוג אותן.

3 המערכת ששימשה למימוש הפתרון

המערכת נשארה זהה למערכת המקורית (חוץ מהפיצ'ר שהוסף בה) והיא מתוארת בקובץ סיכום הפרוייקט המקורי.

4 מתודולוגיה ניסויית

בכל ניסוי הורץ משחק. במשחק יש 20 שלבים, כאשר בשלב יש מפה. בכל ניסוי הרצתי את המשחק 3 פעמים ולקחתי את הממוצע של 3 ההרצות לכל שלב.

4.1 מפות

כדי שהמשחק יהיה מגוון, בחרתי שלושה מאפיינים למפה, ודאגתי לשונות במפות במאפיינים אלו.

- אורך השביל - כמה משבצות מסוג שביל יש במפה
- שטח אפקטיבי - מחושב ע"י שטח המלבן הקטן ביותר שאיתו ניתן "ללכוד" את השביל
  - פרמטר זה מודד בכמה מהמפה "משתמש" השביל, כאשר ההגיון במדידת פרמטר זה הוא שככל שהשטח האפקטיבי קטן יותר, יש פחות נקודות טובות לשים מגדלים.
- מספר זוויות התקפה - מחושב כסכום כל ה $neighboringRoads\,(x,y)$ (שהוגדר ב 3.2.1) על כל המשבצות שאינן שביל.
  - פרמטר זה מודד כמה "חשוף" השביל. כאשר שביל איננו חשוף (למשל כשהוא קרוב לקיר או באלכסון עם שביל אחר) יש פחות נקודות תקיפה עליו, וכנראה שהמפה תהיה קשה יותר.
- "חשיפות ממוצעת" של שביל - מספר זוויות התקפה \ אורך השביל.
  - פרמטר זה מודד כמה חשופה כל יחידת שביל במפה באופן ממוצע.

מצורפת טבלת הפרמטרים בציור 4.2

level	path length	effective area	attack angles	average angles
[levels/level1.txt]	32	108	156	4.88
[levels/level2.txt]	40	120	185	4.62
[levels/level3.txt]	25	60	123	4.92
[levels/level4.txt]	35	96	163	4.66
[levels/level5.txt]	54	120	240	4.44
[levels/level6.txt]	66	120	288	4.36
[levels/level7.txt]	37	120	143	3.86
[levels/level8.txt]	52	120	184	3.54
[levels/level9.txt]	36	100	195	5.42
[levels/level10.txt]	13	32	68	5.23
[levels/level11.txt]	44	120	158	3.59
[levels/level12.txt]	52	120	188	3.62
[levels/level13.txt]	27	88	135	5
[levels/level14.txt]	43	110	187	4.35
[levels/level15.txt]	35	96	155	4.43
[levels/level16.txt]	53	120	236	4.45
[levels/level17.txt]	51	120	214	4.2
[levels/level18.txt]	45	120	190	4.22
[levels/level19.txt]	48	108	217	4.52
[levels/level20.txt]	25	54	137	5.48

ציור 4.2 - טבלת פרמטרי המפות

4.2 הפרמטרים שנבדקו

בכל הרצה של השחקן החכם ניתנו כמה פרמטרים לתוכנית:

1. t - זמן ריצת אלגוריתם האלומה (milliseconds)
2. w - גודל האלומה של אלגוריתם האלומה
3. $heuristic$ - ההיוריסטיקה בה משתמשים

4.3 מדדים להצלחה

בכל שלב יש 40 מפלצות. ההצלחה של שחקן באחוזים בכל שלב היא $k/40$ כאשר k הוא מספר המפלצות שהשחקן הצליח להרוג בשלב (נשים לב כי כמובן, אם מפלצת עוברת ולא נהרגת היא לא נספרת)
בסוף לוקחים את הממוצע של 20 השלבים כציונו של השחקן בכל ניסוי.

5 הניסויים

5.1 ניסוי 1 - השוואת הפיצ'ר של הספירה לפי מסלולים לפיצ'ר המסלול הישן

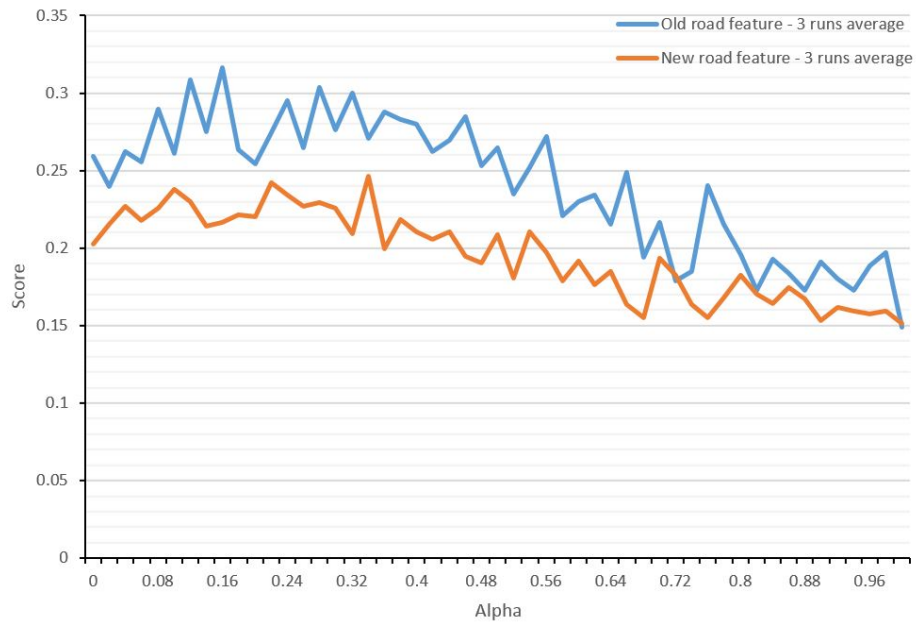
מוטיבציה לניסוי: בניסוי זה אנסה לענות על השאלות:
האם עדיף לספור כמה נקודות **במסלול כלשהו** כל מגדל רואה או עדיף לסכום את המשבצות שהן חלק מכמה מסלול יותר פעמים?

תיאור הניסוי : הרצתי את המשחק עם הפרמטרים:

- t - 3.5 שניות לכל חיפוש.
- w - גודל אלומה מקסימלי הוא 200
- היוריסטיקות - $\alpha \in [0, 0.02, 0.04, \ldots, 1]$
  - היוריסטיקה משוקללת של פיצ'ר השביל הישן ופיצ'ר הרמות
  - היוריסטיקה משוקללת של פיצ'ר השביל החדש ופיצ'ר הרמות

דוגמא : עבור $\alpha = 0.2$ ההיוריסטיקה היא $0.2 \times f_{ROAD} + 0.8 \times f_{LEVEL}$

תוצאות הניסוי: סה"כ יש 51 נקודות, להלן גרף התוצאות:



הבחנות ואבחנות :

1. כמצופה, כאשר מגיעים ל $\alpha \approx 1$ הגרפים מתקרבים, כיוון שההשפעה של פיצ'ר השביל הופכת לזניחה
2. כצפוי, התוצאות משמעותית נמוכות יותר מהתוצאות של ניסוי דומה לפני הרחבת הפרוייקט - כלומר הפיצ'רים לא מספיק "חכמים" (בניסוי המקורי הגענו לכ 65% הצלחה)
3. באופן מפתיע, דווקא ההיוריסטיקה שמשתמשת בפיצ'ר הישן, טובה יותר באופן מוחלט.

סיכום ניסוי: גילינו כצפוי, שהמשחק קשה יותר מהמשחק המקורי, אך לא הצלחנו להראות שפיצ'ר השביל החדש טוב יותר מהישן, ניתן להסביר תופעה זו בכך שחישוב הפיצ'ר החדש הוא כבד יותר. בניסוי הבא ננסה לבדוק טענה זו. בנוסף ניתן לראות שפיצ'ר השביל הוא הפיצ'ר המשמעותי יותר במקרה זה והערך המקסימלי מתקבל ב $\alpha \approx 0.16$ עם תוצאה 0.32 עבור הפיצ'ר הישן

5.2 ניסוי 2 - השוואה בין פיצ'ר השביל החדש והישן, לאורך זמנים שונים

מוטיבציה לניסוי: בניסוי זה אני אנסה לענות על השאלות:
האם באמת הפיצ'ר השני כבד יותר חישובים (נראה את זה כאשר $t \to \infty$)?
האם תוצאותיו באמת מדיוקות יותר?
האם היוריסטיקת המלבן המוכללת תתן תוצאות טובות יותר מההיוריסטיקות הפשוטות יותר?

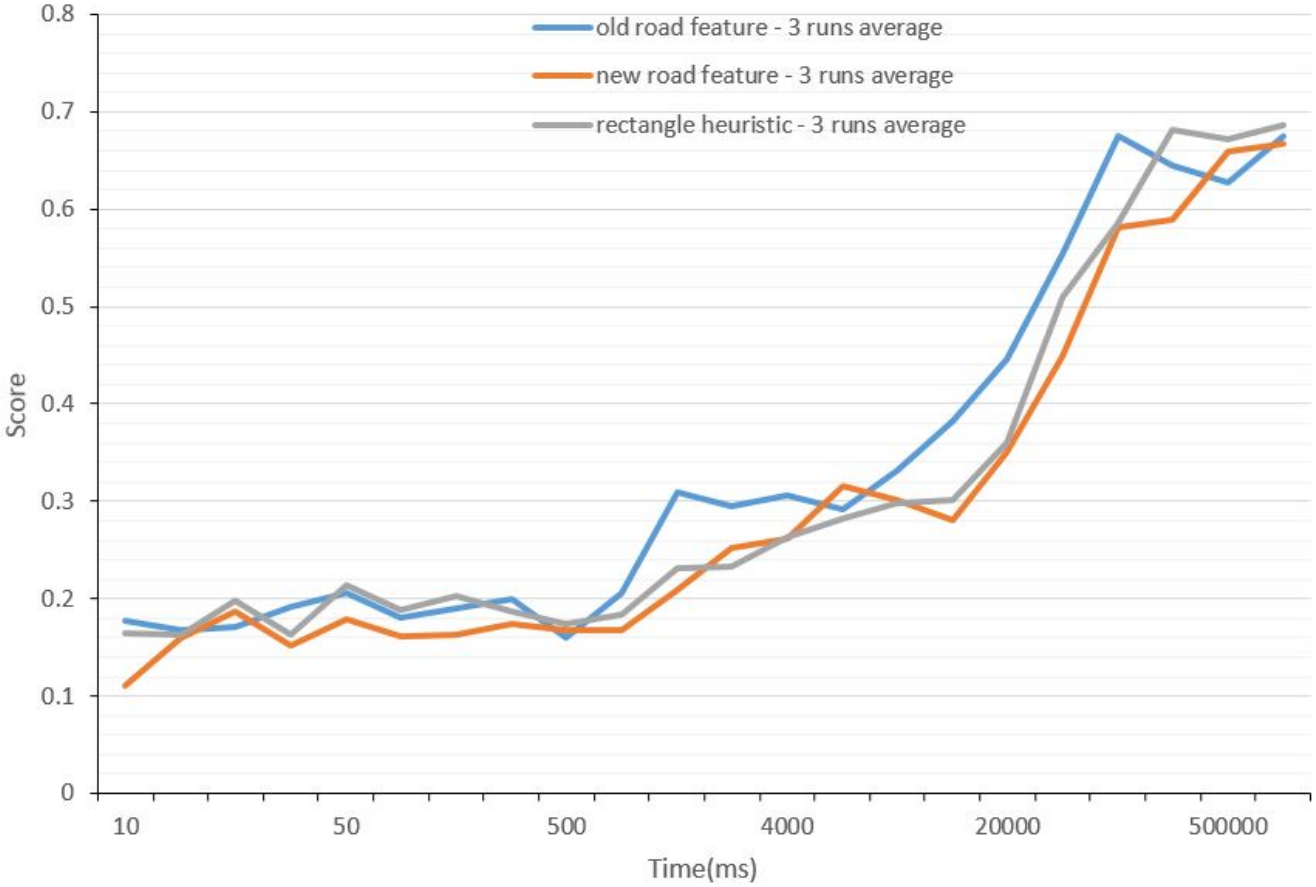
תיאור הניסוי : הרצתי עבור
הרצתי את המשחק עם הפרמטרים:

- t - הפרמטר עליו נעשה הניסוי

$$t \in \{10, 20, 30, 40, 50, 100, 200, 300, 500, 1000, 2000, 3000, 4000, 5000, 7500, 10000, 20000, 50000, 100000, 200000, 500000, 1000000\}$$

- w - 100 , גודל אלומה מקסימלי הוא 100
- $heuristic$
  - שתי היוריסטיקות המשוקללותת כמו מקודם)עם שלפא קבוע הפעם(
  - היוריסטיקת המלבן המוכללת

תוצאות הניסוי: סה"כ יש 22 דגימות, להלן גרף התוצאות.



הבחנות ואבחנות :

- ניתן לראות שהגדלת זמן החיפוש, משפרת את הביצועים של השחקנים.
- ניתן לראות רוויה בזמנים המאוד גדולים (כמה עשרות שניות לחיפוש).

- ניתן לראות כי יוריסטיקת המלבן, מצליחה להגיע לתוצאות הכי טובות בזמנים הגדולים, אך לא בפער גדול.
- פיצ'ר השביל החדש לא הצליח גם הפעם להוכיח את עצמו כטוב יותר מהפיצ'ר הישן

סיכום ניסוי: תוצאות הניסוי הפתיעו אותי, כיוון שציפיתי לשיפור גדול יותר של ההיוריסטיקה שמתחשבת במסלולים השונים, אך התוצאות מראות שלאורך כל הדרך הם כמעט זהים - עם ייתרון לפונק' הישנה(!)
תוצאה נוספת מפתיעה שניתן לראות היא שגם היוריסטיקת המלבן לא נתנה לנו שיפור בביצועים, שגם זה לא היה צפוי, כיוון שציפיתי שהיוריסטיקה זו, שהיא הכללה של ההיוריסטיקה מהדו"ח הקודם, תראה שוב עליונות על ההיוריסטיקות האחרות.

5.3 ניסוי 3 - ניסוי רוחב אלומה

מוטיבציה לניסוי: בניסוי זה אני אנסה לענות על השאלות:
איך ישפיע רוחב האלומה על היוריסטיקות השונות?
מה יקרה כאשר רוחב האלומה הוא 1 - כלומר החיפוש הופך ל SAHC קלאסי?
האם ככל שנגדיל את רוחב האלומה התוצאות יפגעו, כפי שראינו בניסוים שערכנו בחלקו הראשון של הפרוייקט?

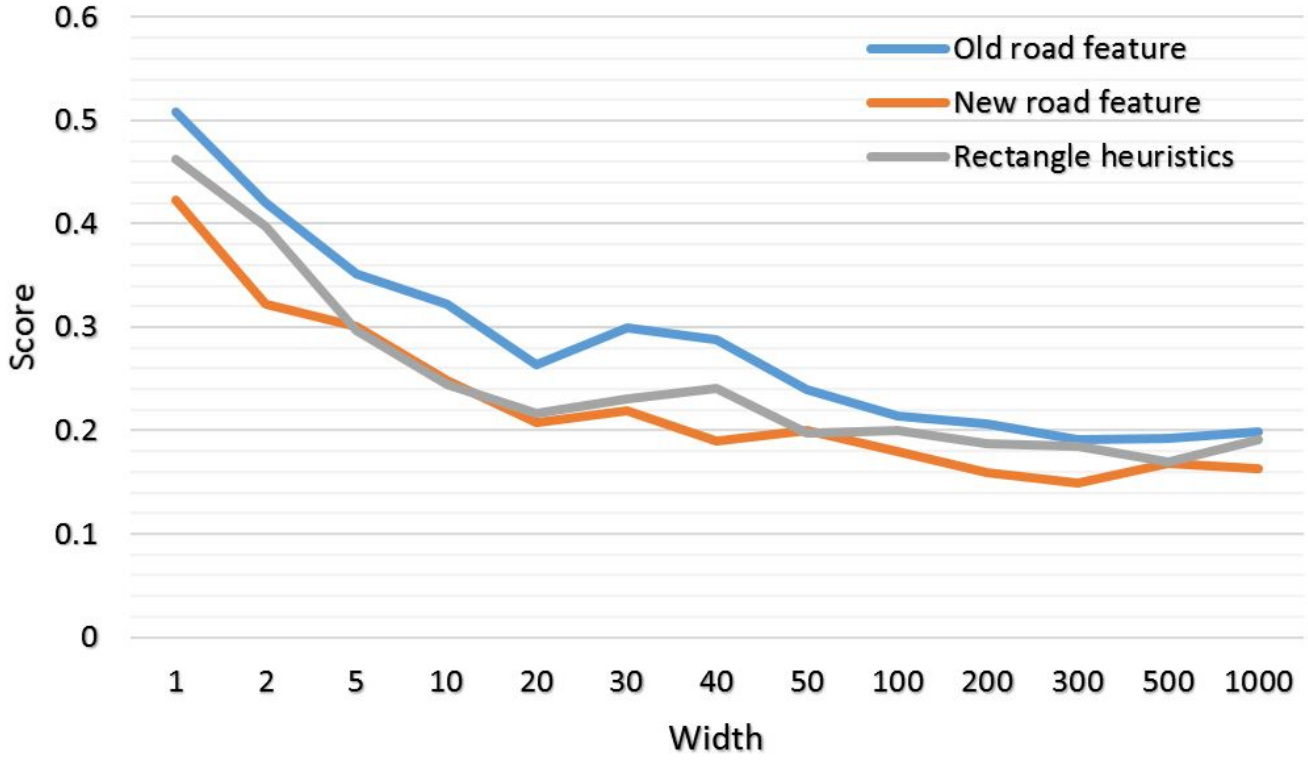
תיאור הניסוי : הרצתי את המשחק עם הפרמטרים:

- t - 1000, שניה אחת לכל חיפוש
- w - הפרמטר עליו נעשה הניסוי

$$w \in \{1, 2, 5, 10, 20, 30, 40, 50, 100, 200, 300, 500, 1000\}$$

- $heuristic$
  - כמו בניסוי הקודם

תוצאות הניסוי: סה"כ יש 13 דגימות, להלן גרף התוצאות:



הבחנות ואבחנות :

- היוריסטיקות מוצו מאוד מהר, ועושה רושם שהגדלת רוחב האלומה לא עזר כלל.
- היוריסטיקות מתנהגות מאוד דומה ביחס לשינוי גודל האלומה
- התוצאות הטובות ביותר התקבלו כאשר $w = 1$ כלומר, כשהחיפוש הוא חיפוש $SAHC$ (שלוש היוריסטיקות החזקות הגיעו לציון של כ־0.9)

סיכום הניסוי: תוצאות הניסוי תאמו את הציפיות שהיו לי מהניסוי המקביל בחלק הראשון של הפרוייקט: ניתן להסביר את התוצאות ע"י כך שככל הנראה, זמן החיפוש היה קצר מדי. רק בערכים הנמוכים של w הצליח האלגוריתם להגיע "לעומק" שם נמצאים הצמתים הטובים. כנראה שכשהגדלנו את גודל האלומה, האלגוריתם לא הספיק להגיע לעומק גרף המצבים, אלא רק לרוחבו.

6 סיכום

6.1 דיון בתוצאות

6.1.1 פי'צרים והיוריסטיקות

- כפי שנראה בבירור הניסויים, פיצ'ר השביל החדש לא הצליח להתעלות על הפיצ'ר הישן - תוצאה מפתיעה. יכול להיות שהסיבה לכך היא שהפיצ'ר החדש העדיף למקם מגדלים ליד שביל שנמצא בכמה מסלולים - אבל אולי המיקום שלו לא טוב, למשל הוא רואה רק שני שבילים לידו (אבל הם נכללים ב3 מסלולים שונים ולכן הערך היוריסטי שלהם הוא) ומעדיף את זה על נקודה אחרי שרואה 5 שבילים שונים (אבל הם חלק רק ממסלול אחד- ולכן ערכו היוריסטי הוא 5)
- הרחבת המשחק הפכה את המשחק להרבה יותר קשה
  - כל היוריסטיקות החדשות שלנו נכשלו בהשגת תוצאות טובות
  - לא הצלחנו להתקרב כלל לאחוזי הצלחה טובים כמו בפרוייקט הקודם - אז כמעט הגענו ל100% במקרה הטוב ביותר(!!) - פה אנחנו מגרדים את ה 70%.
- היוריסטיקת המלבן כבד, ובקושי הצליח להביא תוצאות טובות יותר מאשר ההיוריסטיקות השונות.

6.1.2 תלות בין פרמטרי בדיקה

בניסויים השונים גילינו תלות בין הפרמטרים של האלגוריתם:
ראינו שהגדלת רוחב האלומה יכול לעזור או לפגוע בציון השחקן, תלוי בזמן החיפוש. כלומר לכל בחירת לכל זמן ריצה של האלגוריתם, קיים רוחב האלומה שעבורו מקבלים ציון אופטימלי.
ראינו שהגדלת זמן הריצה עוזר, עד הגעה לרוויה. כלומר בחירת זמן הריצה איננה תלויה בפרמטרים האחרים, ככל שנקח זמן ריצה גדול יותר נקבל בממוצע ציון טוב יותר.

6.2 סיכום

במסגרת פרוייקט זה הצענו אלגוריתם anytime לפתרון בעיית ה - subset selection : האלגוריתם היה חיפוש אלומה לוקאלי כאשר הוא מתקדם כמו $SAHC$ האלגוריתם קיבל כמה פרמטרים: רוחב אלומה, זמן ריצה, וההירוסטיקה איתה הוא עושה חישובים.
בחרתי 3 פיצ'רים שמייצגים את השיקולים המרכזיים שעומדים בפני שחקן אנושי שמשחק במשחק :

1. מיקום המגדלים במקום אסטרטגי
2. כיסוי של המסלולים השונים במפה
3. חיזוק מגדלים קיים ע"י הצבת מגדלים שכנים.

בחרתי שלוש היוריסטיקות : שילוב של (1) ו (2), שילוב של (1),(3) והיוריסטיקת המלבן שהוא שילוב האלמנטים בצורה מתמטית שונה (כדי להגיע לאיזון)
מהניסויים ניתן לקראות בבירור את הדומיננטיות של פיצ'ר השביל הישן - אותו לא שינינו מהפרוייקט הקודם, וכנראה שיתרון זה נובע כיוון שהפיצ'ר החדש פחות בוחר את המיקומים שרואים הרבה שביל ויותר בוחר שבילים שמוכלים בהרבה שבילים.
בנוסף, ניתן לראות שיוריסטיקת המלבן לא מצליחה ממש להתעלות על ההיוריסטיקות האחרות.
בניסויים הראשונים על רוחב אלומה גילינו שהגדלת רוחב האלומה פגע בביצועים, ושהביצועים הטובים ביותר היו של אלומה ברוחב 1 (כלומר חיפוש SAHC) , תוצאה דומה נצפתה גם בניסויים בפרוייקט הקודם וההסבר להם הוא: כנראה שהזמן שאיתו הורץ האלגוריתם היה קטן מדי, וכשהגדלנו את האלומה, האלגוריתם לא הצליח להגיע לעומק גרף המצבים - כי רוב זמנו הושקע ברוחב.
מסקנה נוספת מהניסוי היא שכדי לנצל את רוחב האלומה לביצועים, צריך להגדיל את זמן הריצה בסדרי גודל , ניתן לראות את זה בניסוי (2).

לסיכום הגענו למסקנה שהמשחק המורחב הרבה יותר קשה מהמחשק המקורי, והשינוי קטן בהיוריסטיקות לא מספיק על מנת להגיע לתוצאות ממש טובות. ישנה אפשרות גם, שבלתי אפשרי להגיע לתוצאות טובות יותר, ותוצאות המשחק האופטימליות חסומות לפני ציון של 100%, אך את היפוטזה זו מאוד קשה לבדוק.