236502 - פרוייקט בבינה מלאכותית

תומר לוי stlevy@t2.technion.ac.il

19 באוקטובר 2014

נושא: **שימוש בחיפוש לוקלי למשחקי** Tower Defense

מנחה: פרופ' שאול מרקוביץ

מתרגל : מר עומר גייגר

תוכן עניינים

1 מבוא

1.1 ז'אנר Tower Defense

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.

- 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:
  - מורכבת מ־2 סוגי משבצות : שביל, ואדמה.
  - בגודל 12×10.
  - בסוף השביל יש את המטרה אליה צועדות המפלצות.
- המפלצות:
  - המפלצות מתחילות משביל על הפאה השמאלית ויכולות ללכת רק על השביל, ולהגיע למטרה. (בציור: המפלצות כחולות והשביל צהוב)
  - כל מפלצת שמצליחה להגיע למטרה מורידה חיים אחד לשחקן, ונעלמת מהמשחק.
- המגדלים:
  - המגדלים יורים במפלצות, ומורידים להם חיים.
  - כל מגדל עולה רמה על כל מגדל אחר שנמצא בשכנות אליו , כלומר רמת מגדל היא מספר המגדלים השכנים לו. (מדובר על שכנות־8 , כלומר גם אלכסונים)
  - בכל רגע, 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.

לדוגמא בציור 1.1 אפשר לראות את משחק ה־TD.

ציור 1.1 - דוגמא למשחק TD בזמן ריצה

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

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

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

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

  - **צומת התחלתי:** נבחר ע"י אלגוריתם חמדן שעובד מאוד מהר, ומתעלם מרמות המגדלים.
  - **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 קבוצות האופרטורים החוקיים מצומת לפי קבוצת ה־successors שלו, פורמלית :

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

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

2.2 אלגוריתם החיפוש

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 יוריסטיקות

2.3.1 יוריסטיקת השביל

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, יוריסטיקה זו ממקסמת 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 $road$

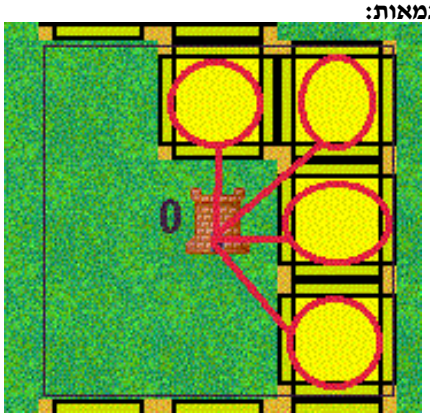
$$neighboringRoads(x,y) = \sum_{-1 \le i,j \le 1} road(x+i, y+j)$$

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 יוריסטיקת השביל h_{ROAD}:

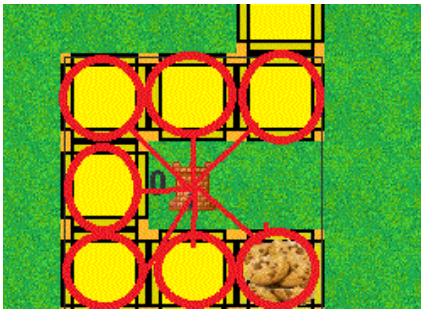
$$h_{ROAD}(NODE) = \sum_{(x,y) \in TOWERS} neighboringRoads(x,y)$$

חסרונות : זוהי יוריסטיקה בעייתית 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.

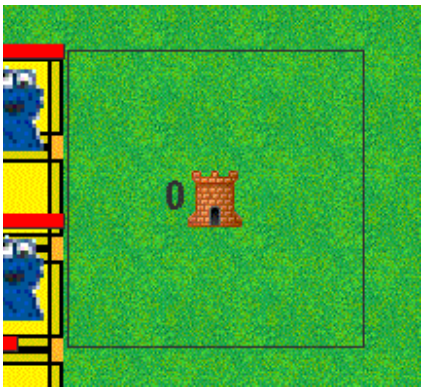
דוגמאות:



ציור 2.1 - המגדל מוסיף ערך יוריסטי 4



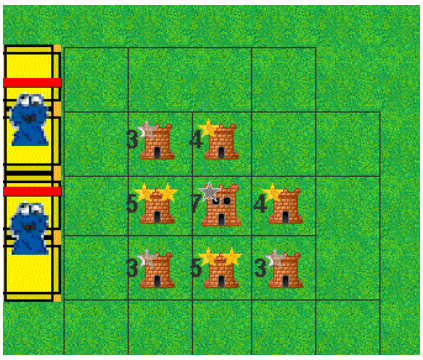
ציור 2.2 - המגדל מוסיף ערך יוריסטי 7



ציור 2.3 - המגדל לא מוסיף ערך יוריסטי

2.3.2 יוריסטיקת 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, יוריסטיקה זו ממקסמת את הרמות של המגדלים במשחק.
עשוייה להווצר בעיה של צבירי מגדלים שכולם ברמה גבוהה אבל לא רואים שביל כלל (לדוגמא, ציור 2.4).
לכן היוריסטיקה סופרת רק את הרמות של מגדלים שרואים שביל כלשהו.



ציור 2.4 - 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.

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

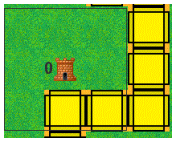
נשים לב כי אנו מתייחסים אל הפרדיקט כמחזיר 1 אם מסתפק ו 0 אחרת, וכי כמובן לא נחשיב את המגדל עצמו ברמות.
יהיה $seesRoad\,(x,y)$ פרדיקט שמסתפק אם המשבצת שכנה לשביל כלשהו.
בגלל התנאי הנוסף שהוספנו ליוריסטיקה, נגדיר מושג "רמה אפקטיבית" שהיא רמת המגדל, אם ורק אם הוא רואה שביל $effectiveLevel\,(x,y)$:

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

נגדיר את יוריסטיקת הרמות h_{LEVEL}:

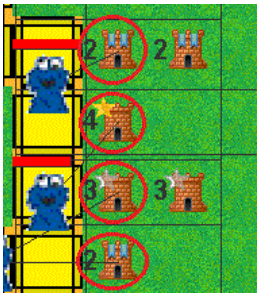
$$h_{LEVEL}\,(NODE) = \sum_{(x,y)\in TOWERS} effectiveLevel\,(x,y)$$

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 2.4

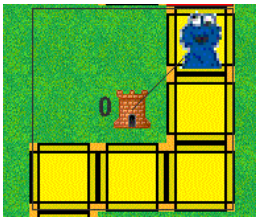


ציור 2.4 - היוריסטיקה תתן ערך זהה לכל מצב שבו נקנה מגדל ליד המגדל שבציור, אף על פי שבבירור יש מצבים עדיפים אסטרטגית (למשל מימין למגדל הקיים)

דוגמאות:



ציור 2.5 - הערך היוריסטי המוסף ע"י המגדלים הוא 11, כיוון שהמגדלים מימין לא אפקטיביים, ולכן רמתם לא נספרת.



ציור 2.6 - המגדל לא מוסיף ערך יוריסטי.

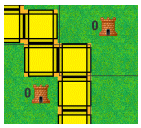
2.3.3 יוריסטיקה משוקללת

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 $0 \leq \alpha \leq 1$ היוריסטיקה היא:

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

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 2.5.



ציור 2.5 - 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)

2.3.4 יוריסטיקת המלבן

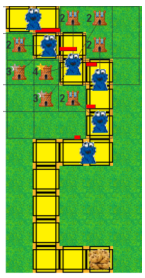
מוטיבציה: המטרה של יוריסטיקה זו היא ליצור איזון בין מיקום מגדלים במקומות טובים, לבין החזקה במגדלים ברמה גבוהה.
ראשית נסביר רעיונית במה היא שונה מהיוריסטיקה המשוקללת: היוריסטיקה המשוקללת עשויה "להרעיב" את אחת מהיוריסטיקות, למשל אם יש מפה שיש בה הרבה פינות שרואות הרבה שביל, היא עשויה לתפוס את כל הנקודות האלה, בלי להתחשב בעובדה שכבר נבחרו כמה נקודות עם נקודת תצפית ושכל המגדלים ברמה נמוכה.
ביוריסטיקה זו, אנו מכפילים את ערך שתי היוריסטיקות הטהורות שלנו.
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.

הגדרה פורמלית: ערך היוריסטיקה הוא:

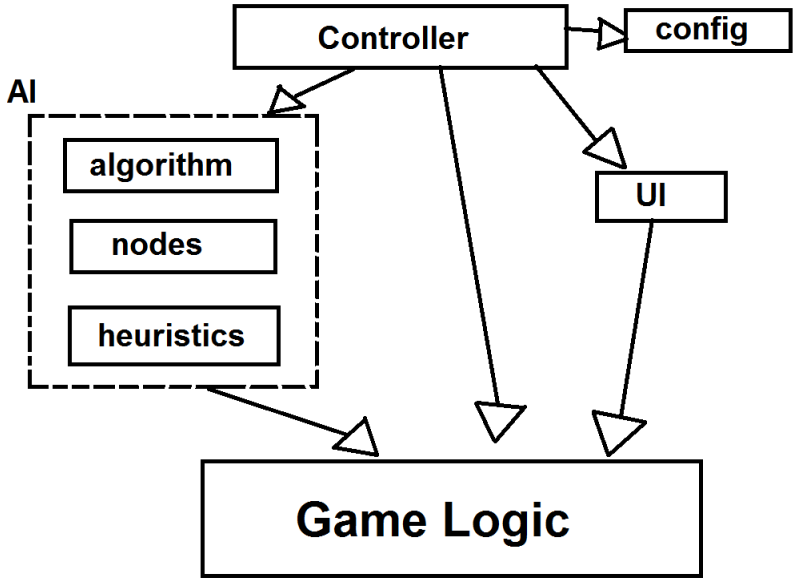
$$h_{PRODUCT}(NODE) = h_{ROAD}(NODE) \times h_{LEVEL}(NODE)$$

חסרונות: לא מתחשבת במפלצות הנמצאות במקום מתקדם במפה ויכולים להרג בקלות, רק שאין מי שיהרוג אותן, לדוגמא, ציור 2.6



ציור 2.5 - היוריסטיקה לא תתחשב בכך שהקמת מגדלים למטה יכולה להרוג מגדלים שבוודאות לא יהרגו אחרת (כי אין מי שיהרוג אותן)

3 המערכת ששימשה למימוש הפתרון



3.1 לוגיקת המשחק

למשחק מספר חלקים לוגיים שמתחברים ע"י ה"שרת" למשחק.

3.1.1 מפת המשחק

- שליטה על משבצות המפה - הוספת מגדלים
- שאילתות לגבי המשבצות במפה - מה סוג המשבצת, האם יש בה מגדל וכו'

3.1.2 מפלצות

- ניווט המפלצות במפה
- ניהול החיים של המפלצת
- שאילתות לגבי המפלצות : האם המפלצת חיה או מתה, מיקום וכ'

3.1.3 מגדלים

- חיפוש מפלצות בטווח המגדל
- ירי על המפלצות שבטווח
- חישוב הנזק של המגדל לפי הרמה שלו

3.1.4 שרת מרכזי

אחראי על האינטגרציה בין כל חלקי המערכת :

- מנהל את החיים והכסף במשחק
- מספק למפלצות נתוני מפה על מנת שיוכלו לנווט
- מספק למגדלים נתונים על המפלצות כדי שיוכלו לחפש אותם
- מחבר בין המפלצות למגדלים בכל הנוגע להורדת חיים והריגה
- אחראי על זימון מפלצות חדשות

- העלאת מגדלים רמות בכל קניית מגדלים חדשה

אחראי על דיסקרטיזציה של הזמן במשחק, ברכיב לוגי זה בעצם מתבצעת יחידת הזמן הבסיסית במשחק שממנה נגזרות כל המהירויות במשחק

- מהירות הליכה של מפלצות
- מהירות יריה של מגדלים
- מהירות זימון מפלצות חדשות

3.2 אלגוריתמים , צמתים ויוריסטיקות

אחראים על כל החלק של הבינה המלאכותית בפרוייקט.
האלגוריתמים רצים עם הפרמטרים שניתנים להם של זמן וגודל אלומה.
הצמתים עובדים עם לוגיקת משחק שניתנת להם, מעלים לה שאילתות, כדי לבצע הרחבה של צמתים.
היוריסטיקות עובדות עם הצמתים ומחשבות את הערך היוריסטי של כל צומת, לפיהן האלגוריתמים ממינים את הצמתים.
החיפוש רץ בכל פעם שהשחקן הרוויח מספיק כסף לקנות מגדל חדש, הוא רץ עם אותם פרמטרים שקיבל המשחק, בזמן הרצת האלגוריתם המשחק עוצר.

3.3 קונפיגורציות

הקונפיגורציות איתן המשחק רץ, הן קבועות לכל המשחק ומגיעות בקובץ קונפיגורציות. הקונפיגוריציות מכילות:

מגדלים: מהירות ירי, טווח תקיפה, נזק התחלתי, בונוס התקפה באחוזים לכל עליית רמה

מפלצות: חיים התחלתיים, מהירות הליכה, תגמול כספי על הריגה, מהירות זימון

חדר: מימדי החדר, גודל כל בלוק

חנות: מחיר המגדל

אחר: חיים התחלתיים, כסף התחלתי, כמות המפלצות בשלב, כמות השלבים במשחק

קובץ הקונפיגורציות הוא /configurations/conf.ini
גם השלבים עצמם ניתנים לקינפוג ונמצאים בקבצים בשם /levels/level$\langle i \rangle$.txt

בנוסף, קיים קובץ קונפיגורציות ל GUI : /configurations/gui_conf.ini

3.4 ממשק משתמש

בחלק זה יש את כל ממשקי המשתמש:

- GUI- הממשק הגרפי
  - ניתן להריץ את המשחק ללא שחקן AI כמתואר בקובץ $readme.txt$, ואז צריך לבחור את המגדלים מהתפריט ולהתחיל את המשחק ע"י לחיצה על הטקסט שמעל המפה.
- ממשק טקסטואלי שמדפיס הודעות על ערכי יוריסטיקה

3.5 קוד חיצוני

בפרוייקט השתמשתי בכמה קודים חיצוניים, כאשר כולם שונו מצורתם המקורית:

- משחק TD בסיסי עם GUI נלקח ממדריך מאתר יוטיוב
  - אין בקישור את הקוד, אלא את המדריך, לא מצאתי את הקוד המקורי מוקלד, הקלדתי תוך כדי צפייה בסרטון https://www.youtube.com/playlist?list=PLB85864F4E46196A9
  - השימוש בקוד זה מפוזר בכל ה $package$ שנקרא $logic$ וקצת ב gui
- מחלקות אבסטרקטיות של אלגוריתם וצומת
  - http://cs.gettysburg.edu/ tneller/resources/ai-search/uninformed-java/index.html
  - מהקבצים SearchNode,Searcher (יש קישורים באתר)
  - השימוש שלהם נמצא במחלקות AbstractNode,AnytimeAlgorithm בהתאמה.

4 מתודולוגיה ניסויית

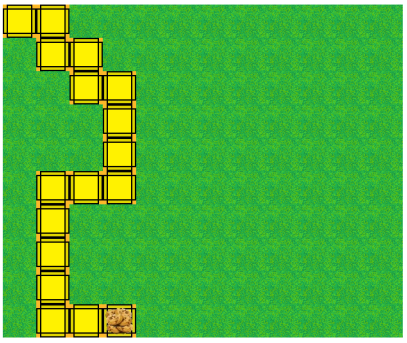
בכל ניסוי הורץ משחק. במשחק יש 20 שלבים, כאשר בשלב יש מפה. בכל ניסוי הרצתי את המשחק 4 פעמים ולקחתי את הממוצע של 4 ההרצות לכל שלב.

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. לדוגמא, ציור 4.1
- 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



ציור 4.1 - השטח האפקטיבי של מפה זו הוא 40, כי רוחב הנקודה השמאלית עד הימנית ביותר בשביל הוא 4, והגובה הוא 10.

	A	B	C	D	E
1	level	path length	effective area	attack angles	average angles
2	[level1.txt]	17	40	71	4.18
3	[level2.txt]	33	90	152	4.61
4	[level3.txt]	16	24	75	4.69
5	[level4.txt]	14	30	62	4.43
6	[level5.txt]	54	110	240	4.44
7	[level6.txt]	66	120	292	4.42
8	[level7.txt]	21	120	83	3.95
9	[level8.txt]	43	120	130	3.02
10	[level9.txt]	31	70	167	5.39
11	[level10.txt]	11	24	55	5
12	[level11.txt]	43	120	148	3.44
13	[level12.txt]	41	108	145	3.54
14	[level13.txt]	21	55	100	4.76
15	[level14.txt]	40	90	167	4.17
16	[level15.txt]	23	45	110	4.78
17	[level16.txt]	45	120	205	4.56
18	[level17.txt]	28	120	118	4.21
19	[level18.txt]	38	96	157	4.13
20	[level19.txt]	36	110	168	4.67
21	[level20.txt]	13	40	67	5.15

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. $alpha$ - האלפא איתה משקללים את היוריסטיקות כמתואר ב-2.3.3

4.3 מדדים להצלחה

במשחק זה השתמשתי בשני מדדים דיי קורולטביים (אך לא לחלוטין), והם אחוז ההצלחה בשלב, ומספר השלבים בהם השחקן הפסיד (תזכורת: שחקן מפסיד אם הוא איבד את כל החיים)

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.

בסוף המשחק לוקחים את כמות השלבים שבהם השחקן הפסיד למדד השני.

הסיבה שבחרתי את המדד השני היא שבין שני שחקנים עם ציון זהה, אני אעדיף שחקן שבוא המדד השני נמוך יותר, כלומר: אעדיף שחקן שבכמה שלבים "טיפטפו" מפלצות שהצליחו לחדור מאשר שלבים שלמים שהשחקן כשל בהם.

5 הניסויים

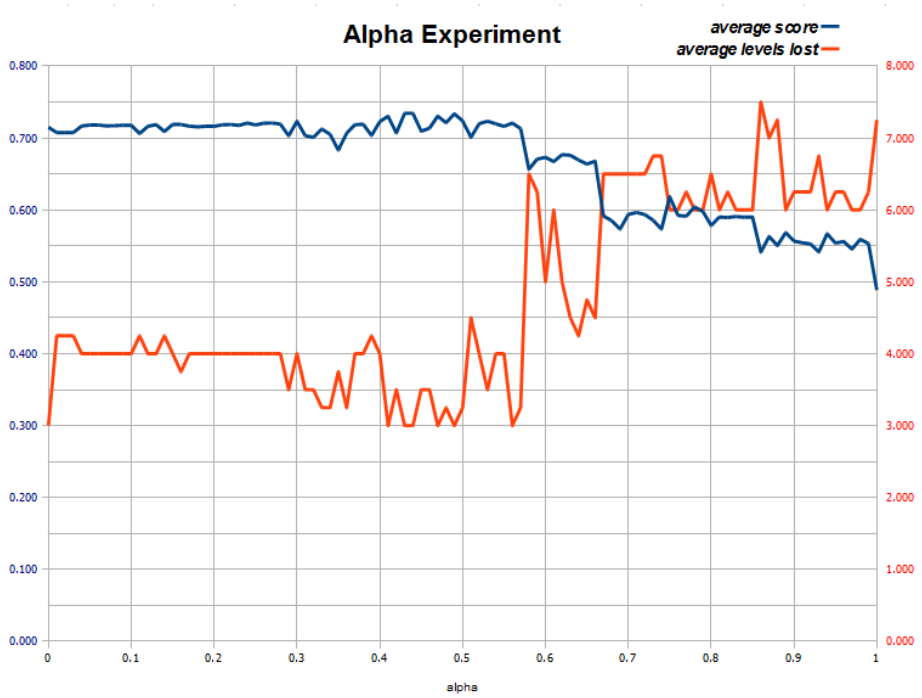
5.1 ניסוי 1 - מציאת אלפא אופטימלי

מוטיבציה לניסוי: בניסוי זה אנסה לענות על השאלות:
האם אחת היוריסטיקות "חזקה" מהיוריסטיקה השניה?
מה הנקודה בה השחקן משיג את הציון הטוב ביותר?

תיאור הניסוי : הרצתי את המשחק עם הפרמטרים:

- t - 1000 , שניה אחת לכל חיפוש.
- w - 100 , גודל אלומה מקסימלי הוא 100
- $alpha$ - עליו בעצם נערך הניסוי, הרצתי עם כל האלפות בין 0 ל-1 (כולל שניהם) בקפיצות של - 0.01 . כל אלפא היא שחקן

תוצאות הניסוי: סה"כ יש 101 נקודות, להלן גרף התוצאות, כאשר בגרף האדום רואים את כמות השלבים הממוצע שבהן השחקן הפסיד (נגמר השלב כי נגמרו לו החיים) ובגרף הכחול רואים את הציון הממוצע של כל שחקן.



הבחנות ואבחנות :

- נקודות המקסימום הגלובלית בגרף הן ב $\alpha = 0.44, 0.43$, שם הציון של השחקן הוא 0.734, עם ממוצע של 3 הפסדים במשחק.
- נקודת המינימום הגלובלית היא עבור $\alpha = 1$, שם הציון הוא 0.489 , עם למעלה מ-7 הפסדים בממוצע במשחק.
- על אף שנקודת המקסימום נמצאת בערך באמצע, ניתן בבירור לראות שיוריסטיקת המגדלים "חזקה" יותר מיוריסטיקת השביל. ניתן לראות כי קיבלנו ציונים קרובים למקסימלי כבר באלפות הנמוכות , וכי החלק הימני של הגרף שבו $\alpha > 0.6$ יש ירידה משמעותית בביצועים של השחקנים שמגיעה עד להכפלה ויותר של כמות השלבים שמפסיד השחקן. לשם השוואה, הממוצע של 33 הדגימות האחרונות הוא 0.573 והממוצע של 33 הדגימות הראשונות הוא 0.715.

סיכום ניסוי: לדעתי, ההסבר לכך שיוריסטיקת המגדלים חזקה היותר, וגם לכך שבלעדי יוריסטיקת השביל עדיין מגיעים לתוצאות טובות, היא שמאוד קל למצות את יוריסטיקת השביל. כדי למצות את יוריסטיקת השביל, צריך "לתפוס" מספר בודד של מקומות ממש טובים במפה (פינות שרואות הרבה שטח), ומשם יש המון המון מיקומים למגדלים שהערך היוריסטי שלהם זהה או דומה , ולכן כבר באלפא נמוך אם הצלחנו "לתפוס" את המיקומים הבודדים שהם דיי טובים (וזה סביר כי הערך היוריסטי שלהם גבוה, גם בהכפלה במקדם נמוך) מיצינו בסבירות גבוהה מאוד את יוריסטיקת המגדלים, וכנראה שגם כשלא בוחרים את המקומות הכי טובים במפה, אבל משפרים את המגדלים לרמה גבוהה, זה נותן תוצאות דומות.

5.2 ניסוי 2 - ניסוי זמן החיפוש

מוטיבציה לניסוי: בניסוי זה אני אנסה לענות על השאלות:
איך זמן הריצה ישפיע על היוריסטיקות השונות?
האם ככל שנגדיל את זמן החיפוש התוצאות ישתפרו?

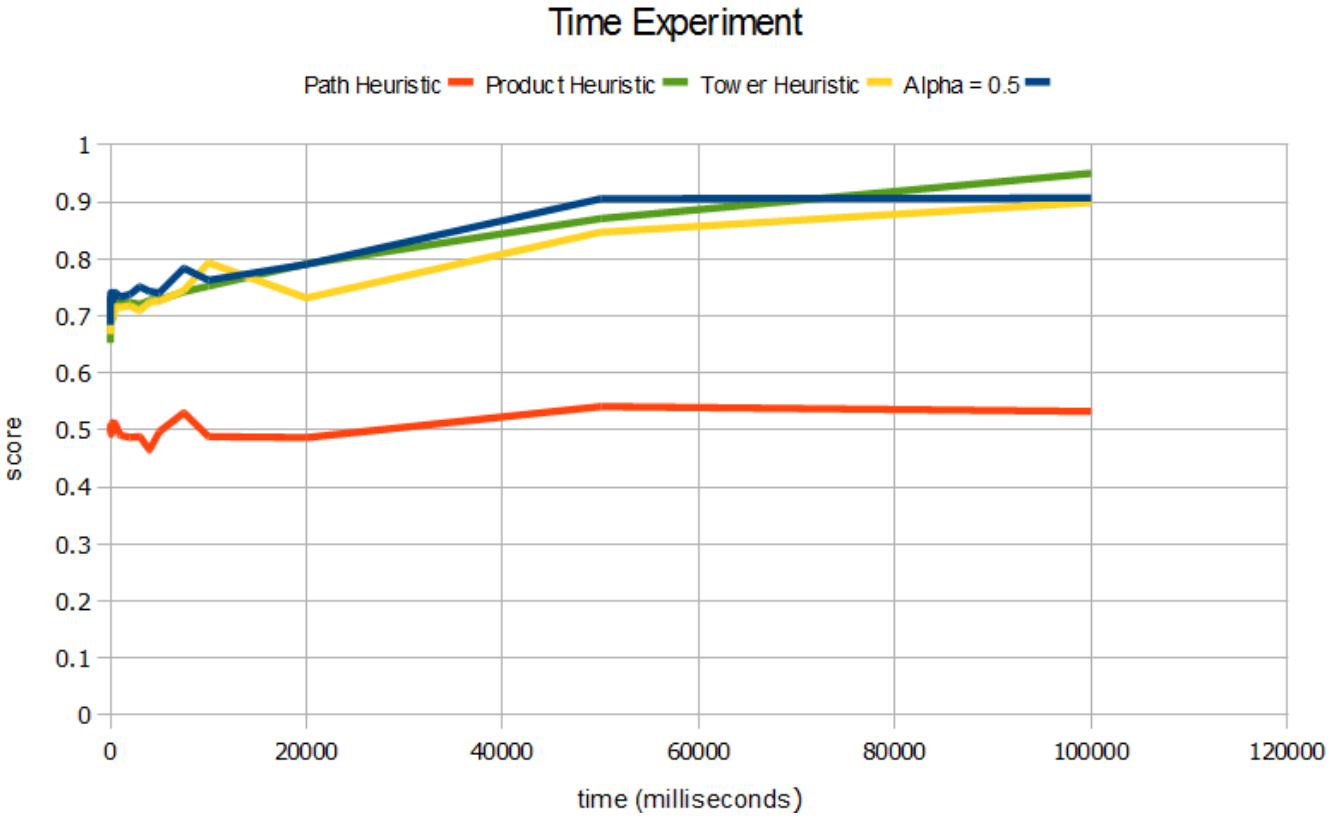
תיאור הניסוי : הרצתי את המשחק 4 פעמים עם הפרמטרים:

- t - הפרמטר עליו נעשה הניסוי

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

- w - 100 , גודל אלומה מקסימלי הוא 100
- $heuristic$ - $\alpha = 0, 0.5, 1$ ו $h_{PRODUCT}$, כלומר אנו בודקים את כל היוריסטיקות הטהורות, את הממושקלת עם פרמטר 0.5 ואת יוריסטקת המלבן.

תוצאות הניסוי: סה"כ יש 19 דגימות, להלן גרף התוצאות.



הבחנות ואבחנות :

- ניתן לראות שהגדלת זמן החיפוש, משפרת את הביצועים של השחקנים.
- ניתן לראות רוויה בזמנים הגדולים (כמה עשרות שניות לחיפוש).
- ניתן לראות כי יוריסטיקת המלבן, מצליחה להגיע לתוצאות הכי טובות בזמנים הגדולים, ולא ברור אם הגיע עדיין לרוויה (הגיעה לציון 0.95)
- ביוריסטיקת השביל ניתן לראות חיזוק למסקנה מהניסוי הקודם, שקל להגיע לרוויה.

סיכום ניסוי: בגלל האופי החמדני של חיפוש SAHC לוקלי (גם אם הוא אלומתי), ניתן היה לצפות שלכל גודל אלומה, נגיע לרוויה בזמנים הגדולים.
תוצאות הניסוי הפתיעו אותי, כיוון שציפיתי שבגלל גודל האלומה הקטן יחסית (100) נגיע לרוויה בשלב מאוד מוקדם, ושהגדלת הזמן לא תשנה את התוצאות.

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 ישתפרו?

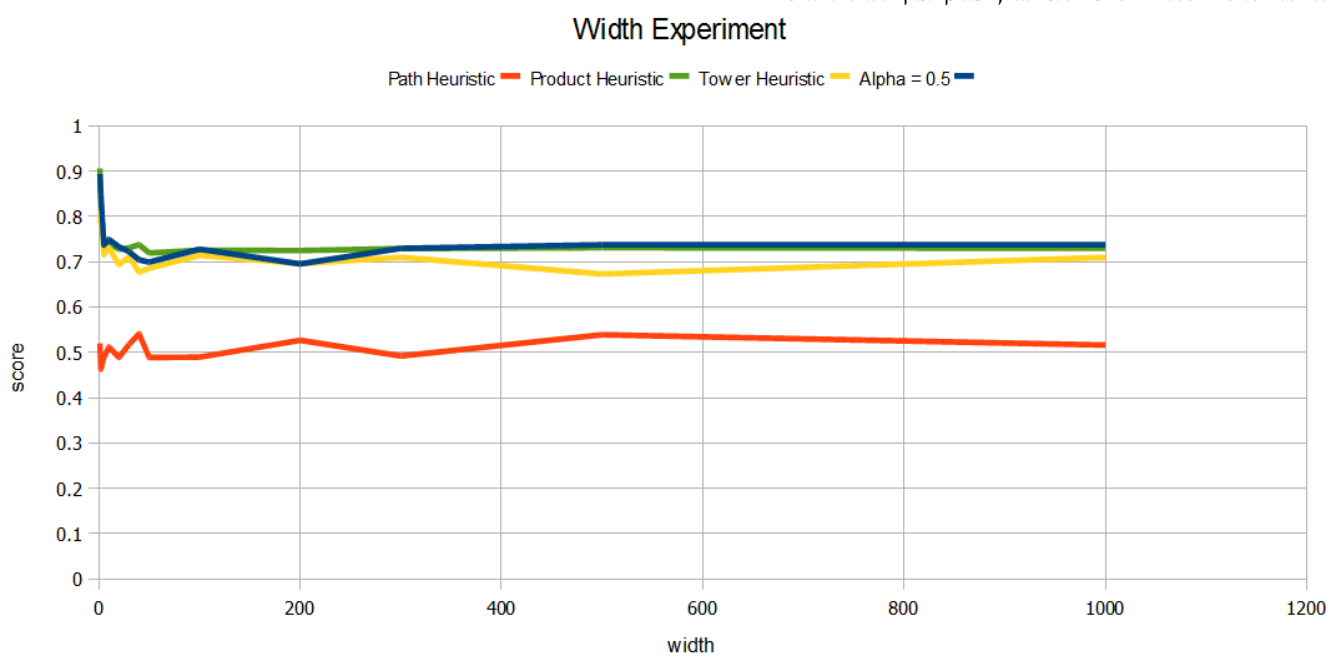
תיאור הניסוי : הרצתי את המשחק עם הפרמטרים:

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

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

- $heuristic$ - $\alpha = 0, 0.5, 1$ ו $h_{PRODUCT}$, כלומר אנו בודקים את כל היוריסטיקות הטהורות, את הממושקלת עם פרמטר 0.5 ואת יוריסטקת המלבן.

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
- יש קפיצה מאוד משמעותית ביכולת כאשר $w = 1$ כלומר, כשהחיפוש הוא חיפוש $SAHC$ (שלוש היוריסטיקות החזקות הגיעו לציון של כ-0.9)

סיכום הניסוי: תוצאות ניסוי זה היו מפתיעות, הצפייה שלי הייתה שכמו בניסוי הזמן, תהיה עלייה בציונים ככל שהאלומה גדלה, כיוון שכעת האלגוריתם חוקר את גרף המצבים יותר לרוחב - כלומר רואה יותר מצבים ומתחמק ממקסימות מקומיים. ניתן להסביר את התוצאות המפתיעות (גם את הקפיצה בערכים הנמוכים) 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.

5.4 ניסוי 4 - ניסוי רוחב אלומה , עם הרבה זמן

מוטיבציה לניסוי: ניסוי 5.3 הוא המוטיבציה לניסוי זה. בניסוי זה אני אנסה לענות על השאלה:
האם רוחב אלומה גדול ישפר את הביצועים, כשנותנים לאלגוריתם משמעותית יותר זמן?

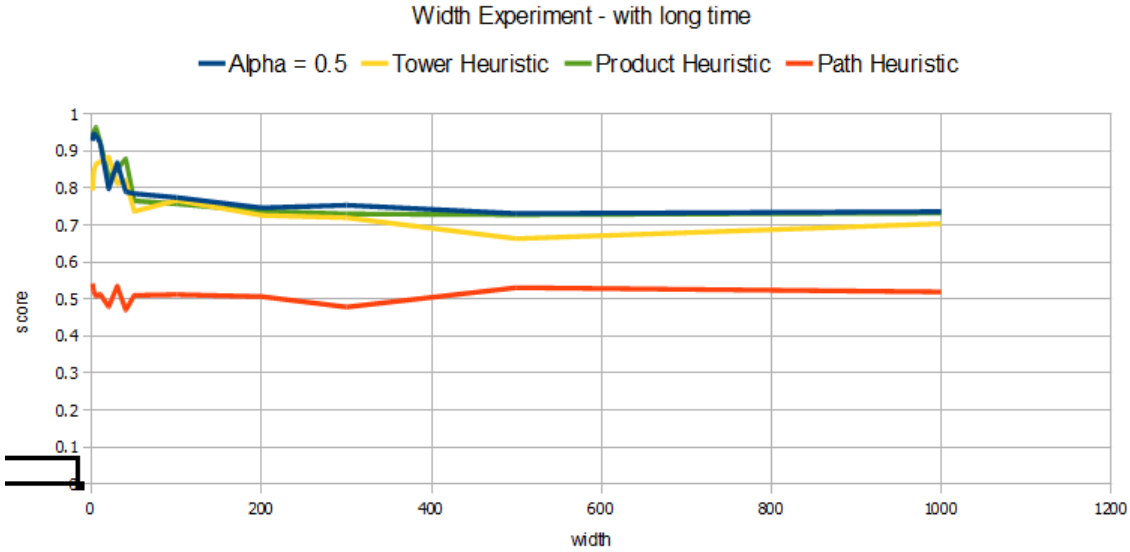
תיאור הניסוי : הרצתי את המשחק עם הפרמטרים:

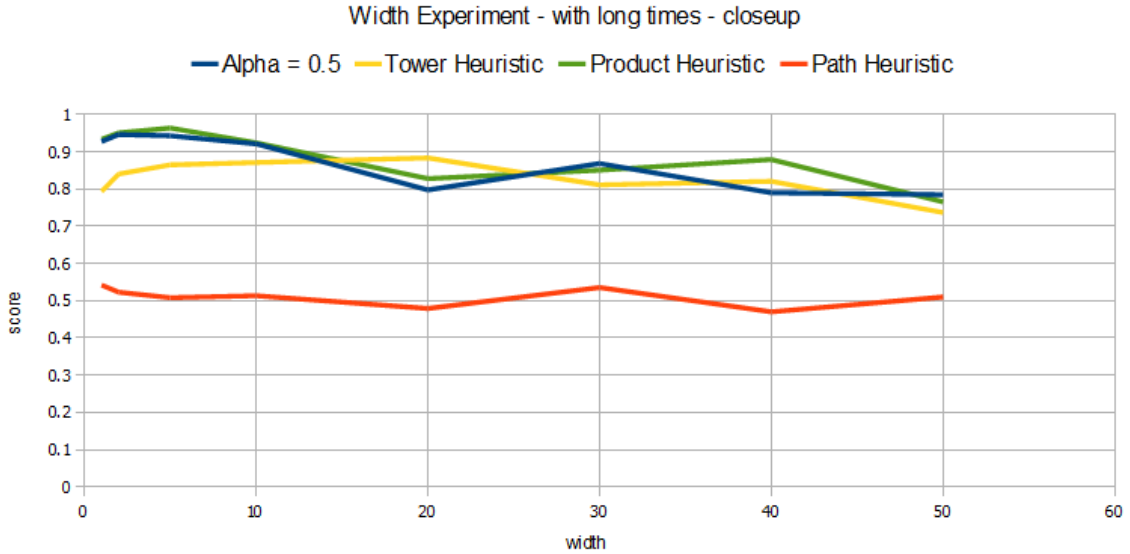
- t - 10000, 10 שניות לכל חיפוש (פי 10 מהניסוי הקודם)
- w - הפרמטר עליו נעשה הניסוי

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

- $heuristic$ - $\alpha = 0, 0.5, 1$ ו $h_{PRODUCT}$, כלומר אנו בודקים את כל היוריסטיקות הטהורות, את הממושקלת עם פרמטר 0.5 ואת יוריסטקת המלבן.

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





הבחנות ואבחנות :

- קיבלנו גרפים מאוד דומים לגרפים הקודמים.

- ניתן לראות בגרף הממוקד, שנקודות המקסימום הפעמים מתאגדות ב $w = 5$, ושהן גבוהות יותר מבניסוי הקודם - כאן הן מגיעות לשיא של 0.96 עבור יוריסטיקת המכפלה.

סיכום הניסוי: ראשית, ניסוי זה תומך בהשערה שהועלתה בניסוי הקודם כיוון שרואים שנקודת המקסימום זזה ימינה. שנית, ניתן לראות שצריך להגדיל את הזמן משמעותית (בסדרי גודל) כדי להגיע לתוצאות טובות ברוחבי אלומה די קטנים. תוצאה מעניינת של הניסוי היא הנקודה הראשונה, $w = 1$, כיוון שניתן לראות שהציון $0.9 <$, וגדול יותר מהציון בניסוי הקודם, כלומר אפילו בחיפוש הכי חמדני שיש - SAHC קלאסי, לא הצלחנו להגיע למקסימום מקומי בניסויים הקודמים (ואולי גם בזה) והציון עדיין משתפר, מה שמקרין על אופי גרף המצבים - הצמתים הטובים נמצאים מאוד עמוק בגרף.

6 סיכום

6.1 דיון בתוצאות

6.1.1 היוריסטיקות

- כפי שנראה בבירור הניסויים, יוריסטיקת השביל איננה מחזיקה בפני עצמה.
- עבור שאר היוריסטיקות שנבדקו (המגדלים, $\alpha = 0.5$, המלבן) התוצאות היו מאוד דומות בניסויים
- לא ברור אם הצלחנו להגיע לרוויה של יוריסטיקת המלבן, או שהיא יכולה לתת תוצאות טובות יותר עם יותר זמן

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 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 מה חסר וכיוונים להמשך המחקר

יוריסטיקות מתוחכמות:

- יוריסטיקה שמקבלת כקלט את הפרמטרים השונים של כל מפה (שטח אפקטיבי, אורך מסלול..) ומשקללת את הפרמטרים בחישוב.
- יוריסטיקה שמשקללת גם את איכות המגדל אותו היא משפרת (שפותרת את החסרון שהוצג ב 2.3.3 ביוריסטיקה המשוקללת)
- יוריסטיקה שמתחשבת גם במיקום הנוכחי של המפלצות : כשחקן אנושי כשאני רואה שמפלצת הצליחה לעבור עם מעט מאוד חיים ואין עוד מגדלים עד סוף המסלול אני בוחר לשים מגדל לקראת סוף המסלול בידיעה שזה לא מיקום אופטימלי אבל הוא יחסל את המפלצת שעברה (שפותרת את החסרון שהוצג ב 2.3.4)

יותר סוגי מפלצות :

- סוג מפלצות עם פחות חיים שנע יותר מהר
- מפלצות שעפות ־ כלומר לא עוברות בשביל ועפות בקו ישר לסוף המסלול
- סוג מפלצות שרגיש למגדל מסוים (מקבל ממנו יותר נזק, במקרה של הרבה סוגי מגדלים)

יותר סוגי מגדלים:

- מגדלים יותר חזקים, ושיורים יותר מהר, או שיש להם טווח יותר גדול
- להוסיף אופציה להעלות מגדלים רמות בכסף, בלי לבנות לידן מגדל
- להוסיף אופציה להגדיל למגדלים את הטווח, בכסף
- אסטרטגיות ירייה חדשות: למשל, המגדל יחשב את הסיכויים להרוג מפלצת לפי הרמה שלו, כמות השביל שהוא רואה, מיקום המפלצת, מהירותה, והחיים שלה, וכך יחליט את במי כדאי לירות

יותר סוגי מפות:

- להוסיף מפות עם שבילים מרובים (נקודות התחלה מרובות ו\או נקודות סיום מרובות)
- נקודות מיוחדות למיקום מגדלים , למשל "הר" שמגדיל טווח. "משבצת קסם" שמכפילה נזק

סוג אלגוריתם אחר:

- לעבור לחיפוש אלומה סטוכסטי, שבו כשנכניס איבר חדש, נוציא איבר בצורה הסתברותית ביחס לטיב שלו ושל האחרים באלומה.

בהרחבת המשחק יש טווח אינסופי ליצירתיות, שמאפשר לפתח את המשחק למרחב הרבה יותר גדול ליוריסטיקות מעניינות.

6.3 סיכום

במסגרת פרוייקט זה הצענו אלגוריתם anytime לפתרון בעיית ה - subset selection : האלגוריתם היה חיפוש אלומה לוקאלי כאשר הוא מתקדם כמו $SAHC$ האלגוריתם קיבל שתי פרמטרים: רוחב אלומה, וזמן ריצה, האלגוריתם קיבל גם יוריסטיקה.
בחרתי 2 יוריסטיקות טהורות שמייצגות את השיקולים המרכזיים שעומדים בפני שחקן אנושי שמשחק במשחק :

- מיקום המגדלים במקום אסטרטגי
- חיזוק מגדלים קיים ע"י הצבת מגדלים שכנים.

2 היוריסטיקות הנוספות שבחרתי הן הכלאה בין הטהורות : אחת היא קומבינציה ליניארית של שניהם, והשניה מכפילה ביניהם.
מהניסויים ניתן לקראות בבירור את הדומיננטיות של יוריסטיקת חיזוק המגדלים על פני יוריסטיקת המיקומים.
בנוסף, ניתן לראות שיוריסטיקת המכפלה מביאה תוצאות מאוד דומות ליוריסטיקה המשוקללת כאשר היוריסטיקה המשקוללת נמצאת עם פרמטר (0.5) שמאוד קרוב לפרמטר שנתן תוצאות מקסימליות שלה (0.44).
בניסויים הראשונים על רוחב אלומה גילינו שהגדלת רוחב האלומה פגע בביצועים, ושהביצועים הטובים ביותר היו של אלומה ברוחב 1 (כלומר חיפוש SAHC) , ההסבר שהוצע לתוצאה מפתיעה זו היה: כנראה שהזמן שאיתו הורץ האלגוריתם היה קטן מדי, וכשהגדלנו את האלומה, האלגוריתם לא הצליח להגיע לעומק גרף המצבים - כי רוב זמנו הושקע ברוחב.
ההיפוטזה שנבעה מניסוי זה הייתה שבגרף המצבים הנתון, המצבים הטובים נמצאים בעומק יחסית גדול, על מנת לנסות לאמת את ההיפוטזה חזרנו על הניסוי עם זמן גדול פי 10 . תוצאות ניסוי זה תמכו בהיפוטזה זו, והראו שאפילו בחיפוש SAHC קלאסי, צריך זמן גדול כדי להגיע למקסימום **מקומי** מה שמראה את העומק הגדול של המצבים הגדולים ואת האופי "ההרי" של גרף המצבים - כלומר שניתן ב"טיפוס" גרדיאנטי להגיע מאוד גבוה (ציונים מעל 0.9).
מסקנה נוספת מהניסוי היא שכדי לנצל את רוחב האלומה לביצועים, צריך להגדיל את זמן הריצה בסדרי גודל. כדי לאמת את ההיפוטזה סופית, הרצתי עוד ניסוי קטן שבו ניסיתי שוב "להזיז" את המקסימום כתלות ברוחב האלומה ע"י הגדלת זמן החיפוש.
הרצתי את המשחק עם יוריסטיקת המלבן וזמן 1,000,000 (1000 שניות לחיפוש). כמובן שהפעם האלגוריתם הצליח להגיע למקסימות מקומיות (ולהתכנס לפני סיום הזמן). שוב הצלחתי להזיז את רוחב האלומה האופטימלי, הפעם ל 50, כאשר במקסימום הגיע הציון ל 0.98!