**בינה מלאכותית ש.ב. 2**

חלק א'

נריץ 3 פעמים כאשר simple\_player מתחיל:

The winner is O random

The winner is X simple

The winner is O random

כעט 3 פעמים random\_player יתחיל:

The winner is O simple

The winner is X random

The winner is O simple

לפי התוצאות קיבלנו שכל שחקן ניצח 3 פעמים.

חלק ב'

1. נגדיר את היוריסטיקה הבאה:

כאשר:

1. הסבר על הפונקציות והפרמטרים לעיל:

ראשית נשים לב כי אם אין לנו יותר מטבעות על הלוח (שקול לכך שאין לנו יותר צעדים) כלומר בצעד זה נפסיד, ניתן למצב זה ערך יוריסטי של באופן שקול אם ליריב אין יותר מטבעות על הלוח כלומר אנו ננצח ולכן במצב זה ננצח ולכן ניתן לו ערך של .

לכל מצב אחר ניתן ערך בין 100 ל- 100- כאשר:

* 5% מהנקודות יורכבו מיתרון מטבעות (ערך חיובי עבור יתרון שלנו וערך שלישי עבור יתרון ליריב).
* 60% (רוב הניקוד) ניתן למצב בו יש לנו יתרון של פינות זאת משום שפינה שנתפסה לא תוכל להשתנות בעתיד ולכן היא קובעת עובדה בשטח שהיריב לא יכול לשנות – עבור כל פינה שנתפוס ערך זה יגדל ב25 נקודות ועבור כל פינה של היריב הערך יקטן ב25 נקודות (משום שלכל היותר ניתן לתפוס 4 פינות ולכן הניקוד המקסימלי עבור חלק זה הינו 100 לפני חישוב המשקל).
* 25% הינו קירבה לפינה – מסיבות דומות להסבר הסעיף הקודם קירבה לפינה (כלומר נוכחות באחת מ-3 המשבצות הקרובות לפינה) נחשבת לחסרון משום שהמצאות בנקודה זו תאפשר ליריב להשתלט על פינה (ולתת לו יתרון חשוב) לכן עבור כל משבצת שקרובה לפינה שבשליטתנו ערך זה יקטן ב- (באופן דומה לסעיף הקודם, ישנם 12 משבצות שקרובות לפינה) ועבור כל משבצת כזו בשליטת היריב הערך יגדל בהתאם.
* 10% האחרונים יחושבו על ידי הפרש בין כמות הצעדים העתידיים שמצב זה פותח עבורנו וכמות הצעדים האפשריים שמצב זה פותח עבור היריב.

אנו צופים כי הגדרה זו תשפר את הביצועים על פני השחקן הפשוט.

1. מומש.
2. תוצאות:

The winner is X better

The winner is X better

The winner is X better

The winner is O better

The winner is O better

The winner is O better

כפי שציפינו, ההגדרה שלנו נותנת ביצועים טובים יותר מהביצועים של השחקן הפשוט.

כאשר הרצנו עם verbose = y ראינו כי השחקן שלנו תמיד תופס את רוב הפינות בלוח ולכן לשחקן הפשוט אין אפשרות לנצח.

חלק ד

אנו מצפים שיהיה הבדלים בין ביצועי min max ל alpha\_beta כאשר זמן החישוב מוגבל. אם ל2 השחקנים זמן חישוב לא מוגבל שניהם פועלים בצורה זהה. אלפא בטא הוא רק שיפור ליעילות והשיפור נכנס לתוקף במשטר any time שבו 2 השחקנים יכולים לבצע חישובים עד שנגמר להם הזמן – במקרה כזה alpha beta יצליח "לראות" עמוק יותר בעץ החיפוש ולכן יש לו יתרון על min\_max

חלק ה

1. מתוך כל השחקנים אנו מצפים לביצועים הטובים ביותר מalpha\_beta\_player .

ההסתברות ש random\_player יבצע את ההחלטה הנכונה בכל מצב היא לכל הפחות כאשר D הוא עומק העץ ומספר האפשרויות גדול מ4 בממוצע .

Simple\_player מסתכל רק לעומק אחד בעץ החיפוש ולכן כל שחקן שמסתכל יותר מעומק 1 יהיה טוב יותר.

Better\_player גם הוא מסתכל לעומק אחד אך הוא משקלל יותר פרמטרים להיוריסטיקה שלו ובמבחן אמפירי הוא תמיד מנצח (20 משחקים) לא משנה מי מתחיל.

Min\_max מסתכל לעומק יותר מ1 עם היוריסטיקה של better\_player ולכן יש לו יתרון עליו.

Alpha\_beta\_player מפתח בין 20% ל50% יותר עומקים (כלומר 20% עד 50% יותר איטרציות על iterative deepening ) מmin\_max . במבחן אמפירי התקבל כי alpha\_beta מנצח בכל פעם את min\_max לא משנה מי מתחיל (כ10 משחקים).

\*מספר הפעמים שהמשחקים רצו אינו משנה מפני שהשחקנים דטרמיניסטים ורק תנאי ההתחלה (מי מתחיל) משפיעים על התוצאה.

2. selective deepening for min\_max

השיפור טמון בעובדה שענפים שלמים בעץ החיפוש לא ייפותחו וזה יפנה זמן לפיתוח ענפים אחרים לעומק רב יותר. הענפים שלא ייפתוחו הם ענפים שהאלגוריתם יכול לשערך די בדיוק .

ניתן לממש במספר דרכים:

1. כאשר ערך היוריסטיקה נשאר קבוע יחסית עבור פיתוח ענף מסויים – נפסיק לבצע העמקות. ובניגוד לכך, אם נראה שערך יוריסטי של מצב כלשהו גבוה \ נמוך מהערך של אבא שלו בלפחות C נבחר לפתח בכיוון הזה.
2. Pre-processing חישוב ושמירת ערך יוריסטי למצבים נפוצים לפני הריצה האמיתית.
3. Caching שמירה של טבלה המכילה טאפל (מצב , ערך , עומק ) ובכך בכל פעם שנבצע deepening מההתחלה (זהו תהליך איטרטיבי ) , לא נצטרך לחשב חלקים ממנו – פשוט נקח את הערך מה cache ב O(1)

Time for step

באופן כללי – פונקציה כזאת מאפשרת לנהל את הזמן בצורה טובה יותר. למשל בשילוב עם selective deepening אם נתקל במצב שכל ענף אינו דורש העמקה נוספת לפי פרמטר C שהוזכר בסעיף הקודם, נוכל לסיים את השלב "מוקדם יותר" ולהעניק את הזמן לצעד הבא. **לא בטוח לגבי זה**

חלק ו'

1. התוכנה משתמשת בנוסף לאסטרטגיות שאנו מימשנו גם בספר מהלכים שמכיל מידע שנאסף מראש ולכל מצב מכיל את התגובה הטובה ביותר וכמו כן התוכנה משתמשת במיון צעדים כך שאין צורך לכל מצב לחשב מחדש את ערך פונקציות הUtilty אלא פשוט בעזרת גישה לhash table ניתן לקבל את הערך שחושב מראש וכך לחסוך זמן חיפוש ולהעמיק יותר בעץ החיפוש.
   1. Opening book הינו קובץ המכיל מספר גדול מאוד של משחקים שבוצעו בעבר והניתוח שלהם בנוסף להערכה עבור הצעדים שניתן לבצע בתגובה לכל צעד שבוצע במשחקים אלו. המידע שמתקבל מקובץ זה הינו מידע שנאסף טרום המשחק ולכן השחקן שלנו מגיע מיודע יותר ויש לו לא רק את פונקציית הUtility שתכנתנו לו אלא בנוסף יש לו מידע ממשחקים שבוצעו בעבר.
   2. חמשת המשחקים הפופולריים ביותר:

13493 +d3-c5+f6-f5+e6-e3+c3-f3+c4-b4

13493 +d3-c5+f6-e3+c3-f5+e6-f3+c4-b4

13493 +d3-c5+e6-f5+f6-e3+c3-f3+c4-b4

7432 +d3-c5+f6-f5+e6-e3+d6-f7+g6-e7

7432 +d3-c5+e6-f5+f6-e3+d6-f7+g6-e7

* 1. מומש.
  2. החיסרון בשיטה זו היא שבנינו את הספר לפי הפתיחות הכי פופולריות אבל לא בהכרח מהלך פופולרי מוביל לניצחון, אנחנו לא משתמשים בפונקציית הUtil שכן נותנת לנו אינדיקציה לגבי הקרבה שלנו לניצחון.
  3. דרך נוספת להשתמש בקובץ המשחקים הינה לחלק את הקובץ לשני קבצים שונים, בקובץ הראשון יהיו רק משחקים בהם השחקן שמתחיל מנצח ובקובץ השני רק משחקים בהם השחקן השני מנצח. לאחר מכן נבצע את אותה פעולה על שני הקבצים כמו שעשינו בחלק הנוכחי. במשחק עצמו נבחר את המהלכים שלנו לפי הקובץ שמתאים לנו (אם אנו מתחילים נבחר מהקובץ בו השחקן שמתחיל מנצח ואם אנו שניים אז מהקובץ השני).