**בינה מלאכותית לשח רגלים**

­­

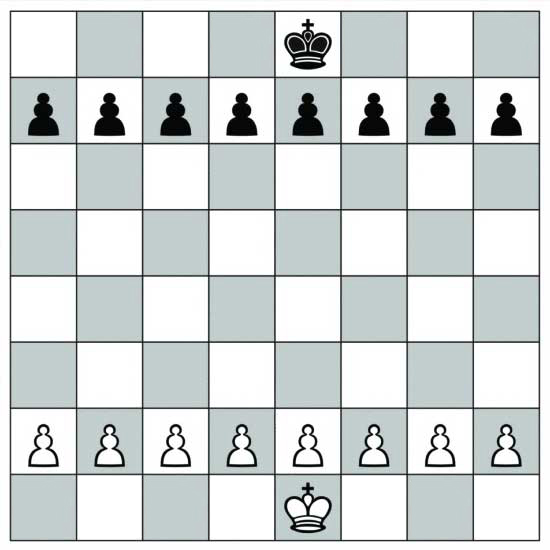
מגישים:

אילן גודיק – 316315332

יובל אלפסי – 318401015

מנחה:

מר שי בושינסקי

תאריך הגשה:

13 ליוני 2016

**תוכן עניינים**

מבוא – עמוד \_

אלגוריתם – עמוד \_

ארכיטקטורה – עמוד \_

סטטיסטיקה ובקרת איכות – עמוד \_

סיכום – עמוד \_

**מבוא**

השתתפנו בהאקתון CRI שהתקיים בתאריך 2-3.6.2016, בו בנינו בינה מלאכותית שמשחקת שח רגלים – Peshka. את התוכנה שלנו בנינו על בסיס תוכנה קיימת – על בסיס הקוד הפתוח של מנוע השחמט Stockfish.

המנוע שלנו ל-Peshka נבנה בהשראת AlphaGo – הבינה המלאכותית שניצחה את אלוף העולם ב- Go. כמוהם, התבססנו על עץ חיפוש מונטה-קרלו בעל חיפושים חכמים במקום רנדומיים.

טרם תחילת ההאקתון, קראנו את המאמר על AlphaGo ביסודיות, והבנו לעומק כיצד המהנדסים שבחברת הבת של Google פיתחו את הבינה המלאכותית שהביסה את אלוף העולם. לאחר מכן, חשבנו להתחיל למממש את משחק השחמט מהתחלה בשפת תכנות High-Level שנוחה לנו, ובנוסף, להבין כיצד עובד פרוטוקול ה-UCI, אך במהרה נוכחנו להבין שזוהי אופציה לא ריאלית. לפיכך, שינינו כיוון, והחלטנו לפתח את המנוע שלנו על בסיס מנוע קיים – הורדנו את מנוע השחמט של Stockfish, בו היינו צריכים להתמודד עם עשרות אלפי שורות קוד. למדנו ביסודיות כיצד המנוע פועל, היכן הוא מבצע קלט-פלט, כיצד הוא מפעיל את החיפוש שלו והיכן נמצאת פונקציית היוריסטיקה שלו. להאקתון הגענו עם תכנית מפורטת היכן להתחיל ומה לעשות.

במשך עשרים וארבע השעות של ההאקתון שינינו את הקרביים של המנוע של Stockfish ובנינו בינה מלאכותית שמשחקת שח רגלים באופן טוב למדי.

**אלגוריתם**

עץ חיפוש מונטה-קרלו עם צורת חיפוש חכמה הוא שמנחה את הבינה המלאכותית שלנו. נסביר כיצד הוא פועל.

מדובר בעץ חיפוש בו הצמתים הם לוחות אפשריים וקשתות הן מהלכים אפשריים למשחק. מדובר בעץ חיפוש אסימטרי שמתמקד באזורים 'מעניינים' של המשחק יותר מאשר באזורים הפחות סבירים שנגיע אליהם.

העץ מתפתח לעומק ולרוחב באופן איטרטיבי.

תחילה, בוחרים עלה 'מעניין' בהתאם לסטטיסטיקה שנצברה בכל צומת – Selection

לאחר מכן פותחים בן 'מעניין' שלו ומוסיפים אותו לעץ – Expansion

עושים סימולציית משחק חכמה של שח רגלים עד שמגיעים לניצחון, תיקו או הפסד – Rollout

מעדכנים לאחור את הסטטיסטיקה של המשחק שנצברה – Back Propagation

Selection:

עלינו לבחור צומת 'עלה' מעניין – צומת שלא הוספנו עוד את כל בניו לעץ. לצורך כך, עלינו לבחור קבוע Exploration -Exploitation – כמה diversity נרצה, כמה נרצה להתעניין בצמתים שחקרנו מעט לעומת צמתים שנראים מעניינים וחקרנו הרבה.

השתמשנו בנוסחאת ה-UCT לצורך כך. מתחילים משורש העץ, ומשם מתגלגלים למטה תוך בחירת הצומת בעל ערך ה-UCT המקסימאלי. בהתאם לסטטיסטיקה שנצברה, ערך ה-UCT של צומת יהיה:

כאשר:

w – מספר הניצחונות שבצומת

n – מספר הסימולציות שנעשו בצומת

c – קבוע המחליט כמה diversity נרצה. exploration-exploitation.

t – סך כל מספר ה-simulations שנעשו.

כפי שניתן לראות – ככל שמספר הניצחונות היחסי של צומת גדול יותר, כך ערך ה-UCT שלו גבוה יותר, כמו כן, ככל שעשינו יותר סימולציות באופן כללי על שאר הצמתים ועל הצומת הספציפי עשינו מעט סימולציות, ערך ה-UCT של הצומת יהיה יותר גבוה. בנוסף, הקבוע c קובע כמה משקל חשיבות יהיה לנו בחישוב לכמה שהצומת מעניין, ל-exploitation, , וכמה משקל יינתן ל-exploration, ל-diversity, .

אפוא, בחירה של צומת 'עלה' מעניין נעשתה באופן איטרטיבי מהשורש ומטה תוך שימוש באלגוריתם UCT שמישקל כמה צומת מעניין תוך התחשבות בקבוע diversity.

Expansion:

לאחר שבחרנו צומת מעניין בעץ, נרצה לפתח את העץ ממנו. לצורך כך, ניבחר בן 'מעניין' שלו אותו נוסיף לעץ וממנו נעשה סימולציית משחק.

בעצי מונטה קרלו רגילים בוחרים בן לפיתוח באופן רנדומי. לעומת זאת, אנחנו שאבנו השראה ממה שעשו ב- AlphaGo, שם עשו Expansion באופן חכם באמצעות רשת עצבים מלאכותית שאימנו מראש. לעומת זאת, אנחנו הנחינו את הפתיחה של הבן באמצעות ההיוריסטיקה של Stockfish, שהדריכה אותנו איזה בנים כדאי לפתוח ולהוסיף לעץ, מי מהם 'מעניין'.

בהינתן בן שרצינו לפתוח, ייצרנו את הבנים שלו וחישבנו עבורם היוריסטיקה. את ערכי היוריסטיקה נירמלנו לערכים אי שליליים, ועליהם הפעלנו Rollet Wheal Selection – 'גלגל המזל' כפי שנלמד בהרצאה – ללוחות עם ערכי היוריסטיקה גבוהים יש יותר סיכוי להיבחר לפתיחה בעץ.

Rollout:

לאחר שפתחנו בן, עלינו לסמלץ עבורו משחק עד לניצחון, הפסד או פט. בעצי מונטה קרלו רגילים מסמלצים משחק באופן רנדומי. לעומת זאת, ב-AlphaGo הם סימלצו משחק 'חכם' שהודרך על ידי רשת נוירונים שאומנה מראש. אצלנו, את המשחק הנחתה היוריסטיקה של Stockfish, שאמרה איזה צומת הכי טוב שכדאי לשחק דרכו. כך, למשחק שסומלץ היתה משמעות מבחינת המשחק.

Back Propagtion:

בהתאם לתוצאת ה-Rollout – הפסד ניצחון או פט, ובהתאם לשחקן שמשחק – שחור או לבן, מעדכנים עד לשורש העץ את הסטטיסטיקה שניצברה עבור המשחק. סטטיסטיקה זו תשמש את האיטרציות הבאות לצורך חישוב ערכי ה-UCT וקביעה אילו לוחות הם 'מעניינים', את מי כדאי לפתח ואת מי לא.

**ארכיטקטורה**

המנוע Stockfish הקיים אותו הרחבנו רשום ב C++. נדרשים מהמנוע ביצועים מהירים ככל שניתן. יש לציין שכאשר המנוע מחפש מהלך לביצוע הוא לא מבצע שום הקצאה דינאמית. אין שום malloc, new או free.

פעולת המנוע שלנו:

MainTherad::search:

שינינו את פעולת החיפוש של Stockfish מהתחלת חיפוש של minimax לתחילת חיפוש מהסוג שלנו – של חיפוש מונטה-קרלו

**void** MainThread::search() {  
 mcts\_main\_search();  
}

mcts\_chess\_playing:

קובץ קוד המכיל הרחבות לאחזור מידע לגבי הפוזיציה הנוכחית של הלוח.

mcts\_tablebases:

לוגיקה המנהלת את ה-tablebases: (\*להסביר קצת על tablebase.. \*)

mcts:

דג

**סטטיסטיקה ובקרת איכות**

דגכדגכ

**סיכום**

כחודש לפני ההאקתון שי העלה בפנינו את הרעיון לקחת חלק בהאקתון ולפתח בו בינה מלאכותית לשח רגלים תוך שימוש ברעיונות שדומים לבינה המלאכותית של AlphaGo. לקחנו משימה זו על עצמינו ונכנסנו לנבכיו העמוקים של מנוע השח-מט של Stockfish.

בהאקתון עצמו ניתנו לנו אך ורק עשרים וארבע שעות לפתח את מנוע השח-מט שלנו, זמן מועט וקצר ביותר, ולמרבה השמחה, עמדנו ביעד. היה מדובר בשעות על גבי שעות של למידה של המנוע של Stockfish על בוריו, קריאה יסודית של המאמר של AlphaGo, ובזמן ההאקתון פיתוח דקדקני של המנוע שלנו.

לסיכומו של דבר, הצלחנו להרים מנוע טוב למדי שפועל ומשחק.