חיפוש תת-אופטימלי חסום בעץ משחק מרובה שחקנים

סקר ספרות ותכנון המחקר

מנחה: דור עצמון

שחר מרץ 208240275

פלג ביטון 203842703

אסף זקס 302329693

עומר נגר 307937714

סקר ספרות

תקציר מנהלים –

המחקר עוסק בחיפוש תת-אופטימלי חסום בעצי משחק מרובי שחקנים. נרצה להגדיר את התנאים הדרושים על מנת שאלגוריתם חיפוש, בעץ משחק מרובה שחקנים, יחזיר פתרון תת-אופטימלי חסום - שונה עד כדי קבוע מהערך האופטימלי. באמצעות התנאים נממש גרסאות תת-אופטימליות לפתרונות אופטימליים מוכרים, נוכיח את נכונות אלגוריתמים אלו ונבצע ניסויים במטרה לבחון את ייעילותם.

למחקר ערך בתחומים שונים, בניהם תורת המשחקים, קבלת החלטות, סטטיסטיקה, פילוסופיה, כלכלה, רובוטיקה ואבטחה. אף על פי כן, המחקר מהווה המשך ישיר של מחקר של Atzmon (2018) החוקר את אותה הבעיה עבור משחקים לשני שחקנים, שבו מצוטט רק מאמר אחד מחמש השנים האחרונות, כלומר מדובר בנושא שלא נחקר בעבר ולא בוצעו מחקרים בכיוון הזה.

בסקירת הרקע נציג רקע על תחום החיפוש בעצי משחק, נבחן עקרונות בסיס המתקיימים בחיפושים אלו ובפתרונות שונים לסיבוכיות הגבוהה של אלגוריתמי חיפוש נאיבים בעצי משחק, בדגש על גיזום קודקודים.

בסקירת ספרות רלוונטית נציג מגוון פתרונות אלגוריתמיים אופטמליים לחיפוש בעצי משחק לשניים או יותר שחקנים, במטרה להכיר את התנאים החלים בפתרונות אלו ולהגדירם תחת התאמות כתנאים לביצוע גיזום תת-אופטימלי חסום.

סקירת רקע –

לדברי Luckhardt and Irani (1986), חיפוש בעצי משחק הוא אחד מהתחומים הראשונים שנחקרו בעולם הבינה המלאכותית. נהוג למדל משחק כסט סופי של מצבים המכיל בתוכו תת-סט של מצבים סופיים בעלי ערך תועלת הנקבע על ידי פונקציית הערכה (מהם לא ניתן להתקדם ולמעשה בהם המשחק מסתיים), וסט נוסף של פעולות למעבר ממצב למצב.

לטענתם, רוב המחקר עוסק במשחקים בהם מספר הפעולות השונות שניתן לבצע בכל מצב הוא סופי, וכך גם מספר הפעולות עד להגעה למצב סופי מכל מצב. הבעיה מאופיינת בצורה של עץ, כאשר כל קודקוד בעץ מתאר את מצב המשחק בתורו של משתתף מסויים, ממנו יוצאות קשתות המייצגות פעולות חוקיות שיכול לבצע המשתתף ויובילו לשינוי המצב במשחק. כל אחד מקודקודי הבן מייצגים מצבים חוקיים, הנגזרים מביצוע הפעולה שבקשת על המצב שבקודקוד האב. במעבר בין קודקוד אב לבן יתקדם התור במשחק לשחקן הבא בסדר.

על מנת להבין את הבסיס למחקר בתחום נסתכל קודם על משחקים סופיים לשני שחקנים עם סכום נקודות קבוע, בדגש על משחקי סכום אפס - משחקים בהם בכל מהלך, הרווח (או ההפסד) של שחקן אחד שווה להפסד (או לרווח) של השחקן השני. היכרות עם עקרונות אלו תסייע לנו בבואנו לעסוק בהמשך במשחקים מרובי משתתפים.

עקרון המינימקס שהציג Wald (1945) הוא עיקרון בסיס בכל הנוגע לקבלת החלטות במשחקי סכום אפס. על פי עקרון זה, בכל תור השחקן יבצע את המהלך בעל התועלת הגדולה ביותר עבורו, תוך הנחה כי השחקן השני יעשה בדיוק את אותו הדבר בתורו. אלגוריתם המינימקס הוא אלגוריתם נאיבי מבוסס חיפוש לעומק, הפועל על פי עיקרון זה ומוצא את ערך המינימקס של עץ משחק (נקרא גם ערך שיווי המשקל של העץ), ומבטא את התועלת הסופית עבור השחקן שבשורש העץ. תוצאת החיפוש הינה האסטרטגיה האופטימלית עבור שחקן השורש למיקסום הרווח שלו. האלגוריתם מבטיח פתרון אופטימלי שכן הוא חוקר את כל קודקודי העץ.

על פי Knuth and Moore (1975), במקרים רבים עצי משחק הם גדולים מאוד – כתלות במספר המהלכים עד לסיום המשחק ובמספר הפעולות האפשריות ששחקן יכול לבצע בתורו. על כן, ביצוע

חיפוש נאיבי על כלל הקודקודים במטרה להחזיר את ערך המינימקס אינו תמיד פיזיבילי. לאורך השנים פותחו טכניקות שונות במטרה להתגבר על קושי זה, בניהן טכניקת הαβ, שמטרתה להאיץ את תהליך החיפוש ללא איבוד של מידע. במילים אחרות, הערך שיוחזר ע"י שימוש בטכניקה זו זהה לערך שיחזור כתוצאה משימוש באלגוריתם המינימקס. בפועל, אלגוריתמים מבוססי αβ חוקרים את העץ באופן דומה לאלגוריתם המינימקס ומבצעים גיזום לקודקודים בעץ כאשר ניתן לדעת באופן ודאי שאין צורך בפיתוחם, שכן קיימת פעולה חוקית אחרת אשר תוביל את השחקן שבשורש העץ למצב עדיף מבחינתו. ע"י גיזום קודקוד מסויים האלגוריתם למעשה מדלג ואינו חוקר את תת-העץ שקודקוד זה מהווה השורש שלו, ומכך מתקיים חיפוש מהיר יותר מבלי לפגוע באופטימליות הפתרון.

גם לאחר החלת חוקי הגזירה הללו, טכניקה זו או אחרות הדומות לה אינן מבצעות מספיק גיזומים על מנת להקל מספיק על מציאת הפתרון. אחת הדרכים להתגבר על בעיה זו היא להקל על הפתרון האופטימלי ולאפשר פתרונות תת-אופטימליים. Atzmon (2018) מציע ליצור טרייד-אוף בין זמן הריצה לאופטימליות הפתרון על ידי אפשור של גיזום נרחב יותר מזה המתאפשר בחיפוש אחר פתרון אופטמלי. מחקרו מגדיר תנאים לתכנון אלגוריתמים תת-אופטימליים חסומים, אשר יקבלו מהמשתמש ערך ϵ ויחזירו פתרון תת-אופטימלי לעץ משחק שערכו רחוק בעד ϵ מערך המינימקס האמיתי של העץ. בפועל, המאמר מציע שיטה להרחבת חוקי גזירה של אלגוריתמים מוכרים כך שיחזירו פתרונות תת-אופטימליים מסוג זה. על פי עקרונות אלו ככל שערכו של ϵ יהיה גדול יותר תתאפשר גזירה נרחבת יותר בעץ.

כך, בעזרת אלגוריתמי משחק תת-אופטימליים חסומים, ניתן לבצע גיזומים רבים יותר מאלגורתמים אופטימליים וכפועל יוצא להשיג אסטרטגית משחק בצורה מהירה יותר. אולם, דבר זה עלול לפגוע באיכות הפתרון החוזר מהאלגוריתם ומכאן שקיים טרייד-אוף בין איכות הפתרון לזמן חישוב הפתרון. בעוד המחקר עוסק בפתרונות תת-אופטימליים חסומים בעצי משחק לשני שחקנים, בפרוייקט נרצה להציע פתרון דומה עבור משחקים מרובי משתתפים, המבוסס על פתרונות אופטמליים מוכרים.

Luckhardt and Irani (1986) הציעו את אלגוריתם max^n, המהווה המקביל האלגוריתמי של אלגוריתם המינימקס למשחקים מרובי משתתפים. האלגוריתם מתאים למשחקים עם סכום נקודות קבוע ולא קבוע, בהם מתקיים שיתוף פעולה בין שחקנים וגם לכאלו בהם כל שחקן משחק עבור עצמו. ניתן להשתמש בmax^n גם במשחקים עם אלמנט של מזל ובכאלו ללא מידע מושלם.

בדומה לאלגוריתם המינימקס, בכל קודקוד לאורך החיפוש נניח כי השחקן שזהו תורו יבחר את הפעולה שתמקסם את הרווח עבורו. בשונה מאלגוריתם המינימקס המיועד למשחקי סכום אפס בשני שחקנים, לא נוכל לייצג את הרווח הצפוי לכלל השחקנים באמצעות מספר בודד ונעבור לייצוג ווקטורי. האלגוריתם שמציגים החוקרים בסיסי ונאיבי (שכן על אף שמאפשר "לגזום" כניסות בווקטור הרווח של קודקוד מסויים, הוא עדיין מחוייב לחקור את כל הקודקודים) ותוצאתו היא סט של אסטרטגיות עבור כל שחקן, המוכח כווקטור שיווי המשקל, שכן כל חריגה של שחקן מהאסטרטגיה שהתקבלה עבורו תוביל לתוצאה קטנה או שווה לזו שיקבל תחת האסטרטגיה האופטימלית שהחזיר האלגוריתם. האלגוריתם מהווה אבן דרך בתכנון טכניקות חיפוש לא נאיביות למציאת פתרון אופטימלי לבעיות חיפוש בעצי משחק מרובי משתתפים, בניהן טכניקות המאפשרות גיזום קודקודים שלמים.

לטובת הפרוייקט המחקרי אותו נבצע, נסקור ספרות בדבר אלגוריתמים מבוססי max^n המאפשרים גיזום קודקודים (מבלי לפגוע באופטימליות הפתרון). בנוסף, נסקור אלגוריתמים נוספים המבצעים תחת הנחות שונות רדוקציה מבעיית חיפוש למשחקים מרובי משתתפים, לעצי חיפוש בעלי מבנה זהה לאלו שנפרשים על ידי אלגוריתם המינימקס במשחקים בשני משתתפים, עליהם נוכל לבצע חיפוש בשיטת αβ. באמצעות הידע שנרכוש ננסה להגדיר תנאים באמצעותם נתכנן אלגוריתמים חדשים המבצעים חיפוש תת-אופטימלי חסום עבור משחקים מרובי משתתפים.

סקירת ספרות רלוונטית –

על מנת להגדיר תנאים לביצוע גיזום תת-אופטימלי חסום בעצי משחק מרובי משתתפים, נרצה ראשית ללמוד על אסטרטיות שונות הקיימות באלגורתמים אופטימליים שונים, ללמוד על ההנחות שחייבות להתקיים במשחק על מנת להשתמש באלגוריתם ועל התנאים הנדרשים לביצוע גיזום אופטימלי של קודקדים תחת הנחות אלו במטרה להגדיר חוקים אלו מחדש עבור גיזום תת-אופטימלי חסום.

נסתכל ראשית על גיזום אופטימלי בעצים מסוג max^n. Sturtevant and Korf (2000) סקרו ביצד תחת הנחות שונות ניתן לבצע חלק מגיזומי αβ על עצים אלו. על פי ממצאי החוקרים, אם מאופי המשחק ידועים לנו סך הנקודות המקסימלי עבור כל השחקים יחד (maxsum) וערך הנקודות המקסימלי עבור שחקן בודד (maxp) נוכל לגזום.

ראשית נוכל לבצע גיזום מיידי (immediate pruning) – גיזום טריוויאלי, נגזום כאשר השחקן הנוכחי מקבל את מספר הנקודות המקסימלי האפשרי. נוכל גם לבצע גיזום רדוד (shallow pruning) – גיזום קודקוד המתאפשר בזכות מידע מקודקוד "דוד" שפותח קודם לכן. סוג נוסף של גיזום המתקיים בטכניקת הαβ אך אינו מתאפשר בגרסה רבת המשתתפים הוא גיזום עמוק (Deep pruning) – גיזום של קודקוד כלשהו על סמך מידע מאב קדמון שלו. החוקרים מראים כי טכניקה זו אינה מתאפשרת בעצים מסוג max^n מאחר וקודקוד שנגזר על ידי גזירה זו אמנם איננו מכיל את ערך שיווי המשקל של העץ, אך גזירתו יכולה להוביל לזיהוי קודקוד שגוי כתוצאת האלגוריתם.

טכניקה נוספת שהציגו החוקרים ועושה שימוש בmaxsum maxpו היא Depth-First Branch-and-Bound (DFBnB), המניחה כי בחוקי המשחק ניתן למצוא פונקציה יוריסטית מונוטונית לחישוב המרחק מניצחון (ניתן למצוא כזו במשחקי קלפים רבים). על פי הטכניקה נוכל לבצע גיזומים דומים לגיזום מיידי וגיזום רדוד על ידי השוואת ערך היוריסטיקה למידע שאספנו עד כה ולדלג על קודקודים שנדע שבוודאות שאין ערך בפיתוחם.

מהטכניקות הללו בונים החוקרים אלגוריתם חדש בשם Alpha-Beta Branch-and-Bound pruning (ABBnB). על פי האלגוריתם, לכל קודקוד יש את גבולות הαβ שלו, שהגיעו מגילוי קודקודים בשלב מוקדם יותר, וכן גבול שמגיע מפונקציית היוריסטיקה, ובאמצעותם ניתן לחשב את ערך השחקן בקודקוד מסוים כחיסור של הנקודות שבטוח יש לשחקנים אחרים (בחיתוך שתי הגבולות) מתוך כלל הנקודות האפשריות בסך הכל לכלל השחקנים, ולבצע יותר חיתוכים מאשר יכלו כל אחד מהאלגוריתמים בנפרד – אך גם במקרה זה לא מתאפשר גיזום עמוק.

בהשוואה שבוצעה בין האלגוריתמים על בסיס מספר הקודקודים שכל אלגוריתם מפתח כאשר הוא משחק Hearts או Sergant Major, התוצאות מראות כי האלגוריתם המשולב (ABBnB) אכן מפתח פחות קודקודים מכל אחת מהשיטות בנפרד (αβ וDFBnB) מאחר ומאפשר את הגזירות שהן מבצעות וגזירות נוספות המתאפשרות באמצעות המידע המשותף משתיהן.

על ידי שינוי התנאים לגיזום בטכניקה זו, או בכל אחת מטכניקות הבסיס, נוכל להגדיר מחדש תנאים לאלגוריתמים המבצעים גיזום תת-אופטימלי בעצי max^n. נסייג בכך שפתרון זה מתאים רק למשחקים בהם נדע את ערכי maxsum maxpו, ובשימוש ב Branch-and-Bound נידרש גם למשחקים בהם מתקיימת פונקציה מונוטונית.

פתרון למגבלה זו ניתן למצוא באלגוריתם נוסף בשם Paranoid המוזכר גם הוא במאמר לטובת השוואת תוצאות האלגוריתם ABBnB במבחן מספר הקודקודים שפותחו. האלגוריתם, בדומה לאלגוריתמים אחרים לפתרון משחקים מרובי משתתפים, מציע חיפוש בדמות עץ משחק לשני שחקנים, תחת ההנחה שכלל השחקנים שאינם שחקן השורש משחקים בקואליציה נגד שחקן השורש. באמצעות הנחה זו ניתן למשקל את עלי העץ בתועלת של שחקן השורש בלבד, שכן הוא פועל למקסם אותה ומנגד הם פועלים למזער אותה מבלי להתעניין בהישגייהם האישיים, ונקבל מבנה של משחק סכום אפס. נייצג את המצב לאחר כל פעולות הנגד שמבצעים השחקנים בתורם תחת קודקוד אחד ומכאן שנקבל עץ משחק הזהה לבעיית המינימקס. בהשוואה שביצעו החוקרים, אלגוריתם

הParanoid פיתח משמעותית פחות קודקודים מאלגוריתמי הmax^n שכן האלגוריתם פורס עצים קטנים משמעותית בעקבות הרדוקציה ולמעשה מאפשר תכנון ארוך טווח שכן הוא חוקר בעומק קבוע צעדים רבים קדימה במשחק ביחס לשחקן השורש. החוקרים מציינים כי ההנחה שהאלגוריתם מבצע מובילה למשחק תת-אופטימלי שכן היא אינה ריאלית ותוביל את שחקן השורש להעדיף מצבים בהם יפגע בצורה מינימאלית במידה ואכן קיימת קואליציה, על פני מצבים אופטימליים.

Schadd and Winands (2011) התייחסו גם הם לבעייתיות שבאלגוריתם הParanoid והציגו במאמר אלגוריתם בשם Best-Reply Search (BRS) שעושה רדוקציה לבעיה בשני שחקנים תחת הנחה אחרת. על פי האלגוריתם, בכל סיבוב רק שחקן אחד מנסה למזער את שחקן השורש, ורק הוא ישחק לאחר שחקן השורש. בפועל בכל תור שני שחקן השורש משחק ומתקבל עץ משחק זהה במבנהו לעץ משחק לשני שחקנים וניתן לחפש בצורה דומה אחרי ערך הMinmax. אחר כל תור של שחקן השורש יחקרו כל אפשרויות המשחק של כל היריבים של השורש והאלגוריתם יבחר מבין המהלכים של כל השחקנים את המהלך שימזער את הניקוד של שחקן השורש. גם כאן, בכל חישוב בעל עומק קבוע מפותחים יותר מהלכים של שחקן השורש ביחס לעצי max^n, ובכך מתקיים תכנון ארוך טווח. מנגד, במשחקים רבים דילוג על תורם של שחקנים מסויימים (למשל במשחקים עם מספר שחקנים קבוע או trick-based) עלול להוביל למצבי משחק לא חוקיים. כמו כן תורות של שחקנים שיתרמו לשחקן השורש לא ילקחו בחשבון.

החוקרים השוו את יכולות BRS אל מול Paranoid וmax^n. האלגוריתמים ששיחקו זה מול זה את המשחקים שחמט יפני, פוקוס ורוליט והחוקרים מדדו את אחוזי הניצחון של כל אלגוריתם. על פי התוצאות BRS ופרנואיד מציגים תוצאות טובות מmax^n ומגיעים לעומקים דומים. BRS מנצח את Paranoid בשחמט, אך בפוקוס ורוליט הם כמעט שווים. במצב בו שלושת האלגוריתמים שיחקו יחד max^n היה החלש ביותר וBRS היה החזק ביותר. יכולתיו של BRS מובהקות יותר כאשר ניתן לו זמן חישוב ארוך יותר. במסקנותינם החוקרים קבעו כי BRS הוא האלגוריתם המיטבי וכוחו בכך שהוא מחפש את היריב עם הצעד החזק ביותר ובכך מתקבל אלגוריתם זהיר הבוחן את כל היריבים. בכך שלא מאפשר לכל השחקנים לשחק, האלגוריתם חוקר יותר קודקודים של שחקן השורש בעומק קצוב ובכך מתאפשר תכנון ארוך טווח.

בעוד במחקר זה עוסקים החוקרים בפתרון אופטימלי, בפרוייקט נרצה להראות פתרון תת-אופטימלי חסום מבוסס פתרון אופטימלי. אופציה אחת תהיה באמצעות BRS – לאחר רדוקציה לעץ משחק לשני שחקנים נוכל לבחון זאת באמצעות האלגוריתם התת-אופטימלי החסום שהציג Atzmon (2018). במאמר מוצג אלגוריתם חיפוש חדש לעצי משחק בשם Bounded Alpha-Beta (BAB).

אלגוריתם זה מבוסס על אלגוריתם αβ ועל אלגוריתם נוסף בשם Minimax-* בו נעשה שימוש בשני ערכים נוספים L וU המשמשים לגיזום במשחקים עם אלמנט של מזל. בדומה לאלגוריתמים אלו, BAB מבצע חיפוש לעומק אך מגדיר סט חוקים חדש להגדרת ערכי המשתנים. בדומה לαβ – גיזום קודקוד n בעץ יתרחש כאשר יתקיים β (n) ≤ α(n) + ϵ. פלט האלגוריתם עבור קודקוד השורש n_1 הינו טווח $[L(n_1), U(n_1)]$ בגודל שאינו עולה על ϵ, וכל פתרון בטווח הינו פתרון תת-אופטימלי (כזה שאינו רחוק מערך המינימקס ביותר מ ϵ).

על מנת לבחון את ייעילות האלגוריתם החדש (BAB), החוקרים ביצעו ניסויים על עצים רנדומלים ועל משחק מוכר שנקרא שוטרים וגנבים, ולמעשה השוו את ביצועי האלגוריתם אל מול אלגוריתמי הαβ וMinimax-*. על מנת לתאר משחקים תלויי מזל נוסף אלמנט של הסתברות לביצוע המהלך הטוב ביותר בקודקוד מסויים אל מול ההסתברות לבצע כל מהלך אחר.

את תוצאות הניסויים בחרו החוקרים להציג באמצעות השינוי בערכם של שני מדדים –

- ER – מספר הקודקודים שפותחו על ידי אלגוריתם BAB חלקי מספר הקודקודים שפותחו על ידי האלגוריתם האופטימלי הרלוונטי. ככל שהערך של ER יהיה קטן יותר – BAB מבצע חיפוש ייעיל יותר.
- AMB – מתקבל מחישוב $|L(n_1) - U(n_1)|$ כאשר n_1 הוא שורש העץ. חסום על ידי ϵ מהגדרתו של BAB.

התוצאות במאמר חושבו על בסיס יותר מ50 חזרות בכל אחת מההגדרות עבורן הוצגו ערכי ER וAMB , ומציגות דפוסים דומים לגבי משחקים דטרמיניסטים ותלויי מזל. על פי התוצאות, ערכו של ER קטן ככל שערכו של ϵ גדל בהתאם לציפיות החוקרים כי יתכנו גיזומים רבים יותר בעץ כתלות במרחק הפתרון התת-אופטימלי המקובל מערך המינימקס. בנוסף לעצים עמוקים יותר ערכי ER נמוכים יותר מאחר ומתאפשרים חיתוכים עמוקים יותר בעלי יותר קודקודים. ערכו של AMB גדול יותר ככל שהערך של ϵ גדול יותר – תוצאה הגיונית מאחר ו ϵ מהווה חסם עבור AMB. כמו כן ערך AMB גדול יותר כתלות בעצים עמוקים יותר בעלי ערך ϵ זהה מאחר ומתאפשרים חיתוכים רבים יותר לעץ ועל כן הזדמנויות גדולות יותר להגדלת מרחב הערכים התת-אופטימליים.

במחקר נוסף שביצעו Schadd and Winands (2011), מבוצעת השוואה נוספת בין האסטרטגיות Paranoid, Best Replay Search (BRS), max^n בשילוב שיטת חיפוש בשם Monte-Carlo Tree Search (MCST) על העצים שנוצרים בשיטות השונות.

חיפוש MCST שונה מחיפוש מבוסס αβ מאחר ואינו מצריך שימוש בפונקציה היוריסטית לחישוב התועלת של קודקוד מסויים. הערכת קודקוד מסויים מתבצעת על ידי שימוש בפונקציה המבוססת על משתנים כמו מספר הביקורים בקודקוד האב, מספר הביקורים בקודקוד הבן, סטטיסטיקת הניצחון של הקודקוד הבן וסטטיסטיקת הניצחון של הפעולה המבוצעת. בנוסף האלגוריתם מסוגל לחשב את ערך הmixed equilibria של העץ בשונה מהאלגוריתמים האחרים שהוזכרו.

שיטת MCST מורכבת מארבעה שלבים - בחירה , התרחבות , סימולציה והחזרה מעלה. על ידי חזרה על ארבעת השלבים האלו בצורה איטרטיבית עץ החיפוש נבנה הדרגתית. על פי השיטה המקורית, MCST עושה שימוש במבנה עץ הזהה לזה של חיפוש של max^n, אך ניתן לבצע שימוש גם בעצים שפורשים Paranoid וBRS תוך שימוש בפונקציית הערכה שמטרתה להחליש את השחקן הנגדי (בשונה מmax^n שבכל תור נבחר במהלך בעל התועלת המקסימלית עבור השחקן שזהו תורו).

בהשוואות השונות שבוצעו במאמר החוקרים השוו את אחוזי הניצחון של האלגוריתמים תחת הגדרות זמן שונות, כאשר שיחקו זה מול זה דמקה סינית, פוקוס, רוליט ובלוקוס.

בסט הראשון של הניסויים שיחקו זה נגד זה האלגוריתמים max^n, Paranoid וBRS. החוקרים מדדו גם את עומק החיפוש הממוצע של כל אחד מהאלגוריתמים. על פי התוצאות max^n הוא האלגוריתם החלש ביותר עם אחוזי ניצחון נמוכים משמעותית המהאחרים, דבר שניתן להסביר בכך שמדובר באלגוריתם שמבצע מעט גיזומים ועל כן בזמן קצוב מגיע לעומקים נמוכים יחסית. בנוסף מצאו החוקרים כי תחת רוב המשחקים והקונפיגרציות BRS מציג תוצאות טובות יותר גם מParanoid.

בסט השני של הניסויים השוו החוקרים את ביצועי שלושה שחקני MCST, כאשר לכל שחקן עץ משחק שנבנה על ידי אחד מבין האלגוריתמים - max^n, Paranoid וBRS. על פי התוצאות MCTS-max^n משיג את התוצאות הטובות ביותר בכך שניצח את מספר המשחקים הגבוה ביותר תחת רובן המוחלט של הקונפיגורציות. החוקרים מסבירים תוצאות אלו בכך שיתרונם של אלגוריתמי הParanoid והBRS הוא הגיזומים הרבים שהם מאפשרים, אך אינם מבוצעים תחת MCST. החוקרים גם מצאו כי MCTS-Paranoid משיג תוצאות טובות יותר מMCTS-BRS - הסבר לך ניתן למצוא בכך שהנחת ה Best Reply של BRS יוצרת מצבי משחק לא אפשריים שפוגעים בביצועי האלגוריתם מבלי להשיג חיפוש עמוק יותר תחת MCTS.

הסט האחרון של הניסויים כלל השוואה בין המנצחים בהשוואות הקודמות - BRS ו MCTS-max^n (במשחק מרובה משתתפים) ותוצאותיה משתנות - תחת משחקים שונים וקונפיגורציות שונות היתרון עובר צד, אך החוקרים מצאו כי MCTS-max^n מתחזק ככל שהזמן המוגדר לתור ארוך יותר.

לסיכום, מבין שלושת טכניקות החיפוש שהוצגו במאמר – BRS מציג את התוצאות הטובות ביותר (מנצח יותר משחקים) וכוחו בכך שחוקר עמוק יותר בכל תור ובכך מפתח יותר צמתי max^n בכל חיפוש. תחת שיטת MCTS אלגוריתם max^n משיג את התוצאות הטובות ביותר - הסבר לכך ניתן למצוא בכך שגיזומי αβ הם כוחם של האלגוריתמים האחרים ואינם מבוצעים בMCTS. בהשוואה בין שתי השיטות העדיפות – לא נרשם יתרון מובהק לאף שיטה.

המאמר סוקר שיטות נוספות לחיפוש בעצי משחק ומציג יתרונות אלגוריתמים אחדים על פני אחרים. תוצאות מחקר זה יוכלו למקד אותנו בבואנו לבחור באלו אלגוריתמים אופטימליים כדאי להתמקד לפיתוח אלגוריתמים לחיפוש תת-אופטימלי חסום בעץ משחק מרובה שחקנים.

Zuckerman (2009) מציע דרך שונה מBRS להתמודד עם הקשיים שבאסטרטגיית הParanoid - אלגוריתם משולב המבצע חיפוש בעץ משחק מרובה משתתפים למשחקים בהם מנצח בודד ואין מדרוג בין המפסידים. האלגוריתם משלב מספר אסטרטגיות שונות ופועל בצורה כזו שמנצל את ההנחות עליהם כל אלגוריתם בנוי בשלב במשחק בו הנחות אלה אכן מתקיימות.

על פי האלגוריתם החדש - MP Mix, בכל תור שחקן המקסימום יבחר באחת משלושת האסטרטגיות- max^n, Paranoid או Directed Offensive – על פי האסטרטגיה המוצגת לראשונה במאמר זה, שכאשר שחקן מסוים חזק בהפרש ניכר משאר השחקנים ("שחקן המטרה"), יבחר שחקן השורש במהלך המזיק בצורה מקסימלית לשחקן המטרה על מנת למנוע את נצחונו במשחק. על פי האסטרטגיה, שאר השחקנים בתורם ינסו לחזק עצמם כפי שהיו פועלים באסטרטגיית max^n.

האלגוריתם המשולב מקבל ערכי To,Td -trashold, מחשב את היוריסטיקת הניקוד של כל משתתף ואת פער הנקודות בין שני השחקנים המובילים. אם השחקן המוביל הוא שחקן השורש וגם הפער שחושב גדול מTd השחקן יבחר באסטרטגיית הParanoid. אם השחקן המוביל אינו שחקן השורש וגם הפער שחושב גדול מTo השחקן יבחר באסטרטגיית Directed Offensive, אחרת - יפעל על פי אסטרטגיית max^n. כאשר To=0 וגם Td>0 השחקן יבחר באסטרטגיית האופנסיב בכל פעם שלא יוביל. כאשר Td=0 וגם To>0 השחקן יבחר באסטרטגיית הפרנואיד בכל פעם שיוביל. כאשר שניהם יהיו גדולים מערך המקסימום שהפונקציה היוריסטית מחזירה, השחקן ישחק תמיד על פי אסטרטגיית max^n.

המאמר בוחן את יכולות האלגוריתם המשולב אל מול max^n וParanoid תחת שני דומיינים – משחק לבבות ומשחק ריסק, תוך שימוש בהגדרות שונות לפרמטרים אותם האלגוריתם מקבל. תחת רוב הקונפיגרציות האלגוריתם המשולב משיג אחוזי ניצחון גבוהים יותר מהאלגוריתמים הקיימים.

תוצאות האלגוריתם מרשימות יותר עבור ריסק מאשר עבור לבבות ועל מנת להסביר זאת ניסחו החוקרים את Opponent impact factor – מדד לגודל השפעה שיש לשחקן אחד על שאר השחקנים שמתמודדים מולו. ערכו של מדד זה נקבע על ידי חישוב מספר המצבים במשחק שמוגדרים כ-InfluentialStates – כאלו שבהם פעולה של שחקן אחד משפיעה על ערך היוריסטיקה של שחקן אחר, חלקי כלל המצבים האפשריים. ניתן לראות כי בכל עומק של העץ, למשחק ריסק ערך Opponent impact גבוה משמעותית משל לבבות.

לסיכום, מסיקים החוקרים שאלגוריתם החדש מנצח יותר מצבים מהאלגוריתמים המוכרים שמולם נמדד, ומשיג תוצאות טובות יותר ככל שערך הOpponent impact factor של המשחק גבוה יותר. בבואנו לבחור אלגוריתם המבצע חיפוש אופטימלי במטרה לבצע על בסיסו חיפוש תת-אופטימלי חסום, חשוב שניקח בחשבון את תוצאות מחקר זה שכן האלגוריתם המשולב מציג תוצאות טובות אל מול האלגוריתמים המוכרים, שכן מנצל את ההנחות שמבוצעות במסגרתם בשלבים המתאימים במשחק.

בטרם נחליט באילו טכניקות נרצה להשתמש, נרצה לבחון חלופות לטכניקות מבוססות αβ. דוגמה לכך ניתן לראות בסקירה של Plaat (1996), המשווה בין אלגוריתמים מסוג Depth-First לאלגוריתמים מסוג Best-First.

אלגוריתמי הDepth-First ובראשם אלגוריתם αβ נמצאים בשימוש נרחב יותר מאשר פתרונות מבוססי Best-First, ובראשם כאלו שעושים שימוש בTransportation tables ו Iterative deepening. מחקרים קודמים הראו כי אלגוריתמי Best-First בעלי פוטנציאל לבצע חיפושים יותר יעילים, דוגמת SSS* המפתח באופן כמעט מוחלט עצים קטנים יותר, אך היותם סבוכים יותר והצורך שלהם בהקצאות זיכרון נרחבות הופכים אותם לשמישים פחות.

החוקרים מצאו כי ניתן לנסח מחדש את אלגוריתם SSS* במקרה פרטי של אלגוריתם Depth-First גנרי, ובמחקרים שביצעו זאת מצאו כי תחת ניסוח זה האלגוריתם צורך את אותה הכמות זיכרון כמו αβ, אך תחת הגדרות זיכרון זהות יעילותו אינה גבוהה יותר באופן ניכר וכבר היום קיימים אלגוריתמים מבוססי αβ דוגמת NegaScout המציגים ביצועים טובים יותר.

עוד מצאו החוקרים שתחת אותו אלגוריתם גנרי ניתן לבצע שיטות חיפוש מוכרות אחרות דוגמת C* ו*DUAL מאחר וגם הן מהוות מקרה פרטי שלו. המשותף לאלגוריתמים אלו הוא האופציה להמירם לאלגוריתמי Null window בהם החלון בו נעשה שימוש באלגוריתמי αβ הוא מינימאלי וגודלו שווה ל1, תחת הגדרה זו מתאפשר גיזום של יותר צמתים לאורך החיפוש. על מנת להתמודד עם צרכי הזיכרון האלגוריתם מבצע שימוש בtransposition table – טבלאות המרכזות מידע על מהלכים שנאסף בקריאות קודמות לאלגוריתם בחיפוש זה (בקודקודים קודמים) ובכך נחסכים חישובים חוזרים של מידע.

האלגוריתם הגנרי מנוסח בצורה כזו שמבצע קריאות αβ ומעמיק איטרטיבית עד שמגיע לערך המינימקס, ולמעשה ניתן לבצע באמצעותו חיפושים השקולים לחלוטין לחיפושים אחרים - יש להחליף רק את הערך שמתקבל באתחול. חיפוש שקול ל*SSS (AB-SSS*) נשיג על ידי אתחול ב∞+, חיפוש שקול ל*DUAL (AB-DUAL*) נשיג על ידי אתחול ב∞-. החוקרים מציעים במאמר קונפיגרציה חדשה בשם MTD(f) לאלגוריתם זה שלטענתם משיגה תוצאות טובות יותר משאר האלגוריתמים שהוזכרו קודם. MTD(f) הוא שימוש באותו האלגוריתם הגנרי על ידי אתחול כל איטרציה על פי תוצאת האיטרציה הקודמת ובכך מצפים החוקרים להקטנה משמעותית של מספר הקריאות עד להתכנסות לערך המינימקס.

החוקרים השוו בין ביצועי האלגוריתמים Alpha-Beta, NegaScout, AB-SSS*, AB-DUAL*, ו MTD(f) כאשר שיחקו שחמט, אוטלו ודמקה, על בסיס כמות העלים שפיתחו וסך הצמתים הכולל שפיתחו. על פי התוצאות MTD(f) מפתח משמעותית פחות עלים משאר האלגוריתמים ואחריו בסדר SSS*, DUAL*, NegaScout ולבסוף Alpha-Beta. בהשוואה על פי מספר הצמתים הכולל MTD(f) מפתח את מספר הצמתים הקטן ביותר, אחריו NegaScout, Alpha-Beta, DUAL* ולבסוף SSS*. כאשר החוקרים ביצעו השוואה בין MTD(f) לNegaScout על בסיס זמן עיבוד ומספר עלים הם מצאו כי MTD(f) מציג שיפור של כ10% בשני המדדים.

מסקנות החוקרים הן שאלגוריתם NegaScout מציג יכולות טובות מאלגוריתם αβ המשמש מקור להשוואות רבות, אך אלגוריתם MTD(f) מציג את התוצאות הטובות ביותר בכל המדדים שנבדקו. החוקרים מצאו בנוסף שייעילותו של אלגוריתם הSSS* ביחס לאלגוריתם Alpha-Beta אינה מובהקת, שMTD(f) משיג תוצאות טובות ממנו ועל כן אינם חושבים שיש להמשיך במחקר בעניינו.

לסיכום, המאמר מציג טכניקה גנרית לביצוע שיטות חיפוש מתקדמות - הן מסוג Depth-First והן מסוג Best-First, על ידי שילוב טכניקות נפוצות להתמודדות עם צרכי זיכרון – על ידי חיפוש בשיטת Iterative Deepening ושימוש מאגרי מידע כמו Transposition tables נוכל להשיג חיפושים יעילים יותר.

סוג נוסף של אלגוריתמים מסוג Best-First הוא אלגוריתמי Proof-Number Search (PN) אותם סקרו Herik and Winands (2008) במטרה לבצע השוואה בינהם לבין אלגוריתם αβ הנמצא בשימוש ברוב המערכות המבצעות חיפוש בעץ משחק, תוך ביקורת על יכולות אלגוריתם הαβ ב End-Game Positions - הבעיה עליה קבוצת אלגוריתמי הPN באים להתגבר.

PN- אלגוריתם הנועד לחיפוש ערכים בעצי משחק, ומטרתו היא הוכחת ערך השורש כtrue (לעץ ישנם 3 ערכים אפשריים - true המסמל ניצחון, false המסמל הפסד או תיקו, unknown אם לא ידוע). העץ מתואר כבעל קודקודי AND/OR, שלכל אחד מהם שני ערכים pn ו dpn (ערכים המייצגים את המספר המינימלי של קודקודים אותם צריך להוכיח בכדי להוכיח את הקודקוד הנוכחי או להפריכו בהתאמה). ערכיהם של pn, dpn נקבעים בהתאם לערכים של ילדיהם ובהתאם לסוג הקודקוד (AND/OR). האלגוריתם בוחר את הקודקוד הבא לפיתוח על פי מבנה העץ (כמות הילדים לכל קודקוד פנימי) והערכים בעלים, ומבצע חיפוש עבור "הקודקוד המוכיח ביותר", דבר הדורש ממנו זיכרון רב

וקיימים מקרים בהם האלגוריתם קורס מעקבות מגבלת זיכרון הנובעת מכך שכאלגוריתם Best-First הוא נדרש לשמור את כל העץ. על כן פותחו האלגוריתמים $PN^2, PN^*, PDS, df-pn$ שמטרתם להתגבר על בעיה זו.

PN^2- אלגוריתם שנועד להתמודד עם בעיית הזכרון של PN, אלגוריתם בעל שני שלבים– בשלב PN_1 מופעל PN על שורש העץ, כאשר אלגוריתם PN משתמש גם כפונקציית ההערכה של הבנים של השורש וזהו שלב PN_2 . בהערכת הבנים מוטלת מגבלת זיכרון כפונקציה של גודל העץ בPN_1. כאשר מושגת מגבלת הזיכרון בשלב PN_2 ישמר המידע שהושג על הבנים של שורש כל אחד מתתי העצים ושאר המידע של תתי העצים ימחקו. הילדים של קודקוד השורש מPN_2 לא מפותחים בהכרח אלא רק של הקודקוד הmost proving ותהליך זה נקרא Delayed evaluation.

PN^* - אלגוריתם זה פותח בסיס הרעיון של MTD(f) הממיר אלגוריתמי best לתצורת depth, מוכיח את שקלותם ומבטל את התלות בזיכרון רב. זהו האלגוריתם האיטרטיבי המבצע חיפוש לעומק הראשון שמבצע שימוש בmultiple-iterative deepening, וממנו פותחו אלגוריתמים נוספים המתגברים על חוסר יעילותו במקרים בהם יש לבצע disproof לעץ מאחר ומשתמש בערך threshold בודד שמגביל את ערך הproof number אך לא את ערך הdisproof number.

PDS – אלגוריתם זה מתגבר על הבעיה שהוצגה ב PN^*, על ידי שימוש בערכי disproof לצד ערכי proof (ובשני ערכי threshold בהתאמה) ומבצע שימוש בmultiple-iterative deepening בכל קודקוד על ידי שימוש בטבלאות המרה.

df-pn - האלגוריתם מהווה וריאציה של PDS שלא מבצעת iterative deepening בשורש העץ על ידי הגדרת שני ערכי הthreshold לאינסוף. וריאציה זו מוכחת כמחזירה תמיד את ה most proofing node בניגוד לPDS ולעיתים מהירה ממנה, אך סובלת מיותר מקרים של בעיית Graph-History Interaction - בעיה אותה מזכירים במאמר אך לא שמים עליה דגש.

המאמר מציג לראשונה אלגוריתם PDS-PN, הנועד לשלב את היתרונות של 2PN ו PDS. בשלב הראשון, מתבצע חיפוש PDS, כך שברוב הגדול של המקרים הוא מפתח את "הקודקוד המוכיח ביותר" לא רק עבור קודקוד השורש אלא עבור כל קודקוד פנימי. את הקודקודים שכבר פיתחנו נשמור בשתי טבלאות המרה לשימוש חוזר. נגדיר קודקוד כ proof-like אם סביר יותר שנצליח להוכיח אותו כproof מאשר כdisproof על ידי בדיקה מתמטית של מספר תנאים, ובמקרה זה נעלה את ערך הסף שלו בהתאם (עבור dislike-proof הגדלת ערך הסף המתאים לו). ניתן לראות כי קל יותר להוכיח עבור קודקוד מסוג or שהוא proof-like וכן להוכיח עבור קודקוד מסוג and שהוא disproof-like.

בשלב השני, מתבצע אלגוריתם PN שממשיך לחקור את העץ כל עוד מגבלת הזיכרון לא מפריעה, וכל עוד העץ עוד לא הוכח כproof או disproof. לאחר סיום השלב השני כל תת העצים שנוצרו בשלב זה ימחקו, ורק השורש של תת העץ בשלב זה ישמר בטבלת ההמרה. החוקרים ביצעו tuning לפרמטרים a וb שמקבל PDS-PN. מסקנת החוקרים היא שעבור כל ערך של a, מספר המצבים שהאלגוריתם פתר גדל ככל שb גדל – עד ערך מסוים, וציינו את ערכי הפרמטרים עבורם התקבל מספר המצבים הפתורים המקסימלי.

מסקנות החוקרים הן שאלגוריתמי הPN השונים השיגו תוצאות טובות מ$\alpha\beta$ במדדים שנבדקו, שאלגוריתם PN חלש בבעיות קשות בעקבות צרכי זיכרון גבוהים ושבאופן כללי PDS ו PN^2 פותרים מגוון בעיות גדול יותר מ $\alpha\beta$ וPN. כמו כן קבעו החוקרים כי PDS-PN מציג יכולות טובות יותר מ$\alpha\beta$ בשלב הendgame, כי הוא מהיר כמעט כמו PN^2 תחת ההגדרות המתאימות, פותר יותר מצבים קשים מ PN^2 ויעיל יותר מPN ואף מPN^2 מאחר ואיננו מגיע לסף מגבלת הזיכרון ומתפקד בצורה טובה גם במצבי זיכרון קשים. מסקנה נוספת היא ש Df-pn יכול להוות אלטרנטיבה איכותית לPDS-PN בעקבות תוצאותיו הטובות מול PDS.מכאן שאלגוריתמים ממשפחת PN הם כלי חשוב לפתרון בעיות בשלב הendgame ובפרט PDS-PN היעיל יותר מ PDSו PN^2 כendgame solver עבור משחקים קשים.

ביבליוגרפיה –

1. Atzmon, D., Stern, R., & Saffidine, A. (2018). Bounded suboptimal game tree search. Proceedings International Symposium on Combinatorial Search, Sweden, 11, 10-18.
2. Luckhardt, C.A., & Irani, K.B. (1986). An algorithmic solution of n-person games. Proceedings of the AAAI National Conference on Artificial Intelligence, 5, 158–162.
3. Schadd, M.P.D., & Winands, H.M. (2011). Best Reply Search for multiplayer games. IEEE Transactions on Computational Intelligence and AI in Games, 3(1), 57-66.
4. Plaat, A., Schaeffer, Schaeffer, J., Pijls, W., & de Bruin, A. (1996). Best-First and Depth-First Minimax Search in practice. Artificial Intelligence, 84(2), 299-337.
5. Nijssen, J.A.M., & Winands, M.H.M. (2011). An Overview of Search Techniques in Multi-Player Games. Computer Games Workshop at ECAI 2012, France, 50-61.
6. Knuth, D.E., & Moore, R.W. (1975). An analysis of alpha-beta pruning. Artificial Intelligence, 6(4), 293-326.
7. Zuckerman, I., Felner, A., & Kraus, S. (2009). Mixing Search Strategies for Multi-Player Games. Proceedings of the international jont conference on Artifical intelligence, USA, 21, 646-651.
8. Sturtevant, N.R., & Korf, R.E. (2000). On pruning techniques for multi-player games. Proceedings of the Conference on Innovative Applications of Artificial Intelligence, 12, USA, 201–207.
9. Van der Herik, H.J., & Winands, M.H.M. (2008). Proof-Number Search and its Variants. Oppositional Concepts in Computational Intelligence, 155, 91-118.
10. Wald, A. (1945). Statistical Decision Functions Which Minimize the Maximum Risk. Annals of Mathematics, 46(2), 265-280.

תכנון המחקר

מטרת המחקר:

הגדרת התנאים הדרושים על מנת שאלגוריתם חיפוש, בעץ משחק מרובה שחקנים, יחזיר פתרון תת-אופטימלי (עד כדי קבוע מהערך האופטימלי). פיתוח אלגוריתם חיפוש על בסיס תנאים אלו המחזיר פתרון תת-אופטימלי חסום בעצי משחק מרובי משתתפים.

סיכום שלבי מחקר:

1. שלב למידה – מטרת שלב זה היא ללמוד מתוך המאמרים אותם קראנו, את מרחב הבעיה איתה אנו מתמודדים, את מרחב הפתרונות הקיימים כיום לבעיה זו ואת תהליך המחקר אותו ביצעו חוקרים קודמים במחקרים בנושא. תוצרי שלב זה הם הסקירה הספרותית והידע שנרכש במהלך הסקירה, שיאפשרו לנו לפתח אלגוריתם תת-אופטימלי חסום עם ידע מקדים ומידע רלוונטי זמין.
2. שלב ניסוח תנאים – מטרת שלב זה הינה הגדרת התנאים לביצוע גיזומים תוך שמירה על מרחק חסום מאופטימליות הפתרון והוכחת נכונותם. בשלב זה ננסח גם תנאים על אופי המשחק, שחייבים להתקיים בכדי לשמור על נכונות האלגוריתם. התוצרים הצפויים משלב זה הם התנאים לביצוע גיזום ובצירוף הוכחה אנליטית המעידה על מציאת פתרון תת-אופטימלי חסום והתנאים המקדימים על המשחק (לדוגמה אופן צבירת/חלוקת הנקודות במשחק). בכדי לבצע שלב זה, נבחר אלגוריתם אופטימלי פשוט (MaxN), שסקרנו בסקירת הספרות, עליו נבצע גיזומים תת-אופטימלים בצורה נאיבית ונבחן תנאים שיאפשרו לשמור על פתרון תת-אופטימלי חסום. את התנאים אותם נרצה לבחון נוכל לבחור מתוך תנאים שתוארו באלגוריתמים אחרים מתוך הסקירה או הכללתם עבור מקרה גדול יותר, וכן נוכל לנסח תנאים על-ידי הגדרת משתנים וניסוח אילוצים למשוואות.
3. שלב פסאדו אלגוריתמים - מטרת שלב זה הינה פיתוח פסאדו אלגוריתמים.
ראשית, נפתח פסאדו אלגוריתם מבוסס MaxN המבצע גיזומים שונים על-פי התנאים שנוסחו בשלב הקודם. בנוסף, פיתוחי פסאדו אלגוריתמים נוספים, חלקם ירחיבו בצורה שונה את MaxN וחלקם ירחיבו פסאדו אלגוריתמים אחרים (כגון PARANOID , BRS) אשר מבצעים רדוקציה לעץ משחק בשני שחקנים עליהם יהיה ניתן להכליל את התנאים שניסחנו. בתום שלב זה, התוצרים הצפויים הם פסאדו אלגוריתמים המבוססים על אלגוריתמים שונים המאפשרים לבצע גיזומים תחת התנאים שהגדרנו המבטיחים פתרון תת-אופטימלי חסום. הפסאדו אלגוריתם יכלול את סדר סריקת הקודקודים העץ (אלגוריתם חיפוש), מתי ניתן לבצע כל גיזום (תנאי הגיזום שנוסחו) ובאילו תנאים נעדכן את ערך קודקוד האבא להיות ערך קודקוד הבן (אסטרטגיה).
4. שלב ניסוח השערות – מטרתנו בשלב זה היא ניסוח השערות חיזוי על התוצאות העתידיות לניסויים שנבצע על הפסאודו אלגוריתמים שנוסחו בשלב קודם לכן. התוצר הצפוי משלב זה הוא אוסף של השערות H0, שטוענת כי הפסאודו אלגוריתמים שפיתחנו מחזירים תוצאות טובות יותר מאשר אלגוריתמים קודמים אחרים, והשערות המשערות אילו מבין האלגוריתמים שפיתחנו יחזיר את התוצאות הטובות ביותר. לביצוע שלב זה נקבע את המטריקות בעזרתם נמדוד את טיב תוצאות הניסוי. נבחן מטריקות שונות שהוצעו במאמרים בסקירה הספרותית ובאמצעותם ננסח את השערותינו.
5. שלב המימוש – מטרת שלב זה היא מימוש האלגוריתמים שפותחו בשלב 3 ומימוש הדומיין - מספר משחקים אותם נבחר לממש, וזאת בכדי לבצע את הניסויים לבדוק את נכונות ההשערה מהסעיף הקודם. תחילה, נבחר ונממש משחקים העומדים בתנאים שנוסחו בשלב 2 שיאפשרו

לנו לבצע את הגיזומים. ההבדל בין המשחקים יבוא לידי ביטוי בעץ המשחק על-ידי מבנה העץ והשוני בפונקציות האבליואציה של העלים. לאחר מכן, נממש את הפסאדו אלגוריתמים שנוסחו בשלב 3, כך שכל פסאדו אלגוריתם יכול לייצג דרך פעולה מלאה של שחקן יחיד.

6. שלב ביצוע ניסויים, תוצאות ומסקנות – מטרת שלב זה היא הסקת מסקנות על ביצועי האלגוריתמים שמימשנו, על בסיס השוואה בין ההשערה שלנו לתוצאות מביצוע הניסויים. תוצרי שלב זה הם תוצאות הניסוי והמסקנות אותם ניתן יהיה להסיק, והם יהוו מדד להצלחה במחקר. הניסויים יתבצעו על-ידי הרצות של משחקים, כך שבין הניסוי לניסוי נשנה את - עץ המשחק, מספר השחקנים, המרחק מהפתרון האופטימלי. בכל ניסוי, כל שחקן ישחק על-פי אלגוריתם שהוגדר לו, כך יהיה ניתן לבצע קומבינציות שונות של אלגוריתמים להשוואה. את תוצאות הניסוי נבחן באמצעות המדדים שהגדרנו להצלחת האלגוריתם, אותם נוכל להציג כתלות במשתנים ששינינו בין הניסויים. לאחר התבוננות בתוצאות והשוואתם עם השערות שניסחנו קודם לכן, נוכל להסיק מסקנות לגבי ביצועי האלגוריתם ונוכל להמליץ או לא להמליץ על השימוש בו ביחס לאלגוריתמים אחרים.

תכנון הניסויים העתידיים:

שני הפרמטרים אותם נשנה באופן יזום בין ניסוי לניסוי הינם:

ε - אפסילון, המרחק המקסימלי בין ערך הפתרון האופטימלי לבין ערך הפתרון התת-אופטימלי עבור שחקן השורש.

N – מספר השחקנים במשחק, ערכו יהיה לפחות 3 בכל משחק.

לכל ניסוי נבצע מספר חזרות כך שנבצע קונפיגורציות שונות של אלגוריתמים לכל שחקן ושחקן. האלגוריתמים שבהם נשתמש יתחלקו לשניים:

אלגוריתמים המבצעים גזירה אופטימלית בלבד

אלגוריתמים המבצעים גזירה תת-אופטימלית שפותחו על-ידינו

נגדיר שני מדדים להצלחה של אלגוריתם:

ER (Expanded Ratio) – מוגדר כמספר הקודקודים שפותחו על-ידי אלגוריתם עם גיזומים תת-אופטימלי חלקי מספר הקודקודים שפותחו על-ידי אלגוריתם המבצע גיזומים אופטימליים בלבד. ככל שε יגדל כך ER יקטן, ER קטן פירושו גיזומים רבים וזמן חיפוש מהיר יותר.

AW (Actual Winner) – מוגדר כמספר המשחקים בהם ניצח האלגוריתם חלקי מספר המשחקים בה. בכדי שמדד זה יהיה רלוונטי נגביל תור של שחקן לזמן קבוע, ובכך אלגוריתם שיבצע יותר גיזומים יזכה להגיע רחוק יותר בעץ. ככל שε יגדל כך AW יקטן, AW קטן פירושו יותר הפסדים ופחות נצחונות.

בהרצת ניסוי שני הפרמטרים ε וN יקבלו ערכים כלשהם ,כל שחקן ישחק על-פי אלגוריתם שנבחר לו בתהליך הקונפיגורציה, נבצע מספר חזרות עם אותה הקונפיגורציה אך עם עץ משחק אחר, ונבחן את המדדים שהגדרנו להצלחה עבור כל אלגוריתם.

תיאור הנתונים:

אין צורך בשימוש בנתונים ממקורות חיצוניים לטובת המחקר. בכדי לבצע ניסויים נצטרך לבנות עצים רנדומליים או על-פי הדומיין (במשחק קלפים לדוגמה לחלק את חבילת הקלפים בין השחקנים). את הנתונים נייצר בעצמנו לאחר ביצוע סימולציות משחק.

אתגרים וסיכונים למחקר:

האתגרים העיקריים איתם אנו חושבים שנתמודד :

- ישנם מעט מחקרים בנושא, ורובם ישנים.
- הקושי בלהוכיח כי התנאים שניסחנו אכן מאפשרים לבצע גיזומים שמבטיחים את נכונות האלגוריתם כתת-אופטימלי חסום.
- לגלות כי הפתרון התת-אופטימלי אינו מייעל בצורה משמעותית, אינו משתלם מבחינת טרייד-אוף בין זמן ריצה ואיכות פתרון.

כלים ושיטות:

ישנם מספר אלגוריתמים אשר יהוו בסיס לפיתוח האלגוריתם שלנו: אלגוריתם חיפוש DFS, MaxN, Bounded Alpha-Beta.

שפת התכנות בה נבחר לממש הינה פייתון מכיוון שהיא מאפשרת לבצע סימולציות ומכילה ספריות סטטיסטיות.

לו"ז מעודכן לפרוייקט:

תאריך	משימה
11.2020	**שלב למידה – הבנת מרחב הבעיה ומחקרים קודמים בנושאים דומים.**
12.2020	**ניסוח תנאים – ניסוח תנאים שיבטיחו נכונות כאשר מתבצעים גיזומים תת-אופטימליים**
1.2021	**פיתוח פסאודו אלגוריתמים – פיתוח האלגוריתם הכולל גיזומים שונים ועדכון ערך בקודקוד**
2.2021	**ניסוח השערות – השערות לגבי ביצועי האלגוריתמים השונים**
3.2021	**מימוש האלגוריתמים והדומיין - נממש את הפסאודו אלגוריתמים שכתבנו ודומיין לצורך הניסויים**
4.2021	**ביצוע ניסויים – לבצע ניסויים עם פרמטרים שונים וקנופיגורציות שונות**
5.2021	**תוצאות ומסקנות – ניתוח תוצאות לגרפים על-פי המדדים להצלחה, והסקת מסקנות מתוך השוואות בין האלגוריתמים השונים ויחס להשערות.**

חלוקת אחריות בין חברי הצוות:

סקירת ספרות – עומר נגר

ניסוח תנאים – שחר מרץ, אסי זקס ופלג ביטון

הוכחה מתמטית – אסי זקס ופלג ביטון

פיתוח ומימוש האלגוריתמים והדומיין – אסי זקס ושחר מרץ

תכנון וביצוע הניסויים – עומר נגר ופלג ביטון

ניתוח תוצאות והסקת מסקנות – עומר נגר ושחר מרץ