Evolutionary AI for Five O' Poker

By: Idan Hertz, Omer Rimoch and Lior Guz

Five O' Poker

תוכן עניינים:

1. מבוא 3
2. תיאור הבעיה 3
3. Genetic Programing 5
4. Software overview 8
5. ניסויים, מהלך האבולוציה ותוצאות 9
6. מסקנות 13
7. ביבליוגרפיה 14

Five O' Poker

1. מבוא:

Five O' Poker הינו וריאציה של פוקר, אשר משלב אלמנטים ממשחק הפוקר הגנרי וסוליטר, המשחק הינו התמודדות ראש בראש בין שני שחקנים.

מהלך המשחק: המצב ההתחלתי של המשחק הוא שלכל שחקן מחולקים חמישה קלפים עם הפנים כלפי מעלה, כאשר כל קלף מייצג את הקלף הראשון בכל אחת מחמשת ידי הפוקר של השחקן. בכל תור, לוקח השחקן קלף מן החפיסה וממקם אותו באחת הידיים במטרה לבנות כמה שיותר ידיים מנצחות. יד מנצחת אם ורק אם היא חזקה יותר מהיד המקבילה לה אצל היריב. חוזק יד נקבע לפי הגדרות משחק הפוקר הגנרי.

בכל יד, לא יכול להיות יותר מקלף אחד מאשר בשאר הידיים, תנאי זה מבטיח שהשחקנים חייבים למלא את הידיים שלהם בהדרגתיות, כלומר רק כאשר בכל אחת מהידיים יהיו 2 קלפים, השחקן יוכל למקם קלף שלישי באחת מהידיים. הקלף האחרון בכל יד ממוקם הפוך, כך שהיריב לא יוכל לדעת בוודאות איזה יד יש ליריבו. כאשר בכל אחת מהידיים מוקמו חמישה קלפים, השחקנים הופכים את הקלף החמישי בידיים המקבילות, שנמצאות זו מול זו על לוח המשחק. השחקנים משווים בין הידיים המקבילות, שחקן עם יד מנצחת מקבל נקודה. השחקן עם מספר הנקודות הגבוה מנצח את המשחק.

2. תיאור הבעיה:

הבעיה: פיתוח שחקן ל- Five O' Poker שיצליח להרכיב מספר גדול של ידיים מנצחות מאשר יריבים שונים.

מרחב החיפוש: מספר הידיים האפשריות ששחקן יכול להרכיב מחושב באופן הבא: תחילה, חישוב מספר הקלפים האפשריים לקבל מהדילר בתחילת המשחק. אחר כך, נחשב את מספר הקלפים האפשריים שכל שחקן לוקח בתורו מהחפיסה מייצגים את החישוב עבור השחקן אשר משחק ראשון, מייצגים את החישוב עבור השחקן השני.

מאפייני המשחק:

2 Person	Zero-Sum	Perfect Information	No Chance	Well Defined
V	V	X	X	V

המטרה: נרצה לפתח שחקן ל- Five O' Poker באמצעות Genetic Programing אשר ינצח יריבים שונים המחזיקים באסטרטגיות משחק שונות. השחקן יחזיק Eval-Tree, באמצעותו השחקן ירכיב חמישה ידיים חזקות ככל הניתן. ה-Eval-Tree מתייחס ליד אחת ולא לכל חמשת הידיים.

מאפייני שחקן Five O' Poker: הינו שחקן אשר יודע לשלב בזמן משחק מספר מאפיינים שונים:

1. ניתוח חוזקה של כל אחת מחמשת הידיים שברשותו. ניתוח זה מתחשב בעיקרו בחישוב ההסתברות להוצאת קלף רלוונטי מהחפיסה ובחירה לאיזה יד למקם את הקלף הנבחר.
2. ניתוח חוזק הידיים של היריב, מאפיין זה כולל מעקב רצוף של בחירת היריב למקם קלפים בידיים שלו. כמו שהשחקן מנתח את ההסתברות להוצאת קלף מסוים מן החפיסה כדי לחזק ולשפר את אחת הידיים שלו, כך עליו לעשות כלפי היריב.

לפי מאפיינים אלו, השחקן יודע להשוות בין כל יד שלו לבין היריב ולהתמקד בידיים החזקות יחסית ליריב.

Five O' Poker

הבעיות הניצבות מול השחקן:

1. השחקן מחזיק מערך של Hands ורואה חלק מהמערך המקביל אצל היריב. כל יד מחזיקה נתונים על חוזקה – האם היד מכילה זוג, צבע, מספר קלפים להשלמת רצף וכדומה. יד היריב מחזיקה מידע זהה על חוזקה, אך בהינתן אך ורק המידע הגלוי על השולחן – 4 הקלפים הראשונים בידו.
2. השחקן אינו יודע איזה קלפים ישלפו מהחפיסה, הוא יכול אך ורק לחשב מהי ההסתברות לקבלת קלפים אשר יעזרו לו לחזק את היד. כאשר בתחילת הסיבוב הראשון (אם השחקן הוא שחקן מספר 1) יתכנו 42 אפשרויות לקלף שיישלף מהקופה, ובתחילת הסיבוב באחרון יתכנו 12 אפשרויות לקלף שיישלף מהקופה. לכן כדי להסתכל רק סיבוב אחד קדימה במשחק השחקן צריך לחשב בין סדר גודל של !42 מהלכים לבין !12 מהלכים, מה שהופך את הצפייה קדימה לקשה עד בלתי אפשרית.

התמודדות השחקן עם הבעיה: בכל תור השחקן מקבל קלף, כיוון שעץ החלטה עם יותר מרמה אחת ידרוש מהשחקן חישוב של עד מאות מיליוני מצבים, בחרנו להשתמש בPlanning – השחקן מנסה לשים את הקלף בכל אחת מהידיים החוקיות (אלו שעוד לא שם בהן קלף בסיבוב הנוכחי), מעריך את מצב הלוח בעזרת ה-Eval Tree שלו, ובוחר לשים ביד שקיבלה את ההערכה הגבוהה ביותר.

הנחות: בשונה מן הגדרות וחוקי המשחק, בחרנו להתעלם מן החלק במשחק אשר מתעסק בהימורים ובניהול הכסף של השחקן מן הסיבות הבאות:

- כשחקני Five O' Poker, אנחנו לעולם לא שילבנו במשחק שלנו הימורים וכסף, לכן העדפנו להפנות את מרכז תשומת הלב שלנו לחלק ההסתברותי של המשחק.
- Five O' Poker הוא משחק אשר דורש יכולות ניתוח של כל המידע אשר נמצא בכל אחד ממצבי המשחק וההסתברויות השונות לקלפים אשר נמצאים בחפיסת הקלפים (בשונה ממשחק הפוקר הגנרי, בו אין מידע על הקלפים אשר יש ליריב).
- מזל הינו חלק אינטגרלי מ-Five O' Poker. אנחנו רוצים לייצר שחקן שיתגבר על יריב שהמזל "שיחק לידו".

Five O' Poker

3. Genetic Programming:

a. מבנה הפרט:

1. הפרט מורכב מעץ הערכת מצב הלוח, Eval Tree, כך שהצמתים הם פונקציות והעלים טרמינלים. העץ מחזיק ערך מטיפוס float והעלים הינם מטיפוס float,Boolean ו-וקטור של cards.
2. Functions: מלבד הפונקציות הטריוויאליות ב-GP, הוספנו את הפונקציה ProbabilityToGetCard, הפונקציה מקבלת כפרמטר וקטור של קלפים, אותם השחקן צריך כדי לחזק את היד הנוכחית.ProbabilityToGetCard מחזירה לשחקן את ההסתברות לקבלת את אחד הקלפים מ-וקטור אשר נתון כפרמטר.

Function	Type
And	Boolean*Boolean -> Boolean
EqualTo	Float*Float -> Boolean
Greather Them	Float*Float->Boolean
If	Boolean*EvalNode*EvalNode
LessThen	Float*Float->Boolean
Multiply	Float*Float->Float
Or	Boolean*Boolean -> Boolean
Plus	Float*Float->Float
ProbabilityToGetCard	Vector<Cards> -> Float

3. טרמינלים:

a. מצב היד: טרמינלים אלו מתייחסים למצב היד שיש כרגע לשחקן. הטרמינלים מחזירים את ערך השדה הבוליאני של היד אותה השחקן בוחן. טרמינלים אלו הקנו לשחקן את היכולת לנתח את חוזק היד שלו. לדוגמא, הטרמינל hasFullHouse, יהיה True במידה ומהיד הנתונה קיימת אפשרות להרכיב בסוף המשחק FullHouse.

b. מצב היריב: טרמינלים אלו מתייחסים למצב היד של היריב. בעזרת טרמינלים אלו, השחקן מקבל מידע אודות היריב בעזרת ניתוח המידע הגלוי על מצב המשחק. כך ביכולתו של השחקן לנתח את חוזק היד שלו ביחס ליד המקבילה של היריב. לדוגמא הטרמינל opponentHasPair, מחזירה True אם במידע הגלוי בלוח המשחק ליריב יש אפשרות להרכיב בסוף המשחק יד שתכיל זוג קלפים.

c. חיזוק היד: טרמינלים אלו מתייחסים לקלפים שעדיין לא נפתחו. טרמינלים אלה מקנים לשחקן אפשרות לדעת אילו קלפים הוא צריך כדי להעלות את חוזק היד. הטרמינלים מחזירים וקטור של cards, שכעת נמצאים בחפיסת הקלפים. אם השחקן ימקם את אחד הקלפים הללו ביד הנתונה אז חוזק היד יגדל. לדוגמא הטרמינל MissingForFlush, מחזיר וקטור של קלפים שאם השחקן ימקם אחד מהם ביד הנוכחית אז ההסתברות שהיד תהיה "Flush" תגדל (כאמור Flush, הינה יד חזקה ביחס לשאר סוגי הידיים).

Terminal	Description

Five O' Poker

DiffOfHandStrength	מחזיר את הפרש החוזק בין היד של השחקן ליד של היריב.
HasFlush	האם קיימת האפשרות להרכיב מהיד הנוכחית יד שתהיה מורכבת מ5 קלפים מאותו הסוג.
HasFourOfAKind	האם קיימת האפשרות להרכיב מהיד הנוכחית יד שתכיל 4 קלפים עם אותו הערך.
HasFullHouse	האם קיימת אפשרות להרכיב מהיד הנוכחית יד שתכיל FullHouse.
HasPair	האם קיימת אפשרות להרכיב מהיד הנוכחית יד שתכיל זוג קלפים.
HasStright	האם קיימת אפשרות להרכיב מהיד הנוכחית 5 קלפים שהערכים שלהם יהיו רציפים.
HasThreeOfAKind	האם קיימת אפשרות להרכיב מהיד הנוכחית יד שתכיל 3 קלפים עם אותו ערך.
HasTwoPairs	האם קיימת אפשרות להרכיב מהיד הנוכחית יד שתכיל 3 קלפים שני זוגות קלפים עם אותו הערך.
IsLastRound	מחזיר True אם המשחק נמצא בשלב האחרון, בוא הקלפים אינם אמורים להיות גלויים. על השחקן לשאוף לחזק את היד כמה שיותר.
MissingForFlush	מחזיר וקטור של cards שבמידה והשחקן יוציא אחד מהם מהחפיסה וימקם אותו ביד הנוכחית אזי ההסתברות שהיד, בסופו של המשחק, תהיה "Flush" תגדל.
MissingForStraightFlush	מחזיר וקטור של cards שבמידה והשחקן יוציא אחד מהם מהחפיסה וימקם אותו ביד הנוכחית אזי ההסתברות שהיד, בסופו של המשחק, תהיה "StraightFlush" תגדל.
MissingForStraight	מחזיר וקטור של cards שבמידה והשחקן יוציא אחד מהם מהחפיסה וימקם אותו ביד הנוכחית אזי ההסתברות שהיד, בסופו של המשחק, תהיה "Straight" תגדל.
NumForFlush	מחזיר את מספר הקלפים החסרים כדי לקבל חמישה קלפים מאותו סוג
NumForStright	מחזיר את מספר הקלפים החסרים כדי לקבל חמישה קלפים שערכם בסדר עולה
OppenentHasFlush	האם קיימת האפשרות ליריב להרכיב מהיד הנוכחית יד שתהיה מורכבת מ5 קלפים מאותו הסוג.
OppenentHasFourOfAKind	האם קיימת האפשרות ליריב להרכיב מהיד הנוכחית יד שתכיל 4 קלפים עם אותו הערך.
OppenentHasFullHouse	האם קיימת אפשרות ליריב להרכיב מהיד הנוכחית יד שתכיל FullHouse.
OppenentHasPair	האם קיימת אפשרות ליריב להרכיב מהיד הנוכחית יד שתכיל זוג קלפים.
OppenentHasStraight	האם קיימת אפשרות ליריב להרכיב מהיד הנוכחית 5 קלפים שהערכים שלהם יהיו רציפים.
OppenentHasStraightFlush	האם קיימת אפשרות ליריב להרכיב מהיד הנוכחית יד שתכיל 3 קלפים עם אותו ערך.
OppenentHasThreeOfAKind	האם קיימת אפשרות ליריב להרכיב מהיד הנוכחית יד שתכיל 3 קלפים שני זוגות קלפים עם אותו הערך.
OppenentHasTwoPair	האם קיימת האפשרות ליריב להרכיב מהיד הנוכחית יד שתכיל 4 קלפים עם אותו הערך.
RandomInteger	מייצג מספר שלם רנדומלי בין 7 ל-7, מאפשר למשקל תכונות מסויימות.

b. <u>מבנה האוכלוסייה:</u> האוכלוסייה מורכבת מוקטור של n עצי החלטה אשר נבנו בשיטת ramp-half-and-half, כדי ליצור אוכלוסייה מגוונת ככל הניתן. בנוסף לגיוון שמנו דגש על בניית עצים עם גובה מוגבל.

c. <u>האבולוציה:</u>

1. <u>Fitness:</u> בכל תחילת דור, מתקיימת ליגה של $n+m$ שחקנים שנערכת כ-3 מחזורים כאשר בכל מחזור כל שחקן משחק נגד כל אחד מהשחקנים האחרים. הליגה מורכבת מ-n SmartPlayers אשר משחקים לפי ה-Eval-Trees של האוכלוסייה. בנוסף ל-SmartPlayers, בליגה יש עוד m יריבים בעלי אסטרטגיות מובנות (אשר יפורטו בחלק 4) כדי לחזק את האבולוציה. כאשר הליגה מסתיימת, ה-fitness של כל SmartPlayer מחושב כסכום הידיים אשר ה-SmartPlayer ניצח במהלך כל המשחקים. על כל ניצחון מקבל ה-SmartPlayer 10 נקודות בונוס. במצב של תיקו, ל-fitness של השחקן נקודה בונוס.
2. <u>Selection:</u> ביצענו selection tournament, $n-5$ פעמים בחירה אקראית של $selecion_size$ Eval-Trees והעברנו לדור הבא את ה-Eval-Tree בעל ה-fitness הגבוה מבין הEval-Tree הנבחרים. בנוסף, ביצענו elitism, והוספנו את חמשת ה- Eval-Trees בעלי ה- Fitness הגבוה ביותרת כדי להשלים את האוכלוסייה ל-n .
3. <u>Mutation:</u> על כל אחד מה-Eval-Trees באוכלוסייה, אנחנו מבצעים מוטציה בהסתברות של x_m. פונקציית המוטציה בוחרת על איזה סוג של Eval-Node לבצע את מוטציה. בהסתברות של 0.5 הפונקציה תבצע מוטציה על טרמינל, תבחר רנדומלית את אחד הטרמינלים ב-Eval-Tree ותחליף אותו בטרמינל בעל אותו Type. בהסתברות 0.5 הפונקציה תבצע מוטציה על פונקציה, גם במקרה זה היא תבחר רנדומלית את אחת הפונקציות ב-Eval-Tree. פונקציית המוטציה תמחק את תת העץ הגודל מהפונקציה, ותבנה במקומו עץ בגובה שתלוי בגובה תת העץ אשר עבר מוטציה.
4. <u>Crossover</u>: כל Eval-Tree עובר crossover בהסתברות x_{xover}. בעת ביצוע crossover, נבחר Eval-Tree נוסף מהאוכלוסייה לאחר selection. פונקציית ה-crossover בוחרת אקראית Eval-Nodes מתאים בשני העצים ומחליפה את תתי העצים אשר יוצאים מה-Eval-Nodes הנבחרים.

4. <u>Software Overview:</u>

Five O' Poker

a. מבנה המשחק:

1. Hand: מייצגת יד פוקר, היא המחזיקה מערך של Cards בגודל 5 ומידע על חוזק היד בכל שלב של המשחק.

מטודות עיקריות:

- InsertCard(card): מכניסה Card נתון ל-Hand, ומעדכנת את חוזקה.
- GetHandStrength(): מחזירה את חוזק היד במצב הנתון.

2. CardDeck: מייצגת את חפיסת הקלפים שעדיין לא נפתחו. היא מחזיקה מערך של Card בגודל 52.

מטודות עיקריות:

- DrawCard(): מוציאה מהחפיסה את הקלף הנמצא בראשה.
- Shuffle(): מערבבת את הקלפים.

3. Board: מייצגת את לוח המשחק, אשר בעזרתו מתנהל המשחק. המחלקה מחזיקה את המופע של כל שחקן ומופע של CardDeck ואחראית על הקשר בין השחקנים והקופה.

מטודות עיקריות:

- DrawCard(): מבקשת מהCardDeck קלף ומגישה אותו לשחקן שתורו.
- whoWon(): בסוף המשחק משווה בין הידיים של שני השחקנים ומכריזה על מנצח.

4. Game: מייצגת משחק.

מטודות עיקריות:

- Run(): מסמלצת ריצה של משחק, ומחזירה את השחקן המנצח.

5. Tournament: מייצגת טורניר, משמשת לחישוב Fitness של כל שחקן.

מטודות עיקריות:

- runTournament(): מפגישה של זוג שחקנים מספר פעמים במהלך טורניר, ומדרגת כל שחקן לפי הניצחונות שלו במשחקים.

b. פתרון המשחק:

1. Eval-Node: מייצגת פונקציה או טרמינל בעץ הערכת המצב, מופע שלה מחזיק, במקרה והיא פונקציה, וקטור של הקלטים הדרושים להפעלת הפונקציה.

מטודות עיקריות:

- getValue(Hand): מחשבת את ערך הטרמינל או את ערך הפונקציה ומחזירה אותו, בהתאם ליד בה שמנו את הקלף.

2. Eval-Tree: מייצגת את עץ הערכת המצב, מחזיקה שדה Root מסוג Eval-Node שערך ההחזרה שלו הוא float.

מטודות עיקריות:

- getValue(Hand): קוראת לgetValue של הRoot שלה ומחזירה ערך של המצב הנוכחי.
- getRandomNode(): מחזירה את Eval-Node אקראי בעץ לצורך Cross-over או Mutation.

3. CardPlacement: מייצגת את המידע על המיקום הקלף בלוח המשחק (מס' שחקן, מס' יד וכדומה).

4. SmartPlayer: מייצגת את השחקן המפותח באמצעות GP. המחלקה מחזיקה מופע של המחלקה Eval-Tree.

מטודות עיקריות:

- eval(CardPlacement): המטודה ממקמת קלף ביד נתונה, מחשבת את חוזק היד ומוציאה את הקלף מהיד.
- getNextCardPlace(Card): מחזירה מופע של CardPlacement. המטודה קוראת ל-eval ובעזרתה מחשבת את מיקום הקלף האופטימלי מבין הידיים החוקיות.

5. ניסויים, מהלך האבולוציה ותוצאות:

Five O' Poker

a. <u>אסטרטגיות היריבים:</u> בשלב חישוב הfitness של ה-Eval-Trees, הוספנו m שחקנים המשחקים לפי אסטרטגית משחק מוגדרת (Greedy & Pairing Players). מכיוון שאין מאמרים המתייחסים לאסטרטגיות משחק של Five O' Poker, יצרנו שחקנים שמשחקים לפי אסטרטגיות שאנחנו בנינו המסתמכות על ניסיון המשחק שלנו.

1. <u>Random Player:</u> השחקן ממקם את הקלף באחת הידיים החוקיות ברנדומליות.
2. <u>Greedy Player</u> – שחקן בעל אסטרטגיית משחק חמדנית. מבין כל הידיים בהם ניתן למקם את הקלף, ה-Greedy Player יבחר ביד אשר תתחזק הכי הרבה ביחס לשאר הידיים שניתן למקם בהם את הקלף.
3. <u>Pairing Player</u> – שחקן שמנסה "להצמיד זוגות", כלומר, בהינתן 5 ידיים וקלף, הוא ישים את הקלף ביד החוקית שבעת הנחת הקלף עליה, נוצר זוג. על פי אסטרטגיית משחק זו, ה-Pairing Player מרכיב שלישיות, זוגיים, רביעיות ו- Full-House, אך לא מרכיב Straight ו-Flush.
4. <u>Weighted-Pairing Player</u> – שחקן אשר פועל בדומה ל-Pairing-Player, אך בהינתן קלף ניתנת העדפה עבור ידיים בהם מיקום הקלף יגדיל את חוזקה היחסי של היד הנבחרת מול היד המקבילה של היריב. העדפה מתבצעת באמצעות משקול חיובי לידיים אשר מיקום הקלף בהם יגרור יד חזקה יותר מיריב.
5. <u>Full-Strategy Player</u> - דומה ל-Weighted-Pairing Player, כך שהוא משלב באסטרטגית המשחק שלו התבוננות במידע הגלוי על הלוח והרכבת יד חזקה יחסית ליד המקבילה של היריב. אך בניגוד אליו, אסטרטגיית המשחק שלו כוללת חישוב הסתברויות ליציאת קלף ספציפי או קבוצת קלפים מהחפיסה. בנוסף, על פי אסטרטגיה זו השחקן יכול "לוותר" על ידיים שאותם לא יוכל לנצח, חלק זה הוסף לאסטרטגיה בעקבות אסטרטגיות בסיסיות ל-Five O' Poker [3]. נשים לב שאסטרטגיה מקנה לשחקן יכולת לפתח כל יד בפוקר (כולל רצפים וצבעים).

ביצענו כ-100,000 משחקים בין שחקנים אשר משחקים לפי אסטרטגיות המשחק במטרה לבדוק את חוזק האסטרטגיות. לאחר ביצוע הבדיקה, ה-Full-Strategy Player ניצח את כל השחקנים. לעומת זאת ה- Weighted-Pairing Player הפסיד רק ל- Full-Strategy-Player כאשר הוא הפסיד ב-51% מההתמודדויות. ה-Random Player לא הצליח לנצח את אף אחד מהשחקנים יותר מ-16%.

Player Type	Random		Greedy		Pairing		Weighted Pairing	
Greedy	83.3%	16.7%						
Pairing	89.6%	10.4%	58.1%	41.9%				
Weighted Pairing	90.4%	9.6%	57.5%	42.5%	50.2%	49.8%		
Full Strategy	90.6%	9.4%	57.8%	42.2%	52.3%	47.7%	51.5%	48.5%

בכל תא בטבלה, מוצגים אחוזי הניצחונות (חלק השמאלי של התא) של סוג השחקן המתואר בעמודה הרלוונטית ואחוזי הניצחונות (חלק ימני של התא) של סוג השחקן המתואר בשורה, מתוך 100,000 משחקים. לדוגמא בהתמודדות של Greedy Player vs. Random, ה- Greedy Player ניצח ב-83.3% מהמשחקים וה-Random Player ניצח ב16.7 מהמשחקים.

<u>ניסויים אבולוציוניים:</u> בחרנו לבדוק את האבולוציה שלנו על ידי השוואת ביצועי השחקן האבולוציוני בהתמודדות נגד כל השחקנים המתוכננים המתוארים למעלה. ההתמודדות התבצעה בהרצת

Five O' Poker

100,000 משחקים בין השחקן לכל שחקן מתוכנן. כדי להיות הוגנים ככל האפשר, בחצי מהמשחקים השחקן האבולוציוני היה – Player1 וחצי מהמשחקים היה –Player2 .

<u>מהלך האבולוציה:</u> תחילה, ביצענו ניסויים אבולוציוניים עם פרמטרים אשר נלקחו ממצגות (Koza) GP, עם הפרמטרים הבאים: $selection_{size} = 10$, $m = 40$, $n = 210$, $x_{xover} = 0.9$, $x_m = 0.01$, גודל ה-*elitism* הינו 3. במוטציה, ההסתברות לבחור טרמינל הייתה 0.8 וגובה תת העץ החדש שנבנה היה חסום ב-*1.5* מגובה תת העץ שהוחלף. לאחר הרצת מספר ניסויים הבנו שהאבולוציה מפיקה מקסימום מקומי, יתרה מכך, בכל אבולוציה הגענו למקסימום מקומי אחר ואף תוצאה לא הניבה פתרון מספק.

Opponent	Win	Loss
Random Player	79.4 %	20.6 %
Greedy Player	42.2 %	57.8 %
Pairing Player	32.3 %	67.7 %
Weighted-Pairing Player	28.8 %	71.2 %
Full Strategy Player	25.7 %	74.3 %

תוצאות הריצה הכי טובה בניסויים הראשונים לאחר דור 300.

לכן, כדי לנסות לצאת ממקסימום מקומי במהלך האבולוציה בחרנו להגדיל את $x_m = 0.1$ ולהקטין את $x_{xover} = 0.6$. לאחר ביצוע הרצות עם שינוים אלו נתקלנו ב-bloat, העצים הגיעו לגבהים אסטרונומיים (לדוגמא 37) ללא שינוי משמעותי ב-fitness ובתוצאות האבולוציה. כדי למנוע bloat, ביצענו את השינויים הבאים: תחילה, הענשנו ב-fitness עצים מעל גובה 13, כל מדרגת גובה מאבדת 10% מה-fitness. בנוסף במוטציה, שיננו את ההסתברות לבחירת טרמינל ל-0.5, חסם הגובה של תת העץ שנבנה שונה למינימום בין 4 ל-גובה תת העץ שהוחלף+2 והגדלנו את ה- elitism ל-10 כדי להגביר את הסיכוי שפרטים חזקים ישרדו.

לאחר הרצת האבולוציה עם השינויים המתוארים למעלה, קיבלנו שחקנים יותר טובים. אך, עדיין לא הגענו לתוצאה מספקת. בצפייה בתוצאות האבולוציה ראינו כי רוב הפרטים זהים פרט לשינויים מינוריים בעצים. כלומר השינוי במוטציה לא הוציא אותנו מתופעת המקסימום המקומי אשר נתקלנו קודם, אך תופעת ה-Bloat נפתרה.

Opponent	Win	Loss
Random Player	82.7 %	17.3 %
Greedy Player	52.4 %	47.6 %
Pairing Player	41.4 %	59.6 %
Weighted-Pairing Player	40.6 %	59.4 %
Full Strategy Player	39.5%	60.5 %

תוצאות הריצה הכי טובה בניסויים הראשונים לאחר דור 300.

כעת, הסקנו שה-selection pressure גבוהה מדי, כדי לפצות על כך, הקטנו את ה-elitism ל-5 ואת ה-selection size לארבע. בנוסף, כדי לאפשר מגוון גבוה יותר, שיננו את $x_{xover} = 0.5$, $x_m = 0.25$.
בניסויים הבאים השגנו תוצאות מעניינות, אבל ראינו כי החל מדור מסוים אין שינוי גבוה בביצועי השחקנים.

אחוז הניצחונות הממוצע של ה-elitism מול ה-Full-Strategy player לפי דור

לכן בריצה הסופית בחרנו להריץ אוכלוסייה בגודל 210, עד דור 130, אמנם להערכתנו התוצאה היא אופטימום מקומי, אך אופטימום מקומי זה סיפק את תוצאות הניסוי בכך שניצח את ה- Full-Strategy Player ביותר מ-50% מהמשחקים ביניהם. להלן תוצאות הניסוי הטוב ביותר.

דור 1:

Opponent	Win	Loss
Random Player	76.8 %	23.2 %
Greedy Player	32.8 %	67.2 %
Pairing Player	32.1 %	67.9 %
Weighted-Pairing Player	31.3 %	68.7 %
Full Strategy Player	29.3 %	70.7 %

תוצאות השחקן הטוב ביותר בדור 1 נגד 5 השחקנים המתוכננים

דור 65:

Opponent	Win	Loss
Random Player	86.2 %	13.8 %
Greedy Player	51.3 %	48.7 %
Pairing Player	44.8 %	55.2 %
Weighted-Pairing Player	44.3 %	55.7 %
Full Strategy Player	43.1 %	56.9 %

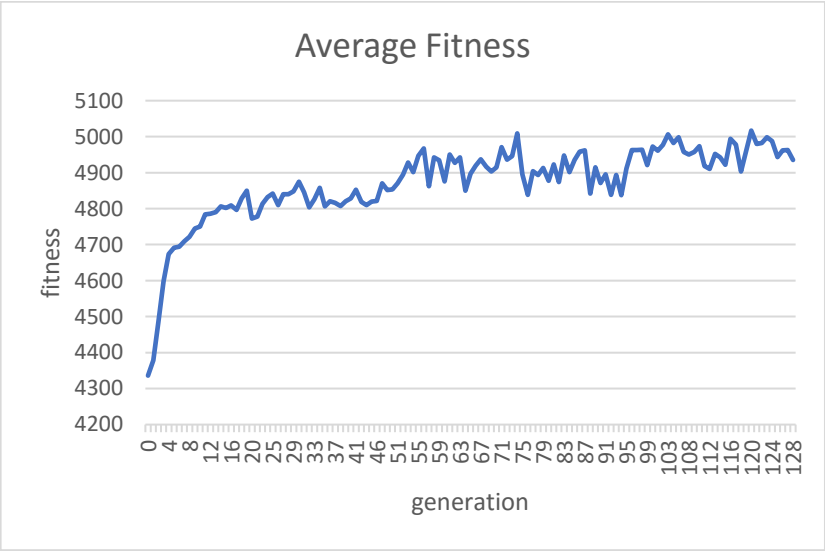
תוצאות השחקן הטוב ביותר בדור 65 נגד 5 השחקנים המתוכננים

Five O' Poker

דור 130:

Opponent	Win	Loss
Random Player	91.2 %	8.8 %
Greedy Player	61.2 %	38.8 %
Pairing Player	54.1 %	45.9 %
Weighted-Pairing Player	53.4 %	46.6 %
Full Strategy Player	52.6 %	47.4 %

תוצאות השחקן הטוב ביותר בדור 130 נגד 5 השחקנים המתוכננים



Fitness ממוצע לדור, באבולוציה אשר הפיקה את השחקן הטוב ביותר

6. מסקנות:

a. השחקן האבולוציוני, מנצח את יריבים בעלי אסטרטגיות משחק שונות ביותר מ-50% מהמשחקים. השחקן מרכיב ידיים חזקות ככל הניתן באמצעות ה- Eval-Tree שלו, ומשמיט ידיים שלפי הערכתו אין לו סיכוי לנצח בהן.
ניתן לראות בתמונה-1 את הרכבת לוח המשחק בגמר התמודדות מול Pairing-Player. נסתכל תחילה על יד מס' 3 השחקן מחזיק יד שממש קרובה ל-Straight Flush שהינה היד הכי חזקה מפוקר , ברגע שלא הצליח להשלים את ה-StraightFlush הוא השלים את היד ל-Flush, כלומר השחקן ניצל את היד בצורה הכי טובה שיכל. בנוסף ניתן לשים לב כי יש שוני בין סגנון הידיים שהשחקן בונה, לדוגמא יד 5 הינה זוגיים ויד 4 היא שלישיית מלכים. בנוסף ניתן לראות בסיבוב הראשון, שהוא לא קיבל קלפים כדי שיהיו לו זוגות, הוא בחר לנסות ליצור Flush, לכן חיזק ככל שיכל את הידיים שלו.

	1	2	3	4	5
Evolutionary player	J Heart	5 Club	5 Diamond	K Club	3 Spade
	8 Heart	2 Club	9 Diamond	6 Club	7 Heart
	10 Heart	Q Heart	8 Diamond	K Spade	7 Club
	J Club	5 Heart	7 Diamond	K Diamond	J Spade
	A Club	Q Spade	2 Diamond	9 Club	3 Diamond
Pairing Player	4 Heart	3 Heart	2 Spade	Q Club	2 Heart
	8 Club	4 Spade	6 Diamond	4 Club	5 Spade
	10 Spade	10 Diamond	A Spade	K Heart	9 Heart
	10 Club	8 Spade	A Heart	3 Club	9 Spade
	Q Diamond	6 Spade	J Diamond	7 Spade	6 Heart

תמונה -1: Evolutionary Player vs Pairing Player. מוצגות הידיים של כל שחקן ממוספרות משמאל לימין הדגשנו את הידיים המנצחות בצהוב. ניתן לראות כי השחקן האבולוציוני ניצח בכל הידיים

בתמונה-2 מוצג לוח המשחק בגמר ההתמודדות מול ה-Full-Strategy Player. כעת ניתן לשים לב שיש לשחקן האבולוציוני העדפה ל-Flush כל עוד הוא ניתן, כאשר לא ניתן, הוא בונה זוגות וזוגיים (יד 5). לעומת זאת השחקן אסטרטגיה מנסה לבנות זוגות יותר חזקים משל השחקנים האבולוציוני. (יד 4 לדוגמא).

	1	2	3	4	5
Evolutionary player	A Heart	7 Diamond	A Diamond	6 Diamond	7 Heart
	5 Heart	K Club	10 Diamond	J Spade	4 Heart
	J Heart	K Spade	9 Diamond	J Club	4 Diamond
	3 Heart	10 Heart	3 Diamond	K Diamond	3 Club
	6 Heart	Q Spade	8 Diamond	6 Club	2 Heart
Full-Strategy Player	9 Heart	Q Heart	8 Heart	5 Club	5 Diamond
	4 Club	3 Spade	J Diamond	A Club	2 Spade
	Q Diamond	Q Club	8 Club	10 Spade	10 Club
	9 Spade	2 Club	9 Club	5 Spade	2 Diamond
	7 Spade	4 Spade	8 Spade	A Spade	6 Spade

תמונה -2: Evolutionary Player vs Full-Strategy Player. מוצגות הידיים של כל שחקן ממוספרות משמאל לימין הדגשנו את הידיים המנצחות בצהוב. ניתן לראות כי השחקן האבולוציוני ניצח בארבע ידיים.

b. לצורך השוואה, ערכנו משחקים בין השחקן האבולוציוני לבין מתמודדים אנושיים. ביצענו מספר התמודדויות שאי אפשר להסיק מהן נתונים סטטיסטים, אבל בחלק גדול מהמשחקים השחקן האבולוציוני ניצח. ההתמודדות מול השחקן הייתה מאתגרת ואפשר להבין מבחירת הפעולות שלו שאכן נוצר שחקן שמעריך את מצב הלוח בצורה הגיונית וטובה.

7. ביבליוגרפיה:

1. https://en.wikipedia.org/wiki/Five-O_Poker
2. http://www.gamezebo.com/2012/01/23/five-o-poker-review/
3. https://pokerglam.com/2013/11/11/strategies-in-five-o-poker/
4. http://www.genetic-programming.com/gecco2003tutorial.pdf
5. M. Sipper, *Evolved to Win*, Lulu, 2011.
6. Evolutionary computation course, presentations:
    a. GP(Koza)
    b. Game Playing : Adversarial Search