קבלת החלטות בזמן אמת עם Monte Carlo Tree Search במשחק Ms. Pac-Man

יאיר לוי, אלה שלום

המשחק Ms. Pac-Man הוא משחק Arcade המבוסס על המשחק Pac-Man שיצא בשנת 1980. במשחק השחקן נדרש לבצע קבלת החלטות בזמן-אמת, כאשר מטרתו העיקרית היא להשיג כמה שיותר נקודות. המשחק כולל מספר מבוכים, בהם פאק-מן רשאי לנוע בחופשיות, ובו זמנית ארבע רוחות רודפות אחריו במטרה למנוע ממנו מלהשיג ניקוד. במסמך זה נשווה בין אלגוריתם MCTS שנלמד בקורס, לבין שיטות אופטימיזציה שונות, במטרה לסייע לפאקמן שלנו להשיג תוצאות טובות ככל הניתן.



**תיאור** **הבעיה**

המטרה העיקרית של משחק הפאקמן היא להשיג כמה שיותר נקודות. נקודות ניתן להשיג באמצעות אכילת pills המפוזרים ברחבי המבוך, או באמצעות אכילת רוחות מפוחדות לאחר אכילת power pill. במשחק יש 11 מבוכים, ולפאקמן 3 נשמות. כאשר פאקמן חוצה כמות נקודות מסוימת, הוא מרוויח נשמה נוספת. כמצוין לעיל, לצד פאק-מן ישנן ארבע רוחות עוינות שמטרתן היא לחסל אותו, ובכך למנוע ממנו נקודות נוספות. בסביבה שלנו, נרצה למקסם את התועלת של פאק-מן ממעבר במסדרוני המבוך, תוך מזעור של הנזק הנגרם מצד הרוחות.

אחד האתגרים הגדולים ביותר – אם לא הגדול ביותר – שנתמודד איתו במשחק הוא יצירת האיזון בין תכנון ארוך-טווח לבין תכנון קצר-טווח. במצבים רבים במהלך המשחק, הדרך האופטימלית של פאק-מן תסתור את הפרס המידי שיכול לקבל בטווח הקצר, ולכן נצטרך למצוא דרך פתרון שכוללת בתוכה מעט גמישות ו-"חיזוי" של העתיד, תוך איסוף כמה שיותר נקודות בדרך.

**גישות הפתרון**

האלגוריתם הראשי אותו בחרנו לחקור הוא לא אחר מאשר האלגוריתם המפורסם Monte Carlo Tree Search (MCTS). אלגוריתם MCTS נחל הצלחה רבה בקרב משחקי Real-Time, וגם בקטגוריות משחקים נוספות כמו פאזלים, שחמט, GO כמו שלמדנו בכיתה. ולכן ציפינו שהוא יהיה מוצלח גם עבור משחק פאק-מן.

בנוסף למוניטין של MCTS בתור אלגוריתם שמניב תוצאות טובות במשחקים הדורשים קבלת החלטות בזמן-אמת, בחרנו בו מאחר והוא מצליח להתמודד עם האתגר שאותו ציינו, והוא האיזון בין תכנון ארוך-טווח לקצר-טווח. הודות לשלב הסימולציה של האלגוריתם, נוכל לקבל הערכות שמשקפות את איכות המצבים השונים בטווח הארוך.

אלגוריתם נוסף אותו נרצה לנסות הוא Minimax. בדומה לאלגוריתם MCTS, האלגוריתם הזה גם מבצע קבלת החלטות באמצעות עץ עם מהלכים אפשריים, אך מאחר והוא בוחן את כל האפשרויות, הגובה של העץ שלו צריך להיות קטן יותר משמעותית מזה של MCTS, מאחר ובמשחק הפאקמן יש הגבלת זמן על קבלת החלטה לבצע מהלך. אנו מצפים שגובה עץ קטן יותר יגרום לכך ש-Minimax ידגיש יותר את הטווח הקרוב מאשר את הטווח הרחוק, אם כי באופן מדויק יותר מ-MCTS.

כדי לנסות לעקוף את בעית זמן החישוב, ננסה להשוות גם את האלגוריתם Alpha-Beta Pruning, שמטרתו לספק תוצאות זהות לאלו של Minimax, עם פחות זמן חישוב.

לבסוף, שיטה אחרונה בה נשתמש תהיה היוריסטיקה. בפשטות, בכל נקודה בה פאק-מן צריך להחליט איזה מהלך לבצע, נסמלץ כל מהלך מבין האפשריים כרגע, ניתן למצבי המשחק החדשים ציון, וניקח את המהלך בעל הציון הגבוה ביותר. שיטה זו מהירה, שמה את כל משקלה על התכנון לטווח הקצר, ובכלל לא מתייחסת לטווח הארוך.

**מה אנו מנסים להשיג?**

בפאקמן צריך לנסות לקבל את מספר הנקודות המיריבי. נרצה לבצע אופטימיזציה על המהלכים אותם פאק-מן מבצע, כדי לאפשר לו לעשות זאת. נרצה להשוות בין השיטות שתיארנו מבחינת התוצאה (הניקוד) שהן מביאות, הזמן שהן לוקחות והאופן בו הן פותרות את הבעיות. השימוש במשחק הפאקמן נותן לנו את האפשרות לראות את בצורה ויזואלית את המשמעות של כל שיטה שאנו מממשים. נוכל לראות איזה צעד פאקמן בוחר לעשות בכל רגע במשחק, ומכך נוכל לנתח את האסטרטגיות שהוא מפתח כתוצאה משימוש בכל שיטה. נוכל להבין בצורה טובה יותר מתי הרוחות מצליחות להערים עליו, מתי הוא מכניס את עצמו לאזורים מסוכנים במבוך ומתי הוא מצליח להתגבר ולצאת ממצבים מסוכנים.

**Heuristic function** **:**

**תיאור:** המטרה העיקרית של הפונקציה הזו היא להשוות את השיטות המתקדמות יותר לפונקציה בסיסית, שמסוגלת להסתכל רק על הפעולות מהמצב הנוכחי ללא הסתכלות מתוחכמת על המשחק. הגענו למימוש הספציפי הזה בצורה ידנית: הרצנו כל פעם את המשחק, התבוננו בפאקמן וניסינו להבין מה ההחלטות השגויות שהוא מקבל ואיך אנחנו יכולים לשנות את המישקול של הגורמים השונים בפונקציה ההיוריסטית בשביל לגרום לפאקמן למצוא אסטרטגיה נכונה. החלטנו לשערך כמה גורמים בפונקציה ההיורסיטת, לחשב אותה עבור כל מצב ולהחליט לבצע את הפעולה שמובילה למצב שנותן את הציון המקסימלי. בשקלול של הגורמים היינו צריכים למצוא איזונים בין התנהגויות שונות של פאקמן שנבעו ממשקולים שונים. איזון בין בריחה מרוחות, לבין לקיחת סיכון בשביל לקבל ניקוד. איזון בין פעולות מסוכנות שיגרמו למשחק להסתיים מהר ופעולות שעלולות לבזבז את זמן המשחק בלי להביא תוצאות.

הגורמים שהגענו אליהם במשקול של הפונקציה ההיוריסטית הם:

1. האוכל: נרצה לתגמל את פאקמן אם המהלך שעשה גרם לו לאכול אוכל (כלומר אוכל שאחרי המהלך לא נמצא במשחק, והיה ממוקם במיקום שפאקמן נמצא בו כרגע) נרצה להוסיף לפאקמן 100 נקודות . אבל נוריד לו נקודות על כך שהוא רחוק מאוכל. נרצה להתבונן בכל מצב רק על האוכל שהכי קרוב לפאקמן באותו הרגע. נחשב את המרחק שלו מהאוכל הכי קורב אליו, נחלק בשתיים ונוריד מכמות הנקודות של פאקמן. בצורה כזו אנחנו "מענישים" את פאקמן ככל שהוא יותר רחוק מהאוכל הכי קרוב אליו.
2. המרחק לרוחות פעילות: אחרי נסיונות רבים, גילינו שעלינו "להעניש" את פאקמן רק אם הוא נמצא בקרבת רוחות. אם הרוחות קרובות אליו מאוד, כלומר במרחק של לפחות 3 ממנו, ניתן לו עונש גדול מאוד (2000 ביחס הפוך למרחק של הרוח הקרובה ביותר ממנו). הסיבה היא שנרצה שאם רוח התקרבה למרחק כזה, פאקמן יעשה הכל כדי לברוח ממנה .
3. רוחות מפוחדות: אם קיימות רוחות מפוחדות במשחק נרצה לתגמל את פאקמן ב 50 נקודות אם הוא הגיע אליהן.
4. קפסולות: אם קיימות קפסולות במשחק ולא קיימות רוחות מפוחדות, אז משתלם לפאקמן לאכול את הקאפסולה. לכן על הגעה לקאפסולה נרצה לתגמל אותו ב 100 נקודות.

**תוצאות:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Ghosts \ avg. stats** | Avg. Score | Avg. Time | Avg. Lives | Avg. Maze |
| Random | 29045 | 13018 | 0.1 | 9.25 |
| Starter | 6712 | 3105 | 0 | 1.77 |
| Aggressive | 4542 | 1410 | 0 | 0.35 |
| Legacy2TheReckoning | 2432 | 1145 | 0 | 0.38 |

**מסקנות:**

לאחר עשרות משחקים, נשים לב שהניקוד יורד משמעותית ככל שהרוחות הופכות ליותר אגרסיביות, אך הקפיצה המשמעותית קורית בין Random ל-Starter. זאת מאחר והפונקציה מודדת את הפעולה הטובה ביותר מבין הפעולות המידיות, ולא לוקחת בחשבון את הטווח הרחוק:

כשהרוחות רנדומליות, הן לא מנסות לתפוס את פאקמן כלל, ומספיק להסתכל על הפעולות הקרובות, ובמקרה הגרוע להסתובב אם נקלענו ליד רוח. שיטה זו מניבה ביצועים טובים מאוד עם הרוחות שלנו הודות לטבע החמדן-אך-זהיר שלה, שבליבו איסוף נקודות ולהתרחק מיד כשאפשר. עם רוחות שכאלה, הסבירות שתיווצרנה מלכודת בה הרוחות מסונכרנות אחת עם השנייה היא נמוכה עד כדי אפסית, לכן כמעט תמיד רק רוח אחת או שתיים ירדפו את פאקמן בו-זמנית, ועדיף לו להיות חמדן.

ברגע שהרוחות מתחילות לנסות אקטיבית לתפוס את פאקמן, הוא לא שם לב בכלל לכך שהן מנסות לתפוס אותו, ובסיכוי גבוה ייכנס למבוי סתום או מלכודת. הרוחות יכולות לרדוף אחריו מכיוונים שונים, אך מאחר ואינו מתחשב ביותר ממהלך אחד קדימה, אינו רואה זאת, ונתפס.

נרצה למצוא אלטרנטיבה, כזאת שתעזור לפאקמן לשרוד יותר זמן ובכך לתת לו יותר זמן לאסוף ניקוד.

**Minimax** **:**

**תיאור:**

בעקבות היכולת של הרוחות לגבור על פאקמן, החלטנו לממש שיטה שתיתן לו את היכול לצפות את המהלכים שלהם. בכל שלב החלטה שלו בעץ, פאקמן בוחר את הפעילות שתביא לו לרווח מקסימלי. ואילו בכל שלב החלטה של הרוחות פאקמן בוחר את הרווח המקסימלי עבור הרוחות (כלומר המינימלי עבורו). הפונקציה שאיתה הערכנו את ציון העלים היא פונקצייה להערכת מצב, לעומת הפונקצייה להערכת מהלכים מהשיטה ההיוריסטית.

ראינו שאין אפשרות לתת לפאקמן לרוץ על כל השכבות בעץ ( שכבה מוגדרת אצלנו כמהלך של פאקמן ושל כל הרוחות), מכיוון שזמן החישוב נעשה גדול מידי, עד כדי לא ישים. לכן החלטנו שפאקמן יקבל כהיפר פרמטר את העומק עבורו הוא יעשה את החיפוש. מאחר ויש 5 סוכנים על הלוח בכל עת (פאקמן ו-4 רוחות), ולכל סוכן יש 5 פעולות אפשריות (כל כיוון או לעצור במקום) נקבל שישנן מסלולים שונים לחישוב, כאשר הוא מספר השכבות.

כמו שניתן לראות, הכמות עצומה. לכן, החלטנו לבדוק מה קורה בגובה 2 בלבד בתור התחלה (יש לציין שאפילו כמה עשרות בודדות של משחקים לקח להריץ בגובה זה מעל לשעה).

**תוצאות (עומק העץ – 2):**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Ghosts \ avg. stats** | Score | Time | Lives | Maze |
| Random | 3228 | 5630 | 0 | 0.94 |
| Starter | 1554 | 2981 | 0 | 0.54 |
| Aggressive | 1850 | 743 | 0 | 0.01 |
| Legacy2TheReckoning | 1581 | 5568 | 0 | 0 |

**מסקנות:**

נשים לב לתופעה מעניינת – פאקמן הופך להיות "פרנואיד". הניקוד שפאקמן השיג במהלך המשחקים נמוך משמעותית מהניקוד שהשגנו בשימוש עם היוריסטיקה, והיחס שבין הזמן שבו שרד לבין כמות הניקוד שהשיג גדל משמעותית (כלומר, היינו מצפים שאם פאקמן שרד כמות זמן שכזו, אז גם הניקוד היה אמור להיות גבוה יותר).

תופעה זו נובעת כתוצאה ישירה מהאפשרות להסתכל "קדימה". פאקמן הופך להיות זהיר יותר, אפילו ממצבים יחסית רחוקים, רק מאחר שהוא "זיהה" שהם בדרך. בעקבות כך, פאקמן עסוק בלהתחמק מהרוחות, יותר מאשר באיסוף נקודות, ואז שורד כמות זמן לא פרופורציונלית לניקוד שהשיג.

**Alpha-Beta Pruning:**



**תיאור:**

רצינו לנסות לאפשר לפאקמן לראות מעט רחוק יותר מ-2 מהלכים קדימה, ולתת לו אפשרות גם להבין שלפעמים אפילו אם מהלך הוא מסוכן יחסית, אולי אפשר להשיג ניקוד בנוסף. בשביל זה מימשנו את אלגוריתם Alpha-Beta Pruning, שמנמיך את כמות החישובים שצריך לבצע ב-Minimax, בלי להתפשר על איכות התוצאה.

**תוצאות (עומק העץ – 3):**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Ghosts \ avg. stats** | Score | Time | Lives | Maze |
| Random | 2907 | 4720 | 0 | 0.66 |
| Starter | 1449 | 2122 | 0 | 0.28 |
| Aggressive | 2171 | 1045 | 0 | 0 |
| Legacy2TheReckoning | 1778 | 4228 | 0 | 0.68 |

**מסקנות:**

מבחינת ניקוד, השיפור אינו משמעותי. אך נשים לב לכך שהיחס גדל בממוצע (כלומר, השגנו יותר נקודות פר יחידת זמן שפאקמן שרד). יחד עם זאת, פאקמן עדיין מעט זהיר מדי. הוא מוותר על הזדמנויות להשיג ניקוד כדי לשחק יותר בטוח, ולהבטיח כמה שיותר זמן הישרדות. נרצה אלגוריתם שמאזן זהירות, עם השגת ניקוד גבוה ככל הניתן.

 **Monte-Carlo Tree Search:**

**תיאור:**

באמצעות אלגוריתם MCTS נוכל למצוא את האיזון הרצוי. האלגוריתם יריץ מספר סימולציות של מהלכים אפשריים, ויראה מתי אפשר להשיג כמה שיותר נקודות, תוך המנעות מפסילה. עבור פונקציית הבחירה, נשתמש ב-UCT:



הנוסחא מספקת ציון המתבסס על הערכה של פעולה a ממצב s, תוך תשומת לב לכמות הפעמים שחקרנו את האיזור בו s נמצא.

**תוצאות:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **MCTS** | Score | Time | Lives | Maze |
| Random | 33449 | 22789 | 1.925 | 7.72 |
| Starter | 7391 | 4952 | 0 | 1.215 |
| Aggressive | 15621 | 4650 | 0 | 1.215 |
| Legacy2TheReckoning | 4842 | 2356 | 0 | 0.265 |

**מסקנות:**

תוצאות האלג' טובות יותר מכל אלגוריתם אחר שניסינו. האלג' שם דגש על איסוף כמה שיותר נקודות, אבל מונע את ה-"פרנויה" והחשש הסתמי שהיה באלג' Minimax, בכך שהוא משחק עד לנקודת זמן עמוקה יותר, ובאמצעותה מתעלם ממצבים מסוכנים שאולי חווה בדרך אם הם לא גרמו לפסילה.

**השוואת השיטות השונות:**

Chart, bar chart, waterfall chart

Description automatically generated

**סיכום:**

מטרתנו היתה לעזור לפאקמן להשיג כמה שיותר נקודות במשחק. כדי לעשות זאת ציינו שצריך לאכול pills, אך יחד עם זאת לשמור מרחק מהרוחות העוינות שרוצות לפסול אותו. תחילה, עשינו שימוש בהיוריסטיקה, שם הראינו שפאקמן משיג מספר נקודות גבוה, אך ככל שהרוחות הופכות חכמות יותר, ביצועיו מדרדרים. לאחר מכן, נתנו לפאקמן את האפשרות לראות מעט קדימה עם אלגוריתם Minimax ולאחר מכן Alpha-Beta Pruning, אך ראינו שזה גורם לו להיות פרנואיד, ובכך הוא יותר זהיר מאשר חכם, מה שמביא לפער גבוה בין השרידות שלו לניקוד שהוא משיג. לבסוף, הראינו כי MCTS הוא האלגוריתם הטוב ביותר למשימה. MCTS משלב רדיפה אחר ניקוד, כמו ההיוריסטיקה שלנו, יחד עם זהירות וצעדי ביטחון, כמו Minimax. דרך האופטימיזציות השונות שחקרנו, גילינו כיצד לגשת לפתרון הבעיה, וכיצד לעבור את המכשול שאליו נקלענו בכל פעם. ראינו איך שיטות שונות ואלגוריתמים שונים גורמים לפאקמן לנוע על גבי הלוח, ומה כל אחד מתעדף (ביטחון מול חמדנות למשל). חקירת סביבות זמן-אמת דומות לזו שלנו, יכולה להוביל להבנה עמוקה יותר של סביבות זמן-אמת נוספות, ולמציאת אופטימיזציות בטוחות וטובות יותר בעתיד.