Introduction to Artificial Intelligence - 236501

AI HW2 spring 2022

tomsmolin@campus.technion.ac.il	313552739	תום סמולין
noamwolf@campus.technion.ac.il	326881240	נעם וולף

חלק א' – סוכן חמדן משופר

א. נגדיר היוריסטיקה $h(s)$ חדשה שתכלול לפחות 3 פרמטרים נוספים מלבד פרמטר הכסף.
באופן כללי, רעיון היוריסטיקה הוא לעשות כמה שיותר נסיעות קצרות ומשתלמות (מונית תל אביבית ולא מונית ספיישל בינעירונית).
ראשית נגדיר מספר ערכים נוספים (כדי לא להעמיס על ההגדרה עצמה בהמשך).

- **עבור נוסע ps בסריג שמחכה למונית נגדיר את ערך התשלום עבור הסעתו במונית:**

$$compensation(ps) = MD_{ps.start\ ,ps.destination}$$

- **עבור כל נוסע ps שמחכה לאיסוף במצב נתון s (ובתנאי שהוא לא מחכה ביעד – $compensation(ps) \neq 0$), הערך $val(ps)$ הוא:**

$$12 + 12 \cdot taxi.cash - (MD_{taxi.pos\ \ ,ps.pos} + MD_{ps.start\ ,ps.destination})$$

- **נגדיר חישוב $taxiWithPassenger(s)$ שמוגדר רק עבור מצבים בהם ישנו נוסע ps על המונית. החישוב יוגדר כך:**

$$12 + 12 \cdot taxi.cash - (MD_{taxi.pos\ ,ps.destination})$$

- **נגדיר לכל s חישוב $goodPassengersExists(s)$:**
שמחזיר $True$ אם קיים נוסע ps שמחכה למונית כך ש $compensation(ps) \neq 0$.
אחרת, מחזיר $False$.

<u>כעת נעבור להגדרה של היוריסטיקה עצמה:</u>

$$h(s) = \begin{cases} \max\limits_{\substack{ps \in env.passengers \\ compensation(ps) \neq 0}} \{val_{ps}\}, & taxi\ is\ empty\ AND\ goodPassengersExists(s) \\ 0, & taxi\ is\ empty\ AND\ !\ goodPassengersExists(s) \\ taxiWithPassenger(s), & passenger\ on\ taxi \end{cases}$$

נשים לב שהפרמטרים הנוספים בהם השתמשנו (3 במספר):

- $MD_{taxi.pos\ \ ,ps.pos}$ – מרחק מנהטן בין המונית הריקה לנוסע מסוים שממתין למונית (תחת התניה נוספת שהיעד שלו הוא לא מיקומו הנוכחי)
- $MD_{ps.start\ ,ps.destination}$ – מרחק מנהטן בין מיקומו ההתחלתי של הנוסע ליעדו הסופי (מחושב רק עבור נוסעים שמקיימים $compensation(ps) \neq 0$)
- $MD_{taxi.pos\ ,ps.destination}$ - מרחק מנהטן בין מיקום המונית לבין מיקום יעד הנסיעה (של הנוסע במונית).

ב. כיוון שהיוריסטיקה מחלקת את המצבים למקרים, נתייחס למוטיבציה לפי מקרים:

- מקרה א' – המונית ריקה וקיים נוסע שעבור הסעתו ניתן לקבל תשלום חיובי ממש.
במקרה זה היוריסטיקה היא כאמור:

$$12 + 12 \cdot taxi.cash - (MD_{taxi.pos\ ,ps.pos} + MD_{ps.start\ ,ps.destination})$$

המוטיבציה היא להגיע ולהסיע את הנוסע שעבורו מתקבל המסלול הכולל הקצר ביותר (רעיון – לעשות כמה שיותר נסיעות קצרות ולמקסם את מספר הנוסעים) וגם לקבל כמה שיותר כסף.
כיוון שבחרנו לממש סוכן שמעוניין למקסם את היוריסטיקות שלו – המקדם של $taxi.cash$ חיובי והמקדם של "אורך המסלול: מונית->נוסע->יעד" הוא שלילי (רוצים ערך קטן ביותר).
בפרט, פרמטר הכסף עוזר לנו לקבל החלטה להוריד את הנוסע ביעד, שכן במצב שבו אנחנו עם נוסע ובאחד המצבים העוקבים אנחנו יכולים להוריד אותו – נרצה שהתשלום הנ"ל יילקח בחשבון בהשוואה ליתר המצבים העוקבים (ביתר המצבים הנוסע עדיין יהיה על המונית). לכן גם הכפלנו ערך זה ב12, כדי לתת לו משקל גדול יותר ביחס ליתר הפרמטרים.

- מקרה ב' – המונית ריקה, אבל לא קיים נוסע שעבור הסעתו ניתן לקבל תשלום חיובי ממש.
במקרה זה היוריסטיקה מחזירה 0. שכן, או שיש שני נוסעים שכבר נמצאים ביעד (ולכן אין טעם להגיע אליהם), או שיש נוסע זמין אחד שנמצא ביעד (ולכן אין טעם להגיע אליו) ונוסע נוסף במונית השנייה (לא נגיש עבורנו).
במקרה זה הפעולות החוקיות הן תנועות בכיוון מסוים או איסוף של נוסע (לא רווחי). כיוון שאנחנו רוצים להישאר זמינים לנוסעים חדשים שעבורם נקבל תשלום, נבצע תנועה אקראית באיזשהו כיוון (כיוון שלפי הפיאצה אנחנו לא יכולים להישאר במקום). בפועל הבחירה בתנועה (ולא באיסוף אם אנחנו נמצאים "על" נוסע) היא כחלק מאקט של "שבירת שוויון". שבירת השוויון באה לידי ביטוי במימוש – תנועה בכיוון מסוים קודמת לאיסוף נוסע (בסדר הפעולות שהוגדר בסביבה), ולכן בקריאה ל max על כל הבנים שבמקרה זה יהיו עם אותו ערך – נבחר את הפעולה הראשונה, שהיא תנועה בכיוון צפון (אם אפשרית, אחרת ננסה את הכיוון הבא).

- מקרה ג' – יש נוסע על המונית עבורו אפשר לקבל תשלום.
במקרה זה היוריסטיקה היא כאמור:

$$12 + 12 \cdot taxi.cash - (MD_{taxi.pos\ ,ps.destination})$$

כדי לא לסבך את היוריסטיקה יותר מדי עם חישובים מורכבים, גם למצב הזה אנחנו מתייחסים בצורה חמדנית, ובוחרים קודם כל להוריד את הנוסע.
כלומר, ניתן עדיפות למצבים שמקרבים אותנו ליעד באמצעות הפרמטר $MD_{taxi.pos\ ,ps.destination}$.
ככל שנהיה יותר קרובים, הערך שהיוריסטיקה מחשבת יהיה יותר גדול. כיוון שאנחנו מנסים למקסם את הערך בין הבנים (המצבים העוקבים), נבחר במצב הקרוב ביותר להורדה.
בנוסף, אנחנו עדיין נותנים משקל לכסף. כלומר, גם אם נעבור בתחנת דלק – נעדיף להמשיך אל היעד ולהוריד את הנוסע כמה שיותר מהר (כי הסוכן אוסף רק נוסעים "רווחיים", והגישה החמדנית שלנו במקרה הזה, רוצה לממש את הרווח קודם כל).

נשים לב שבחרנו להוסיף 12 להיוריסטיקות במקרים א', וג' כדי לשמור על היוריסטיקה אי שלילית.
נשים לב שמרחק המנהטן המקסימלי בגריד הוא 6 (מרחק מנהטן בין שתי פינות נגדיות).
לכן:

$$MD_{taxi.pos\ ,ps.pos} + MD_{ps.start\ ,ps.destination} \leq 12$$
$$(MD_{taxi.pos\ ,ps.destination}) \leq 6$$

ולכן:
לכל $s \in S$:

$$0 \leq h(s)$$

לדעתנו סוכן חמדן המבוסס על היוריסטיקה שלנו ינצח את הסוכן החמדן הנתון (לפחות במרבית המקרים) כיוון שניתן לומר שהוא "יודע" יותר על העולם. נסביר:
הסוכן החמדן הנתון, מחשב לכל מצב את ההפרש בין מאזני הכסף של המונית שהוא מייצג למונית של היריב. כעת, נשים לב כי בזמן שמגיע תורו של הסוכן הנתון לשחק סכום הכסף של המונית היריבה זהה עבור כל המצבים העוקבים (למצב הנוכחי של הסוכן). המונית היריבה "סטטית" בשלב הזה.
לכן, הדרך היחידה שלו לייצר איזשהו אי-שוויון בין הבנים היא במקרה של הורדת נוסע (או תדלוק. נניח ברגע שהתחלנו לשחק ואין כסף לתדלק).
אבל, כדי להוריד נוסע צריך קודם כל לאסוף אותו. כיוון שאנחנו לא מרוויחים כסף מאיסוף הנוסע, הסיכוי היחיד שהסוכן יבחר לאסוף את הנוסע היא אם תהיה בחירה אקראית שכזו כשהמונית תעמוד "על" הנוסע (כי שוב, הערכים של כל הבנים יהיו זהים. למעט אולי מצב עוקב עבורו מתדלקים).
גם אחרי איסוף הנוסע, זה עניין של מזל בכלל להגיע ליעד של הנוסע (מסיבות דומות – הפרש הכספים לא עוזר לנו להבין איפה היעד, אלא אם כן אנחנו עומדים עליו).
כלומר, אפשר להגיד שהסוכן הנ"ל הוא עם "ידע" מאוד מוגבל על העולם בכל רגע נתון. אמנם המטרה הסופית שלו, היא אכן לקבל המקסימום של הערך אותו הוא בא לחשב בכל מצב - אך כאמור, חישוב ערך זה כ"ערך היוריסטי", לא משרת אותו היטב כדי לנצח.

*כדי לחדד את עניין התדלוק בסוכן הנאיבי שמימשתם – גם במקרה שהוא כן הצליח איכשהו לאסוף ולהוריד נוסע ביעד, הוא עדיין יעדיף לא לתדלק (ייתן ערך היוריסטי יותר קטן, בעוד הסוכן הנ"ל מעוניין למקסם) ולכן לא יבחר בפעולה זו.

הסוכן שלנו לעומת זאת, מגיע עם "ידע" על העולם. כלומר, הוא יודע איפה הנוסעים ממוקמים והוא "יודע" איפה להוריד אותם (מהסיבות שציינו).
ולכן הוא "מחפש" (או בוחר לבצע) באופן אקטיבי את הנסיעות הקצרות ביותר שישתלמו לו עד שיגמר הדלק (אפשר היה לעשות היוריסטיקה יותר מורכבת ויותר מתוחכמת, אבל לא נדרשנו לכך).

להלן התוצאות הסופיות בסדר שבו התבקשנו להציג אותן (ועם תיוג של הסוכנים לטובת הבהירות):

```
['greedy', 'greedy_improved']
[0, 10]
taxi 1 wins!
```

```
['greedy', 'random']
[0, 0]
draw
```

```
['greedy_improved', 'random']
[10, 0]
taxi 0 wins!
```

חלק ב' – סוכן Minimax

א. מקרה פשוט יכול להיות בו לסוכן היריב נגמר הדלק והוא מחזיק x כסף כך ש $x \geq 0$.
לסוכן של החבר (מינימקס) יש $y > x$ והגיע תורו לשחק. נניח שיש לסוכן של החבר עוד יח' דלק 1 והוא עומד ריק במיקום של נוסע כלשהו. עכשיו, הוא יכול לסיים את המשחק ולנצח אם הוא יעשה פשוט תנועה בכיוון כלשהו (סכום הכסף לא ישתנה וגם יגמר לו הדלק). אבל אז ערכי המינימקס של איסוף הנוסע או התנועה צפויים להיות שווים (או במקרה של סוכן מינימקס מוגבל משאבים, ייתכן יתרון לאיסוף – תלוי בהיוריסטיקה) ואז תבחר פעולת האיסוף. כלומר, במקום לסיים את המשחק ולנצח, הסוכן בחר לבצע עוד תור ולבסוף לנצח במספר גדול יותר של צעדים ממה שיכל בפועל.

מקרה נוסף ועבור אותם תנאי התחלה שתוארו לעיל – נניח שהסוכן של החבר עומד על תחנת דלק. כעת, אלגוריתם המינימקס מכוון אותו לבחור בצעד התדלוק מתוך "הנחה" שניתן למקסם כך את הרווח הסופי של המונית שלו (למשל, אם חסר לו דלק כדי להוריד את הנוסע שנמצא עליו עכשיו, והרווח הצפוי משתלם). כלומר, האלגוריתם מבין שהוא יכול להבטיח תועלת טובה יותר אם יתבצע הצעד של התדלוק (תחת ההנחה שאפשר גם להמשיך לשחק). לכן נקבל שהסוכן של החבר יתדלק ולכן סכום הכסף שלו יהיה 0 (מתדלקים עם כל הכסף). עכשיו אם אין הגבלת צעדים, ייתכן מאוד שהוא ימשיך להורדת הנוסע ובסופו של דבר ינצח.
*** אבל אפשר למשל להניח שזה היה הצעד האחרון להיום (לא נכתב בשאלה שזה לא אפשרי).
כלומר, בסופו של התור הנ"ל נגמר המשחק.
קיבלנו תרחיש שבו הסוכן של החבר לא מנצח כיוון ש $x \geq 0$ כשהוא היה יכול לבחור צעד אחר (שאינו תדלוק) שהיה מבטיח לו את הניצחון (אם כי בפיאצה נאמר להתעלם ממקרה כזה, אבל הוא עדיין אפשרי בעינינו).

שינוי פשוט שניתן להציע כדי למנוע התנהגות כזאת – להתייחס בצורה פרטנית למקרים בהם לסוכן שלנו יש יותר כסף מלסוכן השני ולסוכן השני נגמר הדלק. במקרים כאלה נגדיר לסוכן של החבר לשבור שוויון בין ערכים זהים, באמצעות תיעדוף תנועה על מנת לשרוף דלק. התועלת שנבטיח תהיה לפחות y ובאופן זה נבטיח ניצחון, שזו המטרה הסופית (ובפרט בדרך המהירה ביותר – שכן המשחק נגמר כשנגמר הדלק לכולם או כשנגמר מספר הצעדים היומי. תנועה בכל כיוון מקדמת אותנו בכל אפיק שכזה לסיום).

ב. שני יתרונות של $Iterative\ Deepening\ with\ Minimax$ על אלפא-בטא:
   a. הוא לא חייב לחכות לסיום מלא של החיפוש והוא **מסוגל לספק "תוצאות ביניים" (עבור אלגוריתמי $Anytime$ למשל)**.
   b. **האלגוריתם (לפחות בסבירות גבוהה) מספק אינטראקטיביות בצורה מובנית. הוא לא "תוקע" אותנו במצבים בהם אנחנו מוגבלים במשאבי חישוב (ליבות חישוב נניח)**.
   אנחנו לא מחויבים לחישובים "ארוכים". לכן אם המשאבים מוגבלים ניתן לנהל את הריצות בצורה גמישה.
   כלומר, לעומת אלפא-בטא יש לנו את הכוח "להפקיע" חישובים בקלות ובצורה עצמאית מבלי להסתמך בצורה עיוורת על מערכות ההפעלה (המשאב אליו אנחנו מתייחסים במקרה זו היא ליבת חישוב). לדוגמא, ניתן לאפשר "הוראות" מהסוג הבא לרובוט שמשחק בעולם ומופעל ע"י סוכן כלשהו – "חשב איטרציה עד עומק מסוים. עצור (בזמן סביר). פידבק לא מספיק טוב? חכה רגע. הרץ משהו אחר שלא בהכרח קשור באופן ישיר להחלטה במשחק על הליבה היחידה שלך (אולי זה רובוט שעושה דברים במקביל בעולם). בצע איטרציה נוספת עמוקה יותר".

שני יתרונות עבור $alpha - beta$ על העמקה הדרגתית עם מינימקס:

a. כפי שראינו בתרגול וכפי שמופיע בספרות (חקירה באינטרנט), בעוד שסיבוכיות מספר הפיתוחים ב העמקה הדרגתית עם מינימקס היא $O(B^D)$ כך ש D הוא העומק המקסימלי אליו מגיעים בהעמקות, סיבוכיות הפיתוחים באלפא בטא היא כזאת במקרה הרע (עבור אותו עומק מקסימלי). במקרה הטוב היא $O(B^{\frac{D}{2}})$ ובמקרה הממוצע $O(B^{\frac{3D}{4}})$.

b. מאפשר ניצול סלקטיבי של הזמן. הוא נמנע מלעבור על המסלולים שלא ישנו את ערך המינימקס או אסטרטגיית המינימקס (כלומר מסלולים שלא ייבחרו במינימקס ובפרט כאלו עם העמקה הדרגתית).

ישנם מספר שינויים שניתן לבצע כדי לשפר את השיטות הנ"ל באופן כללי או בהקשר של המשחק שלנו.

את השיטה הראשונה, ניתן לשפר באמצעות א) גיזום כמו שציינו לעיל ו ב) במשחק שלנו ניתן לשלב היבטים נוספים, מהירים לבדיקה שיכולים להבטיח את הניצחון (המטרה הסופית במשחק). למשל כמו בחלק א', במקרים בהם לסוכן השני נגמר הדלק ולנו יש יותר כסף אז נדע לעצור את אסטרטגיית האלגוריתם כפי שהוא מוגדרת, כדי לעבור לאסטרטגיה שונה דומה למה שתיארנו בחלק א' (שבה מתעדפים שריפת דלק בכל שלב).

את השיטה השנייה ניתן לשפר באמצעות א) מיון של הבנים כפי שראינו בתרגול כדי למקסם את כמות הגיזומים (בנים של צמתי מקס בסדר יורד, בנים של צמתי מינימום בסדר עולה) וב) באמצעות מימוש העמקה דרגתית על בסיס האלגוריתם הנ"ל כפי שעשינו עם מינימקס (ואז לקבל את היתרונות של אלגוריתמי $Anytime$ למשל).

חלק ג' – סוכן Alpha-Beta

א. מימוש.

ב. השינויים שהיינו צריכים לעשות הם כלהלן:
היינו צריכים להכניס לתוך האלגוריתם (ולפיכך למימוש) גם את התורות של יתר המוניות ($k-2$ נוספות). כלומר, אם נדמיין את "עץ המשחק" שמתפתח, אז "תור" המונית הראשונה יופיע אחת ל k שלבים בעץ (ולא 1 ל2). כמו כן, במימוש של ה $rb-alpha-beta$ במקום לשמור דגל בוליאני שמתעד את הסוכן שזהו תורו (דהיינו הסוכן שמחשב את הצעד הבא שלו – או הסוכן שמולו הוא מתמודד) היינו ככל הנראה מעבירים ברקורסיה את מספרו של הסוכן הבא ש"מסמלץ" (וזוכרים באופן כלשהו איזה סוכן מחשב את הצעד, כלומר הסוכן שנמצא בשורש וקרא לאלג').
בנוסף, אם נרחיב על השינויים ומה שעומד מאחוריהם, אז בכל צומת בעץ נצטרך להחזיק $tuple$ בעל k ערכים, המתאר את היוריסטיקות (אם לא מצב סופי) או מצב הכסף (מצב סופי למשל) עבור כל מונית.
נשים לב למספר דברים נוספים – **כשמדובר ביותר משני שחקנים, שיטת המינימקס כפי שהיא פחות מתאימה לפי הספרות** (שכן בשיטה זו מדובר בהתמודדות של שני שחקנים, ובצורה פשטנית ניתן לומר שאם ננסה למזער את "תוצאת" היריב נשפר את הסיכוי שלנו לנצח אבל מוטיבציה זו לא בהכרח באה לידי ביטוי כשיש הרבה שחקנים) **ולכן היינו בוחרים לממש גרסה מעט שונה** של האלגוריתם (באותה הרוח) באופן כזה שיתאים למספר רב של סוכנים. **סביר להניח שבכל שלב היינו מחפשים פשוט למקסם את התוצאה של השחקן "שמסמלץ"** (כלומר אם אנחנו בצומת שמהווה בחירה של שחקן i אז שחקן זה יבחר בצומת הבן עם רכיב ה i המקסימלי בתוך ה $tuple$). לכן הגיזומים לא יגזרו מערכי אלפא ובטא, אלא מסדרה של ערכים שמתארים את המצב המקסימלי ששחקן מסוים יכול להבטיח.
*נדגיש שאנחנו העלינו את הרעיון הכללי, אך בספרות (ממה שחקרנו) כתוב שאם אין חסמים על סכום הרכיבים בכל $tuple$, אז גיזום של ענפים שלמים פחות סביר (אם אנחנו מבינים נכון, זה כי אין שום הבטחה שלא יצוץ מצב מאוד שווה עבור שחקן מסוים בהמשך המשחק) ולכן הגישה הזאת לא התפתחה.
אבל ברגע שיש חסם על סכום התוצאות של כל שחקן, ניתן להניח חסמים תחתונים וחסמים עליונים לתוצאות של השחקנים השונים ולבצע את הגיזומים ע"פ ערכים אלו (שכן אם שחקן1 מבטיח לפחות x אז שחקן2 יכול לקבל לכל היותר $limit-x$ וע"פ זה ניתן לקבל החלטות "מבטיחות" יותר).
****אם לא נלך על הגישה כפי שהיא מופיעה בספרות, אזי נגדיר עבור הסוכנים היריבים במטרה למזער את התוצאות של השחקן בשורש. במקרה כזה כן נתחזק את משתני אלפא ובטא (שכן המוטיבציה של כל השחקנים היריבים היא משותפת).
בפועל – אם נשאיר את הגבלת הזמן לכל חישוב צעד של סוכן ונדאג לבצע "$hard\ exit$" בזמן, זמן הריצה לא אמור להשתנות יותר מדי.
אבל, אותו החישוב יתבסס על ניתוח של פחות צעדים קדימה במשחק. שכן ככל שנוסיף שחקנים ייקח לנו יותר זמן לחשב את אותה כמות של סבבי תורות.
יתרה מזאת, מהסיבות שציינו לעיל – אם אין חסם על סכום הרכיבים ב $tuple$ אז היכולת לבצע גיזומים מידיים (בספרות $Immediate\ pruning$) היא הרבה פחות טריוויאלית (הרי כל שחקן יחפש למקסם עבור עצמו. ואם אין חסם, אין הבטחה שהוא לא יכול למקסם "בהפתעה" בעתיד). לכן גם סביר שנעשה פחות גיזומים.
פחות גיזומים -> זמן ריצה ארוך יותר תחת חישוב של אותה כמות צעדים לסוכן (בפועל אם מגדירים כמות דלק התחלתית קבועה לכל משחק אז כנראה שניתן להגדיר איזשהו חסם גס על כל רכיב ב$tuple$ ולכן גם על סכום הרכיבים. אבל בפרט הגיזום עדיין לא יהיה מיידי כמו זה המתקבל במינימקס עם שני סוכנים ולכן נניח שזמן הריצה יהיה ארוך יותר, עבור אותו מספר סבבים).

ג. בשני החלקים אנחנו משתמשים ב $iterative\ deepening$ ועבור צמתים ששייכים לקב' המצבים הסופיים מחזירים את הפרש הכספים ועבור מקרים בהם לא הגענו למצב סופי מחזירים את אותה היוריסטיקה שפיתחנו בחלק הראשון.

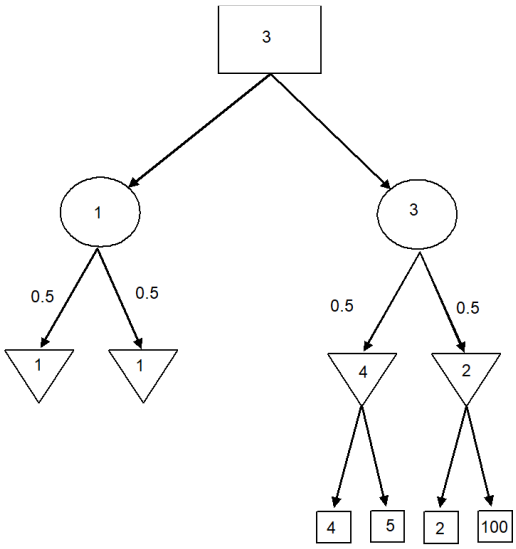
כעת נשים לב, שהאלג' של חלק ג' הוא מעין שיפור של האלג' של חלק ב' (לפחות מבחינת היעילות ובהינתן שיש לנו היוריסטיקה סבירה) שכן הוא מוותר על מסלולים מיותרים בעץ המשחק. **לכן אכן, ברוב הגדול של ההרצות אם נחשב את ממוצע זמני הריצה של כל סוכן (עבור אותם מצבים התחלתיים), נקבל שממוצע זמני הריצה של סוכן ה $rb-alpha-beta$ הוא קצר יותר** תחת תנאי המשחק הנתונים (אלפא-בטא ימצה מהלכים מהר יותר אם לשניהם יש מספיק זמן להעמיק עד הסוף למשל).
יחד עם זאת נציין כי אנחנו קוטעים את ריצת ה $Iterative\ Deepening$ אם עברנו אחוז מסוים מהזמן שהוקצב לנו לתור (כדי לעמוד בהגבלת הזמן במימוש). רוצים לומר – הסוכנים רצים פחות או יותר פרקי זמן דומים מבחינה אבוסולוטית. לכן לעיתים, במקרים שבהם לא הגענו למצבים סופיים בהעמקה שלנו – את הזמן שסוכן האלפא-בטא היה אמור לחסוך, הוא מנצל להעמקה שלא מתאפשרת בזמן הקיים עבור סוכן המיני-מקס. לכן יש צעדים שבהם הזמנים כן יכולים להיות שווים ואף ארוכים יותר עבור סוכן האלפא-בטא.

מבחינת קבלת ההחלטות, תחת היוריסטיקה שפתחנו (יוריסטיקה שמגדירה התנהגות של מונית לנסיעות קצרות, מה שכינינו מונית "עירונית"/"תל אביבית") **הסוכנים מקבלים החלטות זהות במרבית המקרים.** לעיתים יש הבדלים כשסוכן האלפא-בטא מספיק להעמיק יותר או דווקא כשהוא בוחר לגזום בנים עם ערכים שווים (לאלפא או לבטא. התנאי הוא קטן/שווה או גדול/שווה) וע"י כך לוותר על מסלולים לאו דווקא מיותרים (אם היוריסטיקה לא דייקה) ולהשפיע על בחירת הצעדים הבאים (ומכאן לייצר את ההבדל בקבלת ההחלטות).

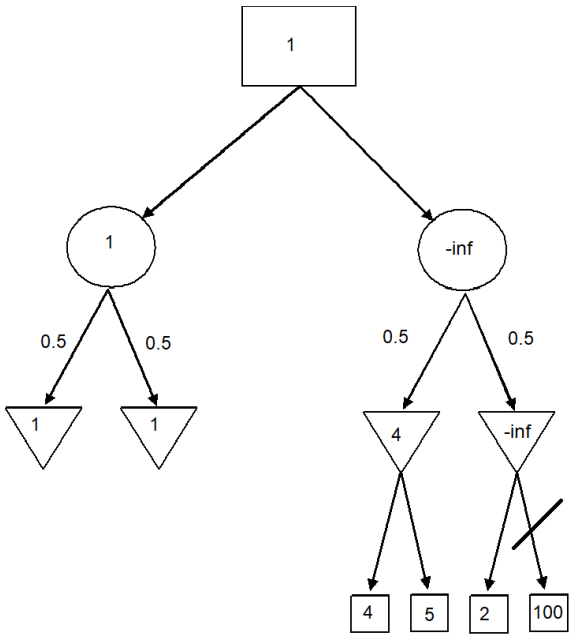
חלק ד' – סוכן Expectimax

א. כן, מדובר בבאג. השימוש בגיזום כזה לא מתאים לexpectimax, מפני שזה ממש הורס את חישוב התוחלת לאחר מכן. כלומר אם מגיעים למצב בו הוחלט לגזום ומחזירים ליריב אינסוף, על מנת שלא יבחר את הצומת הזה, זה הורס לגמרי את התוחלת המוחזרת של אותו בן. נתבונן בדוגמה הבאה, בו סוכן זה ריבוע, יריב משולש וצומת הסתברותית עיגול:

כאשר אין גיזום נבחר ללכת ימינה בסוכן.



אך אם נגזום, נבחר ללכת שמאלה.



כלומר הגיזום גרם לתוצאה שהיא שונה, וגם לא נכונה.

שאלה פתוחה

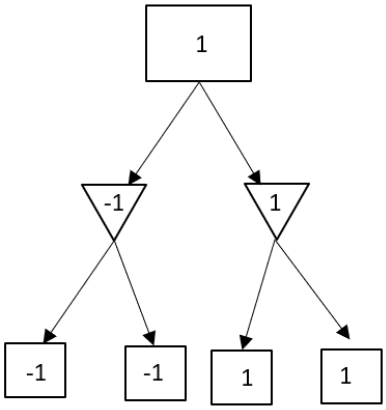
א. ניתן להשתמש בפונקציית תועלת שראינו בתרגול, כאשר n הוא מספר הנהגי מונית, s מצב בסיום וk מספר של נהג מסוים. נגדר נהג מנצח אם יש לו הכי הרבה כסף בסוף היום. נגדיר תיקו אם קיימים שני נהגים או יותר אם אותה כמות של כסף.

$$\boldsymbol{U}(\boldsymbol{s},\boldsymbol{k}) \triangleq \begin{cases} n-1 & P_G(\boldsymbol{s}) \wedge player\ \boldsymbol{k}\ wins \\ -1 & P_G(\boldsymbol{s}) \wedge player\ \boldsymbol{k}\ loses \\ 0 & P_G(\boldsymbol{s}) \wedge tie \\ Undefined & NOT\ P_G(\boldsymbol{s}) \end{cases}$$

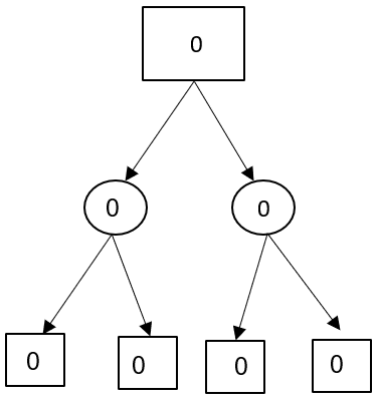
ב. שיניתי את התשובה לאחר העדכון. כן, יכלים להיות שינויים מהותיים. אם למשל מדובר במקרה בו יש שני שחקנים בלבד, אין הבדל בין התועלת של ניצחון והפסד. כלומר אם n=2 אז:

$$\log(\mathrm{n}-1) = \log(2-1) = \log(1) = 0 = -0 = -\log(1) = -\log\big(-(-1)\big)$$

ולכן נוצר מצב שבוא נבחר באופן שרירותי, או על פי המיקום בעץ את הצמתים שנחליט לבחור בשביל היריב ובשביל וסוכן. הדוגמה הבא מדגימה את זה. בדוגמה מצד ימין הסוכן ינצח בכל הבחירות של היריב, ובצד שמאל הוא יפסיד. לכן במינימקס הסוכן יחליט ללכת ימינה על מנת להבטיח ניצחון.



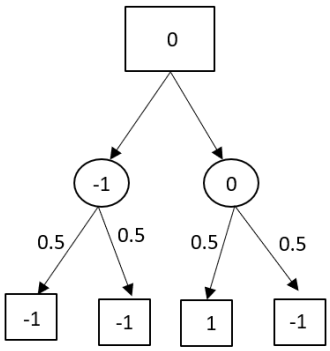
אם במקום להחזיר את ערך התועלת, נחזיר את הערך של הפונק' הלוגריטמי הנתון נקבל את התוצאה הבאה:



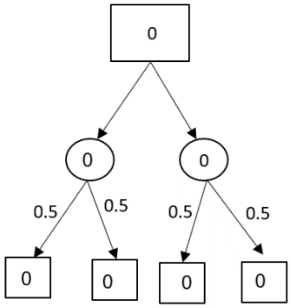
קיבלנו שכעת אין העדפה לסוכן לבחור בבחירה הימנית למרות שזה יגרום לו לניצחון ודאי, ויבחר צעד על בצורה רנדומית/על פי סדר המעבר.

ג. כעת כבר לא יהיה שינוי, אמנם כנראה שלא יהיה מדובר בפונקציית תועלת שהיא סכום אפס, כי כאשר יש מעל 2 שחקנים הסכום שצובר המנצח יגדל אקספנציונלי. מדובר בפונקציה מונטונית עולה, ולכן לכל המפסידים, הערך נשאר -1,(-1)^7 = -1) ובמקרה של תיקו נשאר 0, ולכן בחירות הסוכן יהיו זהות, כי נשמר הסדר בין האופציות, ובכל שלב שניתן הסוכן יבחר לנצח, לאחר מכן לעשות תיקו ואז בלית ברירה להפסיד.

ד. עבור הפונקציית תועלת של שמחה, הבעיה שתיארנו לעיל תשאר, כלומר במידה ומדור בשני שחקנים, נגיע למצב בא הערך של הצומת ההסתברותית מחשובת להיות 0, גם כאשר יש מנצח. נראה זאת:
במידה ואנחנו משתמשים בexpectimax הרגיל, עם שני שחקנים בו היריב הוא רנדומי עם התפלגות אחידה מתואר מטה מקרה בו יש שני אופציות ליריב, 3 בהם היריב מפסיד, 1 בו הוא מנצח. על פי האלגוריתם הסוכן יבחר באופציה הימנית, כי זה האופציה היחידה בה הוא אולי ינצח.(וגם עם הערך המקסימלי)



אך במקרה בו מחזירים את הערך הלוגריטמי כמו שמתואר בשאלה, אם יש רק שני שחקנים אז גם להפסד לניצחון יש ערך 0, ולכן התוצאה תראה כך:

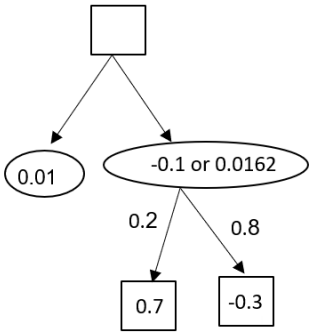


והסוכן יבחר מסלול באופן אקראי/הראשון שהוכנס לתור ויתכן מצב בו הוא בוחר את המסלול המוביל להפסד ודאי שלו.

במקרה בו מחזירים את הערך של התועלת בחזקת 7, גם כן יהיהלהיות שינוי בתוצאה. נביט הדוגמה הבאה, כאשר צומת יריב מסומן במשלוש, רנדומי בעיגול ובסוכן במרובע.

כאשר הכל תקין: $0.7 * 0.2 + (-0.3) * 0.8 = -0.1$

אם מחזירים את הערך בחזקת 7: $0.2 * (0.7)^6 + 0.8 * (-0.3)^7 = 0.0162$



ניתן לראות כי כאשר הערך התקין מוחזר, נבחר ללכת שמאלה, אך במידה והערך בחזרת 7 יותר נבחר ללכת ימינה. בכלל ההיפוך של הסימן. זה מובן מפני שמדובר בפעולות לא ליניאריות ולכן משפיעות על התוצאות. נניח גם כן שמדובר בפונקציה סכום אפס (אם פונקציה שנותנת למפסיד את הערך השלילי של המנצח.)

ה. א. ניתן להשתמש בפונ' על מנת לגזום בלי לפגוע בנכונות לו. נוסיף את הקטע קוד הבא לאלגוריתם, לאחר הבדיקה האם הגענו למצב סופי:

```
Alpha = max{Alpha, f(s) – 2)

Beta = min(Beta, f(s) + 2)

If(Beta ≤ f(s) – 2) then return (f(s) – 2)

If(Alpha ≥ f(s) + 2) then return f(s) + 2
```

הסבר: הערך Alpha מייצג חסם תחתון של ערך המינימקס המקסימלי של צומת מקסימום עד כה, והערך f(s) – 2 גם כן מייצג חסם תחתון על הערך המינימקס של הצומת s. כלומר אם ניקח את המקסימום מבניהם, אז נקבל חסם יותר הדוק של הערך מינימקס המקסימלי של s.
לכן נוכל להשתמש בערך f(s) + 2 כדי לבדוק האם לגזום או לא, כי נובע שהערך מינימקס של צומת s קטן או שווה לalpha ולכן נוכל לגזום, כי אין טעם לבדוק את בניו.
ההסבר על beta הוא סימטרי.

ב. לא אי אפשר להשתמש בזה לגיזום, כי אין קשר בין ערך היוריסטי שעליו מתבסס המשפט לבין הפונקציה. כלומר, אם רק נגיע לעומק נמוך, למשל 1, אז יכול להיות שהערך היוריסטי אינו קשור לנוסחה הנ"ל. בעצם לא ניתן לחסום את המרחק בין ערכי הצמתים לבין ערכי f ולכן לא ניתן לדעת האם אפשר לגזום או לא.