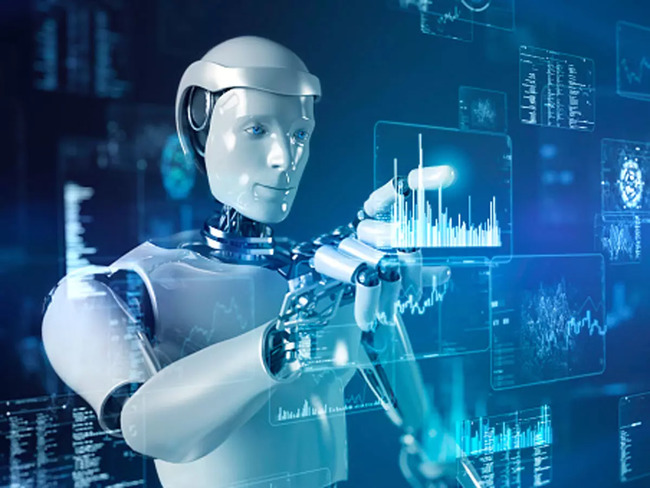
Connect Four advies report

Quinten Bakker





Inhoud

* Probleem stelling
* Welke algoritme heb ik gekozen
* Waarom heb ik dit algoritme gekozen en niet een andere?
* Etnische keuze
* Voor en nadelen
  + Voordeel
  + Nadeel
* Analysis
* Bronvermelding

Probleemstelling

Welke moeilijkheden moest ik overkomen om dit algoritme te maken? Als een grootte probleem wist ik niet wat minimax was en wat een search tree was en hoe het zou moeten werken en hoe het geprogrammeerd moest worden. Om dit om te lossen was er een makkelijke oplossing onderzoek doen op het internet.

Als tweede probleem was wat is mijn minimaal leverbaar product? Dat was een algoritme maken met een simpel Gui met speler interactiviteit. Dat is gelukt en heb nog doorgewerkt om een advance GUI te maken met behulp van Tkinter.

Welke algoritme heb ik gekozen?

Voor dit project heb ik een algoritme gekozen dat gebruikt maakt van minimax en een search tree. Minimax is makkelijke algoritme die vaak wordt gebruikt in games die zero-sum zijn. Dit betekent dat een game maar 3 uitkomsten heeft 1-gewonnen 2-veloren en 3-gelijk gespeeld. Omdat er maar 3 uitkomsten zijn kan je heel makkelijk van de 3 uitkomsten je volgende move bepalen. Dit wordt gedaan door twee spelers tegen elkaar te laten spelen mini en max. Mini probeert in zo min mogelijk stappen te winnen terwijl max zoveel stappen mogelijk probeert te zetten.

De search tree algoritme wordt gebruikt met behulp van de minimax. De search tree heeft een bepaalde depth dat je kan instellen de depth is hoeveelheid de tree in vooruit gaat kijken Als je dit samen stopt met de minimax algoritme kan de algoritme kiezen welke de beste stappen zijn om te zetten om de grootse kans op een win te krijgen.

Minimax en de search tree worden samengevoegd tot 1 algoritme die connect four kan spellen te andere mensen en zichzelf.

Waarom heb ik dit algoritme gekozen en niet een andere?

Ik had twee keuzes om uit te kiezen voor het maken van de connect four algoritme q-learning en search tree/minimax, ik begon met kijken naar q-learning want dat was de eerste algoritme die ik tegenkwam na wat onderzoek en een gesprek met mijn begeleider was het toch niet verstandig om q-learning aan te pakken omdat het veel risico zou bevatten. Hier zou een grootte risico bijzitten zoals dat ik de algoritme niet op tijd af zou hebben en daardoor geen product zou kunnen leveren. Hierdoor ben ik verder opzoek gegaan naar andere algoritmes die wel op mijn skills level liggen. Toen kwam ik dit [artikel](http://www.u.arizona.edu/~wkerr/pubs/tree.pdf) tegen die het over minimax/ search tree had. Met dit methode zou ik ook aan algoritme kunnen maken voor conenct four. Na wat onderzoek ben ik tot de conclusie gekomen dan dit algoritme een goede fit was voor mij en mijn skills level. Van dit algoritme waren veel andere voorbeelden met connect four hierdoor zou ik makkelijker oplossing vinden voor mijn problemen tijden het programmeren van dit algoritme.

Voor- en Nadelen

Voordelen:

Dit algoritme is simpel en heeft niet een te hoge moeilijkheid. Hierdoor kunnen veel mensen het gebruiken en zijn aanpassing op de algoritme goed mogelijk. Zoals de diepte dat het inkijkt en de score dat bepaalde scenario’s op de board waard zijn.

Met dit algoritme kan je makkelijk data meekrijgen door de output te koppelen aan een json file. Hierdoor kan je meer inzicht krijgen hoe goed de algoritme werkt.

Nadelen:

Er zijn maar een paar nadelen aan dit algoritme maar deze nadelen kunnen wel een grootte impact hebben met het runnen ervan. Zoals dat hoe groter de board wordt des te langer het duurt om alle mogelijkheden te zoeken. Hier krijgt u verder meer info over mij de stuk analyticus. Niet alleen bij het vergroten van de board krijg je snelheid problemen je krijgt dat ook met de diepte van de algoritme. Des te diepte je gaat des te langer de duur, hier is meer info over te krijgen van de stuk analyticus.

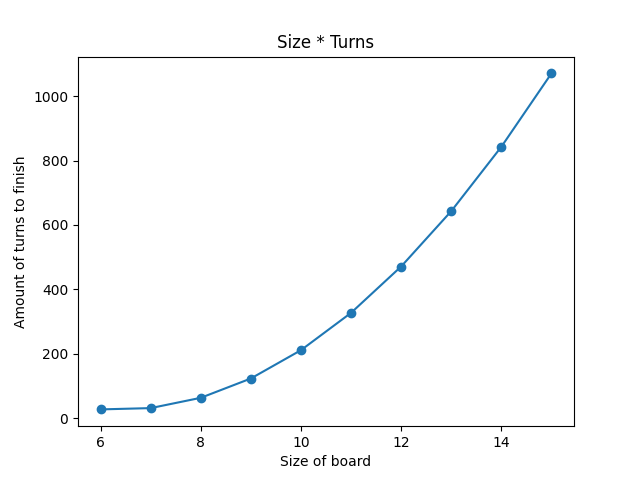
En als laatste nadeel maar wel de grootste, Als de algoritme meerde win scenario’s krijgt kiest het niet voor de snelste (minste stappen). Dit is een groot probleem want zo kan de algoritme mogelijk een spel verliezen. Dit was ik tegengekomen bij het testen van mijn algoritme (persoon tegen AI). De Ai had wel elke keer gewonnen maar niet op de snelste manier.

Etnische keuze

Bij dit algoritme hangt de etnische keuze niet sterk mee. Dat komt omdat dit algoritme geen info opslaat en ook geen moraal gebied over treedt, zoals mensen manipuleren. Om dit reden heb ik ook tijden het onderzoek naar een algoritme en tijden het maken van dit algoritme geen rekening gehouden met etherische beslissingen erachter.

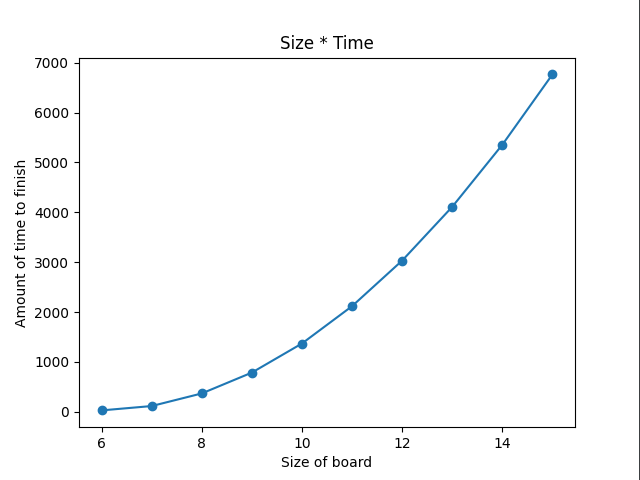
De algoritme houdt geen data, manipuleert niemand en denkt zelf niet na. Hierdoor hoef je bij dit algoritme geen rekening te houden met de etnische keuze.

Analytics



Figuur 1

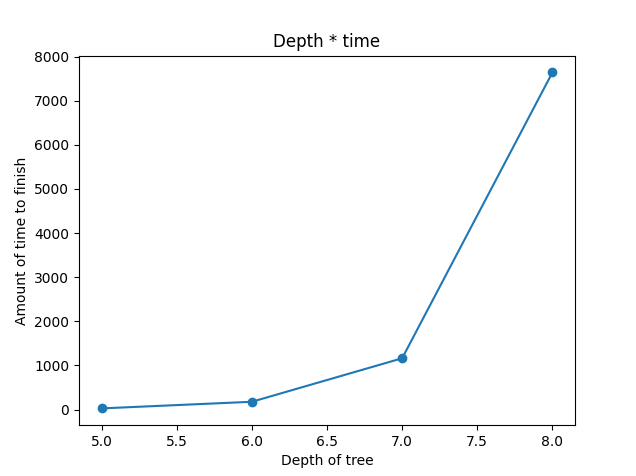
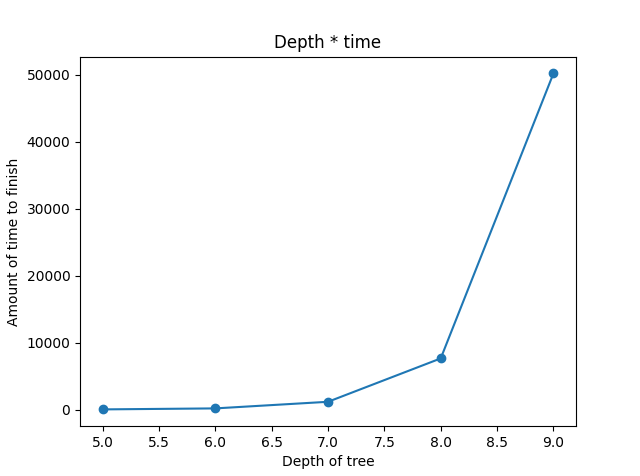
Figuur 1 geeft weer de correlatie tussen de grootte van de board en de hoeveelheid turns dat er totaal afgelegd moet worden om het spel te eindigen. Tijdens dit analyse werd er gekeken tussen twee AI’s een ervan wat depth 5 de ander was depth 3. Met dit grafiek kunnen we zien dat met de board grootte gaat de hoeveelheid turns exponentieel mee naar boven. Door dit ontdekking kunnen we concluderen des te groter de board is des te meer turn de AI’s nodig hebben om het spel afmaken.



Figuur 2 geeft weer de correlatie tussen de grootte van de board en de totale tijd voordat het spel is afgelopen. De tijd is in seconden weergegeven. Net zoals de grafiek hierboven groeit het exponentieel met de board grootte.

Hierdoor kunnen we concluderen des te grote de board gaat de tijd er exponentieel mee omhoog

Figuur 2

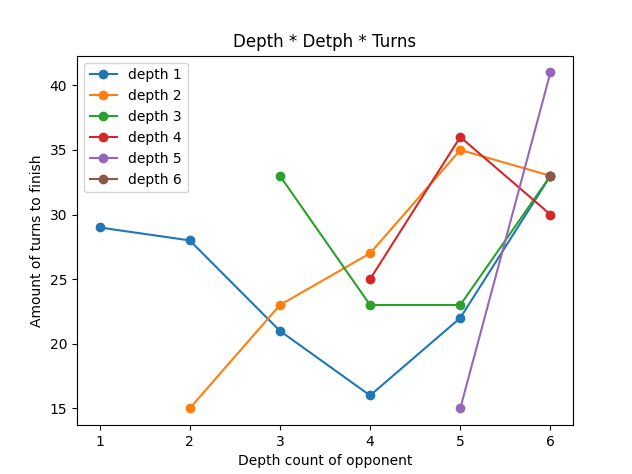


Figuur 3

Figuur 4

Figuur 3 - 5 geeft weer de correlatie tussen de depth van een AI en hoeveelheid tijd voordat het spel is afgelopen. Omdat tijden het analyseren hebben we alleen de Ai van depth 1 gebruikt om ertegen te spellen kunnen we de tijd van de tweede persoon negeren omdat het van toepassing is bij elke andere potje.

Als we gaan kijken naar de grafiek kunnen we zien dat de hoeveelheid tijd er hard stijgt met de depth van de search tree. Hierdoor kan het moeilijk in een grafiek weergegeven worden. Maar we kunnen er wel uithalen dat het tijd exponentieel groeit met de depth van de search tree.



Figuur 5

Figuur 5 geeft de correlatie tussen de hoeveelheid turn van een depth tegen een ander depth. Hier heb ik een belangrijke ontdekking gevonden. En dat is dat een groter depth niet een sneller (minste hoeveelheid turns) win is. Dat kan je zien als je kijk naar depth 1 lijn die tegen een depth 4 speelt daar zie je dat het tussen de 15 en 20 turns heeft gekost. Maar als je vervolgend dieper gaat dan wordt de hoeveelheid turns tot de finish verhoogd.

Dit uitkomst is een verrassing maar hoort bij de AI die ik heb gemaakt. De reden waarom de turns omhooggaan is omdat de AI dieper kijkt waardoor die meerder eind scenario’s te zien krijgt. De manier hoe de AI is gemaakt gaat het niet meteen naar de eerste en dus snelste eind move. Voor dit reden is het zo dat dieper niet per se beter is. De juiste dieper hangt af van de tegen stander en hoe slim die is. Als je gaat kijken hoe slim een persoon ongeveer is zou je dat kunnen inschatten tussen een depth 1 of 2, en dus de ideaalste depth is 4 of 5 hierdoor krijg je de snelste end game scenario.

Bronvermelding

q-learning

<https://en.wikipedia.org/wiki/Q-learning>

<https://web.stanford.edu/class/aa228/reports/2019/final106.pdf>

<https://towardsdatascience.com/a-beginners-guide-to-q-learning-c3e2a30a653c>

<https://towardsdatascience.com/simple-reinforcement-learning-q-learning-fcddc4b6fe56>

MiniMax/Search tree

<http://www.u.arizona.edu/~wkerr/pubs/tree.pdf>

<https://towardsdatascience.com/creating-the-perfect-connect-four-ai-bot-c165115557b0>

<https://medium.com/analytics-vidhya/artificial-intelligence-at-play-connect-four-minimax-algorithm-explained-3b5fc32e4a4f>

<https://www.youtube.com/watch?v=f5dU3xoE6ms>

<https://www.youtube.com/watch?v=trKjYdBASyQ>

<https://stackoverflow.com/>

https://www.w3schools.com/