Universiteit Leiden
The Netherlands

Bachelor Informatica

Hartenjagen met Monte Carlo Tree Search

Rutger Kiele
s2979128

Begeleider:
Aske Plaat

Tweede lezer:
Koen Ponse

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)
www.liacs.leidenuniv.nl

Augustus 2024

Abstract

Kunstmatige intelligentie wordt steeds meer geavanceerd en op veel verschillende manieren gebruikt. In deze scriptie wordt het kaartspel hartenjagen onderzocht; Dit spel wordt gekenmerkt door imperfecte informatie. MCTS is een veelgebruikt algoritme waar verschillende mogelijkheden zijn om met imperfecte informatie om te gaan. Een van die manieren is determinisatie, een strategie die wordt onderzocht in deze scriptie. De uitkomst is dat determinisatie werkt en net zo goed of misschien zelfs wel beter speelt dan een agent met perfecte informatie.

Contents

Deze bachelor thesis is geschreven als deel van de bachelor Informatica aan de Leiden Institute of Advanced Computer Science (ofwel LIACS) en wordt overzien door Aske Plaat.

1 Introductie

De velden van Informatica en Artificiële Intelligentie (AI) worden door de jaren heen steeds geavanceerder. Het begint een steeds grotere rol te spelen op verschillende gebieden van de maatschappij, van de zorg tot in het verkeer. Het gebied waar deze scriptie zich op focust is het gebied van kaartspellen. Veel (kaart)spellen worden tegenwoordig gebruikt om algoritmes te onderzoeken. Voorbeelden van dit soort spellen zijn schaken, Poker of het Chinese bordspel Go. Een software agent is een programma dat op basis van een algoritme een bepaalde taak uitvoert en in het geval van deze scriptie een spel speelt [Ter23]. Voor elk van de hiervoor genoemde spellen zijn software agenten gemaakt die het spel beter kunnen spelen dan de beste menselijke spelers dat kunnen. Voor schaken is dit Stockfish [Sto], een variant van het Minimax-algoritme. Voor poker is er Pluribus, de eerste agent die professionele poker spelers kan verslaan in een complex toernooi met meerdere spelers [Sol19] en voor Go bestaat bijvoorbeeld AlphaGoZero [SHSH18].

Een spel dat minder aandacht krijgt is het kaartspel Hartenjagen. Hartenjagen kent vele varianten en deze scriptie is gebaseerd op een versie van de Amerikaanse variant [Wik]. In tegenstelling tot de spellen die hiervoor zijn genoemd is Hartenjagen nog niet "opgelost" aangezien het een stuk minder aandacht dan de rest. Dit betekent dat het beste programma voor Hartenjagen nog niet beter is dan de beste menselijke speler [SW06]. Dit programma maakt gebruik van meerdere technieken waaronder een algoritme genaamd Monte Carlo Tree Search (MCTS). Dit is een algoritme dat gebruikt maakt van simulaties. Het best verkregen resultaat uit deze simulaties bepaalt de beste zet. MCTS is oorspronkelijk bedoeld voor situaties met perfecte informatie. Dit houdt in dat het algoritme over alle informatie van het spel beschikt. In het kader van Hartenjagen betekent dit dat de agent de handen van alle tegenstanders kent. Hartenjagen is een voorbeeld van een spel met imperfecte informatie; de spelers weten niet welke hand de tegenstanders hebben. Dit maakt de simulaties voor een MCTS agent een stuk ingewikkelder aangezien je niet weet welke kaarten de tegenstanders in de toekomst zou kunnen spelen. Een mogelijke manier om hiermee om te gaan is determinisatie. Dit is een manier om met situaties om te gaan die imperfecte informatie bevatten. Meer hierover wordt uitgelegd in sectie 2.2.

Een variant van MCTS met perfecte informatie is al eerder succesvol geïmplementeerd voor Hartenjagen door Nathan Sturtevant en Adam White in 2008 [Stu08]. Hierom ligt de focus voor deze scriptie vooral op het maken van een MCTS agent met imperfecte informatie. Dit is al eerder geprobeerd door Joris Teunisse die in 2017 voor zijn scriptie een vergelijkbaar onderzoek deed [Teu17] en waarbij hij met twee deterministische agenten ongeveer dertig procent van de spellen won tegen twee agenten die perfecte informatie hadden. Deze scriptie streeft ernaar om dit getal te verbeteren.

Verder is er een regel bij Hartenjagen die inhoudt dat wanneer een speler na dertien handen alle punten (26 in deze variant) heeft gekregen, wat normaal gesproken erg slecht is, deze speler nul punten krijgt en al zijn tegenstanders krijgen er 26. Het doel is daarom om goed te spelen, maar niet te goed. Hier moet een strategie op worden bedacht om dit voor de tegenstanders van een MCTS agent tegen te gaan. Er wordt in deze scriptie ook onderzocht of het tactisch is om zelf voor alle punten te gaan.

1.1 Probleemstelling

De probleemstelling van deze scriptie is:
In hoeverre is het mogelijk om een Monte Carlo Tree Search algoritme te maken dat het kaartspel Hartenjagen zo goed mogelijk speelt?
Dit wordt opgedeeld in de volgende subvragen:

1.1.1 Subvraag 1

In hoeverre kan determinisatie gebruikt worden in Monte Carlo Tree Search om het gebrek aan perfecte informatie tegen te gaan?

1.1.2 Subvraag 2

Is het in Hartenjagen een goede strategie om voor alle punten te gaan?

1.2 Methode

Om de probleemstelling met de subvragen te beantwoorden wordt er eerst een versie van het spel Hartenjagen gecodeerd. Dit wordt gedaan in de programmeertaal C++ en in de code-editor Visual Studio Code. Hierna wordt er een random speler aan toegevoegd om de hierna gemaakte algoritmes mee te vergelijken. Twee versies van een Monte Carlo Tree Search algoritme worden geïmplementeerd: één met perfecte informatie en één die determinisatie gebruikt.

1.3 Leeswijzer

De rest van deze scriptie is als volgt georganiseerd: In sectie 2 wordt er gerelateerd werk besproken en een uitleg over het spel Hartenjagen gegeven. In sectie 3 wordt de structuur van het geschreven programma besproken. In sectie 4 worden de experimenten besproken die worden gebruikt om een antwoord op de probleemstelling te vinden. In sectie 5 worden de resultaten getoond en geanalyseerd en als laatste bevat sectie 6 de conclusie met eventuele obstakels van dit project en mogelijke vervolg onderzoeken.

2 Gerelateerd Werk en Achtergrond

In sectie 2.1 wordt het spel Hartenjagen met alle regels uitgelegd. In sectie 2.2 wordt er uitleg gegeven over Monte Carlo Tree Search en determinisatie.

2.1 Hartenjagen

Zoals in de Introductie (sectie 1) al benoemd focust deze scriptie zich op een versie van de Amerikaanse variant van Hartenjagen [Rul]. Na de algemene uitleg wordt er benoemd hoe deze variant afwijkt van de Amerikaanse variant.

Hartenjagen wordt traditioneel gespeeld door vier spelers die allemaal proberen zo min mogelijk punten te halen. Een standaard kaartspel bestaande uit 52 kaarten wordt eerlijk opgedeeld aan de

vier spelers zodat elke speler dertien kaarten heeft. Hierna worden dertien slagen gespeeld waarbij de speler die begint een kaart naar keuze mag opleggen en de rest van de spelers de "kleur" van die kaart moeten volgen. In een kaartspel betekent de kleur van de kaart één van de vier symbolen: ♣ voor klaver, ♢ voor ruiten, ♠ voor schoppen en ♡ voor harten. De speler die begint met de klaver 2 in zijn hand begint en de rest van de spelers moet dan ook een klaver neerleggen. Een voorbeeld van een slag is te zien in afbeelding 1.

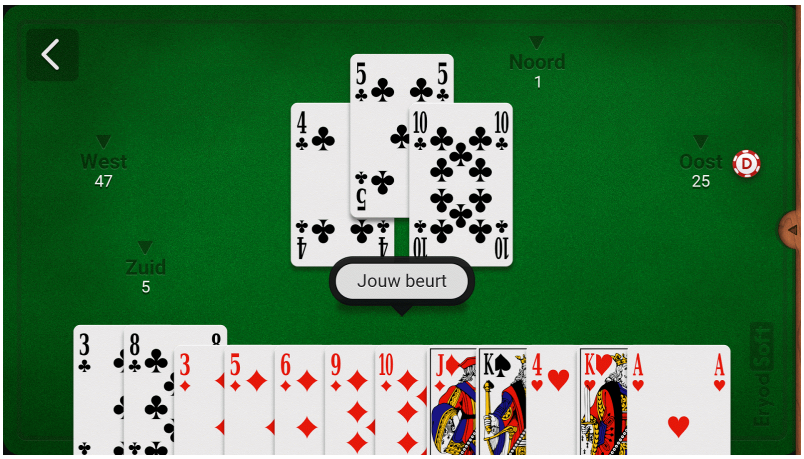

Figure 1: Een voorbeeld van een slag in Hartenjagen. Speler West heeft hier als eerste de klaver 4 gespeeld en de rest van de spelers moet daarna ook een klaver spelen als ze die hebben.

De winnaar van die zogeheten slag wordt bepaald door welke speler de hoogste kaart van de als eerst opgegooide kleur heeft gespeeld. De rang volgorde van de kaarten is (van hoog naar laag): aas, heer, vrouw, boer, 10, 9, 8, 7, 6, 5, 4, 3, 2. Wanneer een speler de kleur die als eerst is gespeeld niet heeft dan mag die speler een andere kaart naar keuze spelen en kan de slag niet meer winnen. Dit is waar de puntentelling van pas komt. Bij Hartenjagen is elke kaart van de kleur harten één punt waard, de schoppen vrouw is dertien punten waard en de rest van de kaarten is nul punten waard. Zoals al eerder gezegd is het doel van het spel om zo min mogelijk punten te halen. Dit betekent ook dat het doel is om andere spelers zo veel mogelijk punten te geven. Wanneer een speler de opgegooide kleur niet heeft kan die er voor kiezen om een harten of de schoppen vrouw te spelen (als de speler die in zijn hand heeft). Dit betekent dat er nu punten in de slag komen te zitten. De winnaar van die slag krijgt deze punten. Wanneer er voor het eerst punten in een slag zitten, is het vanaf die slag toegestaan om als eerste kaart van een slag harten op te gooien. Dit was hiervoor nog niet toegestaan en kan dan een goede strategie zijn om in een keer een andere speler vier punten te laten krijgen. Wanneer er dertien slagen zijn gespeeld is de hand van alle spelers leeg en worden de punten geteld. De punten van deze dertien handen worden bij de totaal score van de spelers opgeteld en de kaarten worden opnieuw verdeeld. Het spel is klaar wanneer de eerste speler, na zo een ronde van dertien slagen te hebben gespeeld, boven of op de honderd punten eindigt. Het spel is dan afgelopen en de speler met de minste punten wint.

Een speciale regel bij hartenjagen is al benoemd in de Introductie (sectie 1). Wanneer het een speler is gelukt om na dertien slagen alle punten te krijgen, dit zijn er 26, dan krijgt deze speler bij het tellen nul punten bij zijn totale scoren geteld en alle andere spelers krijgen er 26 bij. Als dit lukt is het een erg goede manier om een stap dichterbij de overwinning te komen. Deze strategie komt echter niet zonder risico. Wanneer het net niet is gelukt om alle punten te behalen en een

speler er maar bijvoorbeeld 25 heeft. Dan worden er gewoon 25 punten bij zijn totale score opgeteld en is deze speler een stapje dichterbij het spel verliezen. Deze regel zorgt ervoor dat elke speler extra moet nadenken om te zorgen dat de andere spelers niet alle punten halen en om te evalueren wanneer het tactisch is om er misschien zelf voor te gaan.

Aan het begin van deze sectie werd benoemd dat deze versie van Hartenjagen niet precies overeenkomt met de Amerikaanse variant. Het grootste verschil is namelijk dat bij de Amerikaanse variant aan het begin van elke ronde, nadat de kaarten worden verdeeld, elke speler drie van zijn kaarten moet weggeven aan een van de andere spelers. Dit is echter niet geïmplementeerd in de versie van Hartenjagen die in deze scriptie wordt gehanteerd. De reden hiervoor is dat deze functionaliteit niets toevoegt aan de werking van de Monte Carlo Tree Search algoritmes. Het enige voordeel wat de deterministische agent kan hebben is dat die van een van de spelers drie kaarten kent. Dit "voordeel" was echter niet significant genoeg om deze regel van Hartenjagen te implementeren.

2.2 Monte Carlo Tree Search en determinisatie

In de Introductie (sectie 1) is het Monte Carlo Tree Search (MCTS) algoritme al geïntroduceerd. MCTS [Wan21] is een algoritme dat gebruik maakt om bepaalde keuzes te maken. Het wordt voornamelijk gebruikt voor twee speler bordspellen zoals bijvoorbeeld het complexe Chinese bordspel Go. Met behulp van MCTS is het bekende algoritme AlphaGoZero [SHSH18] gemaakt voor dit spel. In het kader van bordspellen wordt MCTS gebruikt om de spelboom op te lossen. Een spelboom bevat alle mogelijke toestanden waarin het spel zich kan bevinden. MCTS bekijkt alle mogelijk zetten en probeert door middel van simulaties de beste te bepalen. Dit wordt in vier stappen gedaan, zie afbeelding 2

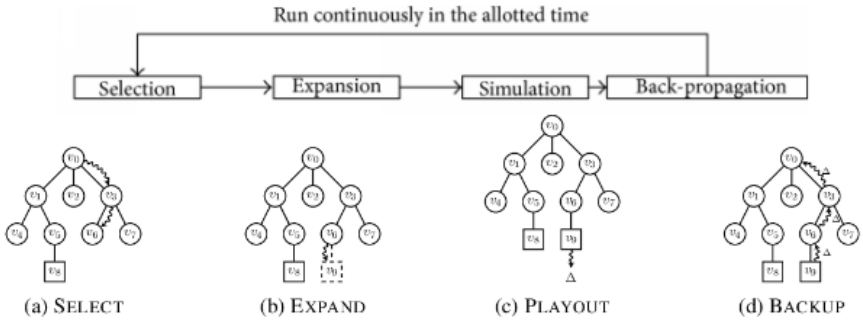

Figure 2: De vier fases van het Monte Carlo Tree Search algoritme: Selection, Expansion, Simulation en Back-propagation

De eerste stap is Selection. Hier wordt er door middel van een vooraf bepaalde policy een nog niet uitgebreide node gekozen. Een voorbeeld van zo een policy is Upper Confidence Bound (UCB) [Rob20]. Deze policy focust meer op de potentiële winst van een actie dan de gemiddelde winst en wordt bijvoorbeeld gebruikt bij AlphaGo [Has16], een ander algoritme voor Go.
In de tweede stap Expansion wordt alleen de gekozen node uitgebreid en een kind node gemaakt. De derde stap is Simulation. Vanaf de nieuwe node worden er nu een aantal keer simulaties met random policies uitgevoerd en de resultaten worden opgeslagen. Deze resultaten worden in de vierde

en laatste stap Back-propagation terug naar de originele node gestuurd. Dit blijft herhaald worden totdat alle mogelijke staten van het spel zijn bekeken of totdat het algoritme eindigt vanwege tijdslimieten. Hierna kan uit de resultaten van alle simulaties een beste zet worden gekozen.
In het kader van Hartenjagen houdt dit in dat voor elke mogelijke kaart (Selection) die de MCTS agent zou kunnen spelen: de agent die eerst "in zijn hoofd" speelt (Expansion). Dan een aantal simulaties doet van een aantal slagen of totdat het spel klaar is (Simulation). Als laatste wordt het resultaat opgeslagen (Back-propagation) en uit de resultaten van alle simulaties wordt een beste kaart gekozen die de agent dan speelt.

Zoals al hiervoor benoemd wordt Monte Carlo Tree Search vooral gebruikt bij bordspellen. Dit komt doordat bordspellen perfecte informatie hebben; de spelers weten precies de status van het spel en wat de opties zijn voor beide spelers. Een voorbeeld hiervan is schaken waar beide spelers precies weten waar elke stuk staat op het bord en welke mogelijke zetten de tegenstander zou kunnen doen. Bij de meeste kaartspellen is dit niet altijd het geval, zoals bij Hartenjagen. Dit maakt het toepassen van MCTS op Hartenjagen ingewikkeld aangezien je in de simulaties niet weet welke kaarten de andere spelers zouden kunnen spelen. Er is al wel een variant van MCTS gemaakt voor een versie van Hartenjagen die het spel met perfecte informatie speelt [Stu08]. Deze variant weet de handen van alle spelers en speelt eigenlijk vals.

Een manier om MCTS te gebruiken bij Hartenjagen zonder vals te spelen is determinisatie. Dit wordt bij MCTS gebruikt om instanties van imperfecte informatie om te zetten naar situaties met perfecte informatie waar MCTS wel op uitgevoerd kan worden. Determinisatie doet dit door een steekproef uit alle mogelijke instanties van imperfecte informatie te pakken en die met perfecte informatie MCTS op te lossen. In het kader van Hartenjagen betekent dit dat het algoritme uit alle mogelijke verdelingen van handen die de andere spelers kunnen hebben een aantal verdelingen kiest. Vervolgens wordt MCTS op de gekozen verdelingen uitgevoerd. De gemiddelde beste zet over al deze simulaties van de verschillende verdelingen van de handen wordt dan uiteindelijk gekozen als de beste zet. Om deze techniek goed te laten werken, het liefst net zo goed als perfecte informatie MCTS, gebruikt dit algoritme eerder verkregen informatie uit het spel om een betere inschatting te maken op de verdeling van de handen. Eerst wordt er gekeken naar de kaarten die de speler zelf in zijn hand heeft en degene die al gespeeld zijn; deze hoeven natuurlijk geen deel uit te maken van de verdeling. Daarna slaat deze agent informatie op over de verschillende kleuren die de andere spelers wel of niet kunnen hebben. Als een speler bijvoorbeeld niet een kaart met de kleur speelt die als eerste in de slag werd gespeeld, dan heeft deze speler die kleur niet meer en kan dit ook opgenomen worden in de verdeling.

Het combineren van determinisatie met Monte Carlo Tree Search is al eerder geprobeerd. Een voorbeeld hiervan is voor het chinese kaartspel Dou Di Zhu [WPC11]. Er zijn ook al twee eerdere studenten die MCTS met determinisatie hebben geprobeerd toe te passen op Hartenjagen. Een scriptie van Joris Teunisse uit 2017 [Teu17] en artikel door Freek Bax [Bax20].

2.3 Strategieën voor Hartenjagen

Net zoals in andere kaartspellen zijn er vele strategieën waarmee je een spel Hartenjagen zou kunnen winnen. De meest belangrijke is het bijhouden van de gespeelde kaarten, dit heet ook wel het tellen van kaarten. Weten welke spelers een bepaalde kleur niet meer hebben kan enorm helpen met het ontvangen van minder punten. Het bijhouden van gespeelde kaarten en mogelijke handen van

andere spelers is ook precies wat de Monte Carlo Tree Search agent met determinisatie doet om de simulaties zo accuraat mogelijk uit te voeren. Een van de meest gevaarlijke maar ook de potentieel meest winstgevende strategieën is voor alle punten gaan. Een artikel op de site Arkadium zegt het volgende: "Shooting for the moon is the best result in the game" [Ark]. Dit artikel benoemt verschillende methodes om Hartenjagen te winnen, maar de meest belangrijke is waarschijnlijk: "Be Mindful of Where the Hearts Are". Het artikel benoemd ook dat het voor nieuwe spelers misschien niet de beste strategie is om voor alle punten te gaan, maar zorgen dat andere spelers dit niet halen is cruciaal om het spel te kunnen winnen. Dit is ook waarom het artikel benoemd dat bij spellen met meer ervaren spelers er vaak gezien kan worden dat iedereen vecht voor het eerste punt zodat de optie om voor alle punten te gaan niet meer bestaat.

Subvraag 1.1.2 van deze scriptie bekijkt of het een goede strategie kan zijn om voor alle punten te gaan. Hoe lastig het ook kan zijn om dit resultaat te behalen, er zijn natuurlijk dingen waarop gelet kan worden. Een artikel van Seth Brown [Bro18] bespreekt er een aantal. Ten eerste is het erg belangrijk dat de omstandigheden goed zijn om voor alle punten te gaan. Er is namelijk een significante kans dat het niet lukt waardoor er opgelet moet worden dat de speler die ervoor gaat het kan veroorloven om bijvoorbeeld 25 punten erbij te krijgen. Ten tweede moeten de verkregen kaarten ook zo goed als perfect zijn; het hebben van veel hoge harten en op zijn minst de schoppen vrouw is een vereiste. Als er eenmaal is besloten om voor deze strategie te spelen is het erg belangrijk om alle hogen kaarten die de speler zelf niet heeft bij te houden. Deze informatie kan later in het spel gebruikt worden om heel wat slagen te winnen. Een andere valide strategie is om niet gelijk te laten weten wanneer je voor alle punten gaat. Dit kan door eerst een aantal kleine punten binnen te halen en later controle over het spel te nemen met de betere kaarten, wat zorgt voor een grotere kans op succes.

3 Programma architectuur

Om een antwoord te krijgen op de probleemstelling en subvragen van deze scriptie is er een programma geschreven in C++. Deze code is te vinden op GitHub via deze link: `https://github.com/RutgerKiele/Hearts-Thesis`. De code-editor die is gebruikt is Visual Studio Code. De code, waaronder alle classes, functies en namen van de variabelen, zijn in het Engels om het voor de meeste mensen begrijpbaar te maken. Als eerst is de versie van Hartenjagen beschreven in sectie 2.1 geïmplementeerd. Het meest belangrijke deel van elk kaartspel zijn de kaarten zelf. Dit is waar de C++ class Card voor is; deze zorgt ervoor dat elke kaart een waarde, kleur en een aantal punten (1 per harten en 13 voor de schoppen vrouw) bevat. Daarnaast is er een Deck class die op initialisatie een standaard deck aanmaakt met 52 kaarten en de punten die gebruikt worden bij Hartenjagen aan de goede kaarten toekent. Om de verschillende spelers te kunnen implementeren is er een basis class genaamd Player die functionaliteiten bevat die elke speler nodig heeft. Deze functionaliteiten bestaan onder andere uit het aanpassen van de hand van de speler, het toevoegen van punten, functies om te kijken of de speler een bepaalde kleur wel heeft en een functie om alle mogelijke kaarten te vinden die op dat moment gespeeld mogen worden. Deze zogeheten "parent class" heeft ook een virtuele functie genaamd playCard die elke verschillende speler zelf moet implementeren. Elke variant speler bepaald namelijk op hun eigen manier welke kaart ze gaan spelen.

Deze player class heeft vier "child classes". Dit zijn classes die al de hiervoor genoemde functionaliteiten overnemen en mogelijk zelf nieuwe functies toevoegen. Deze vier classes zijn de

RandomPlayer, MonteCarloPlayerPI (perfecte informatie), de MonteCarloPlayerDet (met determinisatie) en de ManualPlayer. De structuur van deze classes zijn te zien in afbeelding 3.

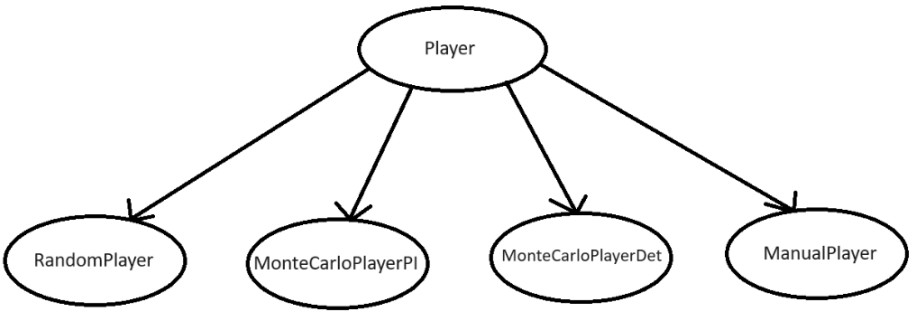

Figure 3: Een boom die de hiërarchie van de speler classes voor stelt. Player is de parent class en de vier sub classes daaronder zijn de child classes.

De RandomPlayer class is de simpelste en bestaat alleen uit de overgenomen playCard functie waarbij er een random kaart uit de speelbare kaarten wordt gepakt en wordt gespeeld. Deze speler kan gebruikt worden om te testen of de Monte Carlo spelers wel goed werken. De MonteCarloPlayerPI class is de perfecte informatie versie van het Monte Carlo Tree Search algoritme. Deze speler krijgt elke keer voordat die aan de beurt is de handen van de tegenstanders te weten. Daarnaast heeft hij ook weer een implementatie van playCard waarbij voor elke mogelijke zet een aantal simulaties worden uitgevoerd waarbij elke speler random mogelijk kaarten speelt. De resultaten van deze simulaties worden opgeslagen in een array en de kaart met het beste gemiddelde resultaat wordt gespeeld en uit de hand van de speler verwijderd. De derde child class van player is MonteCarloPlayerDet. Deze speler werkt praktisch hetzelfde als de MonteCarloPlayerPI maar in plaats van een kopie te maken van de spelers voor de simulaties, roept hij de functie generatePossibleHands aan. Deze functie genereert een aantal mogelijke handen voor elke tegenstander. In de simulaties worden nieuwe spelers aangemaakt die de handen krijgen uit de functie. Dit gebeurt een aantal keer zodat er verschillende handen zijn bekeken door de simulaties. De ManualPlayer class is gemaakt om zelf het spel te kunnen spelen tegen de verschillende agenten. De class van deze speler bestaat vooral uit vele cout statements die teksten naar de terminal sturen wanneer het programma wordt uitgevoerd. Op deze manier is het voor de manual speler ook duidelijk wie welke kaart heeft gespeeld en wat zijn opties zijn.

Als laatst zijn er twee classes om het spel uit op te bouwen. De eerste is de class Trick, wat Engels is voor een slag. Deze class wordt aangemaakt elke keer als er een nieuwe slag wordt gespeeld en bestaat uit onder andere de gespeelde kaarten met wie ze hebben gespeeld en de kleur van de eerst gespeelde kaart. De trick class heeft ook functionaliteiten om de winnaar van een slag te bepalen en de punten aan deze winnaar toe te kennen. De tweede class heet Round. Deze stelt een ronde van het spel voor waarin er aan elke speler dertien kaarten worden gegeven en de dertien handen worden gespeeld. Hierna worden de punten bekeken en er wordt gekeken of iemand alle punten heeft gehaald. Als laatst voegt deze class de verkregen punten toe aan de totaal score van de spelers.

Al deze classes worden samengebracht in de main.cc file. Wanneer het programma wordt uitgevoerd

wordt als eerst de main functie aangeroepen. Hierin begint het programma met het vragen met hoeveel spelers er gespeeld gaat worden, er zitten namelijk ook functionaliteiten in dat het met een n aantal spelers kan worden gespeeld. Als dit antwoord twee of vier is dan wordt er gevraagd of de persoon zelf wilt spelen. Hierna worden de spelers aangemaakt en het deck geschud. Als er geen ManualPlayer is dan worden er vervolgens honderd potjes gespeeld en de resultaten worden op het scherm afgedrukt. Anders wordt er maar één spel gespeeld.

De hiërarchie van al deze classes is te zien in afbeelding 4. Main roept Deck en Round aan die vervolgens weer Card en Trick aanroepen. Main zorgt ook dat de verschillende type spelers worden aangemaakt.

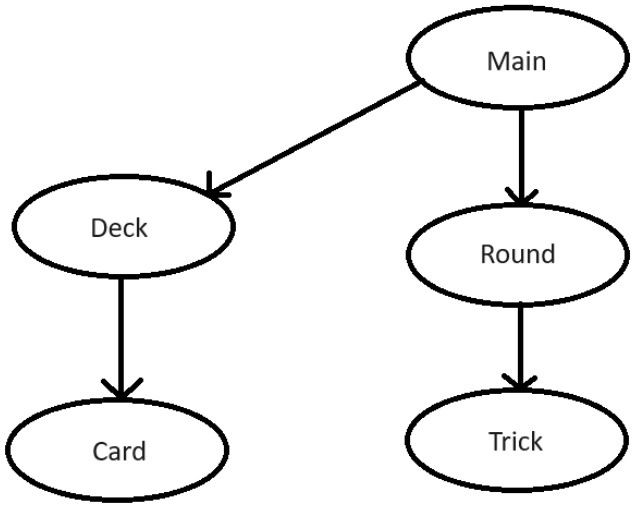

Figure 4: Twee bomen die de class hiërarchie voorstellen. Een Deck bestaat uit Cards en een Round bestaat uit Tricks. Main roept deze beide aan om het spel te spelen.

4 Experimentele opzet

Om de probleemstelling en de subvragen te kunnen beantwoorden moet er eerst gezorgd worden dat de beide algoritmes zo goed mogelijk werken. Beide algoritmes kijken in de simulaties een aantal handen vooruit. Dit aantal kan ergens tussen de 0 en 12 liggen; Bij 0 wordt er alleen naar de huidige hand gekeken en bij 12 naar alle handen tot de ronde klaar is. Om het optimale aantal handen te vinden spelen er een twee agenten met verschillende aantallen handen in de simulaties tegen dezelfde agenten die na de huidige hand nul handen vooruitkijken. Alle agenten doen per beurt dertig simulaties. Te verwachten is dat een hoger aantal handen het beter doet dat een lager aantal aangezien de agent dan verder vooruit zou kijken. Precies ditzelfde experiment wordt uitgevoerd voor de MCTS agent die determinisatie gebruikt.

Zoals eerder gezegd in sectie 2.2 gebruikt deze agent een steekproef van de verschillende mogelijke handen van de tegenstanders om zo een goede schatting van de echte handen te krijgen. De grootte van deze steekproef kan ook variëren. Bij het bepalen van het aantal gesimuleerde handen

wordt deze op drie gezet met een aantal simulaties van tien zodat het aantal totale simulaties per beurt ook weer op dertig staat. Het maakt in dit geval namelijk niet uit hoe groot de steekproeven zijn aangezien ze allemaal dezelfde grootte gebruiken. Wanneer we echter de determinisatie agent met de perfecte informatie agent gaan vergelijken moet deze grootte wel optimaal zijn. Er wordt daarom een soortgelijk experiment als de vorige uitgevoerd waar twee determinisatie agenten met verschillende grootte steekproeven tegen twee determinisatie agenten met steekproef grootte 1 worden gezet. Het aantal gesimuleerde handen wordt bij dit experiment gelijk gezet aan het optimale aantal gevonden in het vorige experiment. Om te zorgen dat het totale aantal simulaties bij beide paren van agenten hetzelfde blijft, wordt het aantal simulaties per hand uit de steekproef aangepast zodat het totale aantal weer op dertig terecht komt. Hierdoor zou de executie tijd ongeveer hetzelfde moeten blijven aangezien het totale aantal simulaties bijna niet veranderd.

Nadat al deze optimale parameters zijn gevonden kan subvraag 1 (sectie 1.1.1) beantwoord worden. Dit wordt gedaan door de twee algoritmes, de ene met perfecte informatie en de andere met determinisatie, met de gevonden parameters tegen elkaar te laten spelen. Dit wordt gedaan over een strekking van 500 spellen om de willekeurigheid van het spel te verminderen. Net zoals bij de vorige twee experimenten worden er twee van elke agent gebruikt en de scores van de twee worden bij elkaar opgeteld.

Om subvraag 2 (sectie 1.1.2 te beantwoorden is eerst een functie geprogrammeerd die bepaald of het tactisch is om voor alle punten te gaan. Deze functie is gebaseerd op strategieën die besproken zijn in sectie 2.3. Om te testen of deze functie wel echt een verschil maakt, wordt dit experiment uitgevoerd op een twee speler variant van Hartenjagen. In deze variant van het spel winnen twee identieke spelers altijd de helft van de spellen tegen elkaar (plus of min 1 procent). De functie is geïmplementeerd op de perfecte informatie versie van MCTS aangezien die sneller is en in een één tegen één ze toch identiek werken. Hierom wordt er in dit experiment een versie van MCTS met perfecte informatie die niet voor alle punten gaat opgezet tegen een versie van dat algoritme dat wel voor alle punten gaat. Als de ene versie significant meer wint dan de ander kan er gezegd worden dat voor alle punten gaan een goede strategie kan zijn.

5 Resultaten

In deze sectie worden de experimenten uitgevoerd die zijn beschreven in sectie 4 om de subvragen en daarmee ook de probleemstelling te beantwoorden.

5.1 Subvraag 1

Om subvraag 1 te kunnen beantwoorden, moeten de algoritmes eerst zo optimaal mogelijk werken. De grootste uitdaging is het kiezen van de meest optimale parameters. De eerste parameter die gekozen moet worden is het aantal handen dat de algoritmes vooruit kijken in de simulaties. Door het eerst beschreven experiment in sectie 4 uit te voeren, zijn de resultaten verkregen die in figuur 5 te vinden zijn. Deze resultaten zijn voor het Monte Carlo Tree Search algoritme met perfecte informatie.

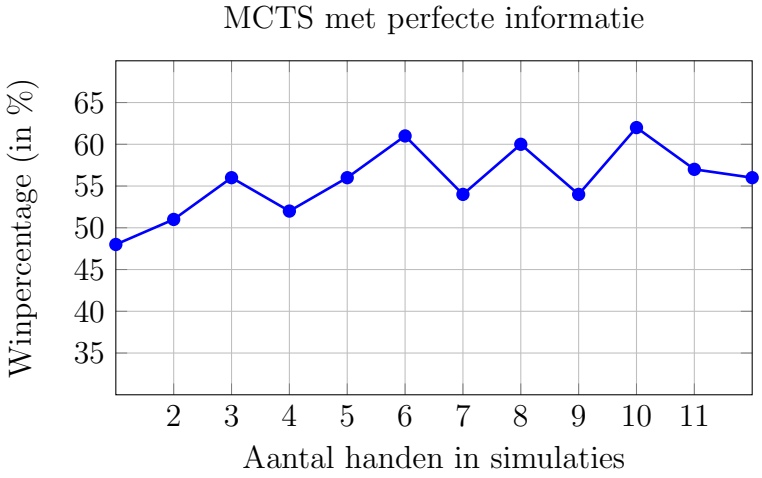

Figure 5: De win percentages van het verschillende aantal handen waarmee de simulaties van MCTS met perfecte informatie vooruit kijken. Deze percentages zijn verkregen wanneer twee agenten met het bepaalde aantal handen in de simulaties spelen tegen twee dezelfde agenten die niet vooruitkijken. Er zijn in totaal honderd spellen gespeeld en de percentages komen overeen met het aantal gewonnen spellen van het duo.

Uit de grafiek blijk dat de resultaten nogal wisselvallig zijn. Dit heeft te maken met de willekeurigheid van het spel (hoe random het kan zijn). Het is natuurlijk mogelijk dat de betere agent een slechtere hand gedeeld krijgt waardoor er alsnog een grotere kans is dat hij dat spel verliest. Uit de resultaten kan opgemaakt worden dat er wel een aanwezige stijgende lijn is wanneer er meer handen gesimuleerd worden. Deze daalt alleen weer op het einde wat indiceert dat een te hoog aantal gesimuleerde handen geen positief effect heeft op het algoritme. De percentages liggen allemaal tussen de 45 en 65 procent wat te verwachten is. De twee spelers die niet vooruitkijken spelen in de simulatie alleen de huidige hand uit terwijl de andere twee spelers er meer spelen. Dit betekent dat de twee spelers die niet vooruitkijken wel beter zijn dan de gemiddelde random speler waardoor het percentage niet hoger ligt dan die 65 procent. Wat betreft de beste keuze voor deze parameter lijkt het alsof tien de beste keuze is. Dit aantal heeft een win percentage dat net iets hoger is dan de rest, ook al scheelt het niet veel.

Om nu deze zelfde parameter te optimaliseren voor de MCTS agent met determinisatie wordt hetzelfde experiment uitgevoerd. De resultaten zijn te vinden in figuur 6.

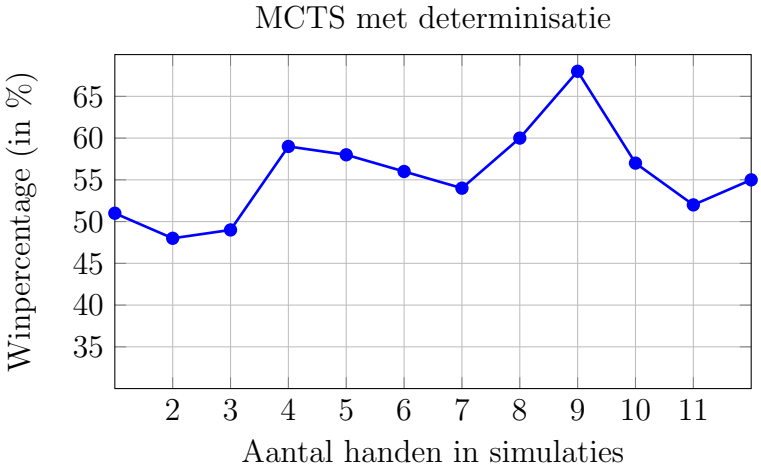

Figure 6: De win percentages van het verschillende aantal handen waarmee de simulaties van MCTS met determinisatie vooruit kijken. Deze percentages zijn verkregen wanneer twee agenten met het bepaalde aantal handen in de simulaties spelen tegen twee dezelfde agenten die niet vooruitkijken. Er zijn honderd spellen gespeeld en de percentages komen overeen met het aantal gewonnen spellen van het duo.

De resultaten uit deze grafiek lijken een stuk minder wisselvallig en lijken een duidelijk resultaat aan te tonen. Er ligt een duidelijke piek in win percentage bij acht, negen en ook tien handen met het aantal negen als de grootste uitblinker. Dit is dan ook de optimale waarde die is gekozen als parameter. Ook hier lijk het alsof een te hoog aantal gesimuleerde handen weer geen positief effect heeft op de werking van het algoritme.

De agent met determinisatie heeft nog één parameter meer om te optimaliseren die de agent met perfecte informatie niet heeft: De grootte van de steekproef die wordt genomen om de handen van de tegenstanders te benaderen. Om deze grootte te optimaliseren wordt er nog een experiment uitgevoerd dat weer is beschreven in sectie 4. De resultaten van dit experiment staan in figuur 7.

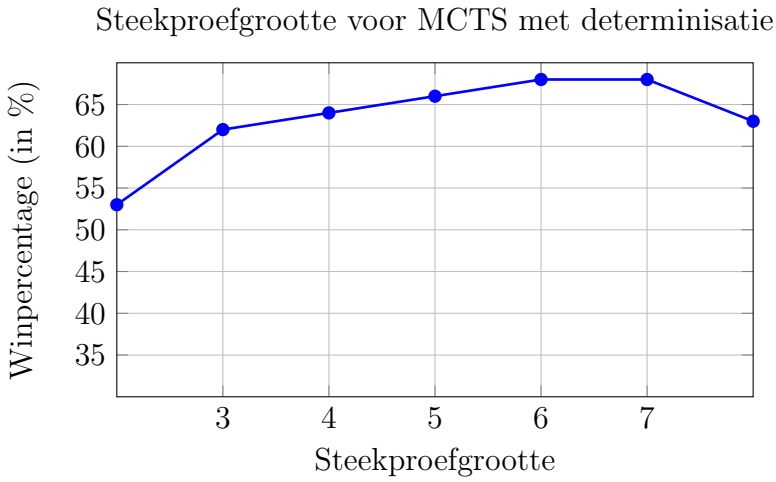

Figure 7: De win percentages van de verschillende steekproef groottes voor het MCTS algoritme met determinisatie. Deze percentages zijn verkregen wanneer twee agenten met de bepaalde grootte steekproef spelen tegen twee dezelfde agenten met een steekproef grootte van 1. Er zijn honderd spellen gespeeld en de percentages komen overeen met het aantal gewonnen spellen van het duo.

In deze grafiek is een duidelijk stijgende lijn te zien wanneer de steekproefgrootte groeit. Deze groei stopt bij een grootte van zeven wat vervolgens wordt gekozen als optimale waarde voor deze parameter. Bij een waarde van acht gaat de lijn weer omlaag. Dit heeft waarschijnlijk te maken met het feit dat het aantal simulaties per hand daalt wanneer het de grootte van de steekproef groeit. Bij een grootte van acht lijken er hier te weinig simulaties te zijn om de handen van de tegenstanders nog accuraat te bepalen.

Nu de optimale parameters van beide algoritmes bepaald zijn, kan subvraag 1 beantwoord worden. Zoals in sectie 4 beschreven wordt dit over 500 spellen gemeten. Aangezien het programma honderd spellen tegelijk uitvoert zijn de resultaten opgesplitst in vijf groepen van honderd. De tabel met de resultaten is te zien in tabel 1.

Aantal spellen gewonnen	MCTS met perfecte informatie	MCTS met determinisatie
Eerste 100 spellen	43	57
Tweede 100 spellen	45	55
Derde 100 spellen	46	54
Vierde 100 spellen	46	54
Vijfde 100 spellen	43	57
Algemeen win percentage	44.6%	55.4%

Table 1: Aantal gewonnen spellen van de duo's van de twee varianten van het Monte Carlo Tree Search algoritme. De onderste rij bevat de samengevoegde win percentages van alle spellen. Er is te zien dat MCTS met determinisatie hier in ieder geval meer spellen heeft gewonnen.

Uit de resultaten blijkt dat de MCTS agent met determinisatie het in het algemeen net iets beter doet dan de agent met perfecte informatie. Dit is opmerkelijk aangezien de determinisatie agent een schatting maakt van de handen terwijl de andere agent die handen weet. Een reden dat de determinisatie agent het net iets beter doet zou kunnen zijn dat optimale parameters voor

deze agent beter werken dan bij de perfecte informatie agent aangezien er bij figuur 5 nog wat speling was tussen de resultaten. Dit zou kunnen resulteren dat de schatting die met determinisatie wordt gemaakt beter is dan de simulaties van de perfecte informatie agent. In ieder geval kan er geconcludeerd worden dat determinisatie in Monte Carlo Tree Search gebruikt kan voor imperfecte informatie.

5.2 Subvraag 2

Voor het beantwoorden van subvraag 2 wordt het laatst beschreven experiment uit sectie 4 uitgevoerd. Net zoals bij het laatste experiment van subvraag 1, waarvan de resultaten in sectie 5.1 staan, worden er vijf keer honderd spellen uitgevoerd om de willekeurigheid tegen te gaan. De resultaten zijn te zien in tabel 2.

Aantal spellen gewonnen	Soms voor alle punten	Nooit voor alle punten
Eerste 100 spellen	55	45
Tweede 100 spellen	63	38
Derde 100 spellen	52	48
Vierde 100 spellen	54	46
Vijfde 100 spellen	60	40
Algemeen win percentage	56.8%	43.2%

Table 2: Aantal gewonnen spellen van de twee varianten van het Monte Carlo Tree Search algoritme met perfecte informatie: één die soms voor alle punten gaat en de ander die dat helemaal niet doet. De onderste kolom bevat de samengevoegde win percentages van alle spellen.

Zoals in de tabel te zien kan er gezegd worden dat de agent die soms voor alle punten gaat significant meer wint dan de agent die dit niet doet. Dit komt omdat in een twee speler spel er altijd iemand is met een betere en iemand met een slechtere hand. Wanneer de speler die voor alle punten kan gaan de slechtere hand krijgt dan is het een makkelijke keuze om voor die strategie te gaan en hij wint daardoor meer spellen.

6 Conclusie

In deze scriptie is het algoritme Monte Carlo Tree Search onderzocht met betrekking tot het kaartspel Hartenjagen. Dit algoritme gebruikt perfecte informatie om verschillende spellen op te lossen en aangezien Hartenjagen een spel is met imperfecte informatie gebruikt het algoritme determinisatie om dit tegen te gaan. Uit de experimenten is ten eerste gebleken dat het kiezen van optimale parameters cruciaal is om de algoritmes zo goed mogelijk te laten werken. Met deze parameters is een tweede experiment uitgevoerd om subvraag 1 te beantwoorden. Deze subvraag luidde:
In hoeverre kan determinisatie gebruikt worden in Monte Carlo Tree Search om het gebrek aan perfecte informatie tegen te gaan?

Uit dit experiment is gebleken dat het zeker mogelijk is om een determinisatie te gebruiken met MCTS voor spellen met imperfecte informatie. Daarnaast is er door middel van een ander experiment onderzocht of het een goede strategie kan zijn om voor alle punten te gaan. Dit is

gedaan naar aanleiding van subvraag 2:
Is het in Hartenjagen een goede strategie om voor alle punten te gaan?

Uit dit experiment is gebleken dat een speler die deze strategie kan toepassen meer spellen wint dan een speler die dit nooit doet.
Een aantal begrenzingen zijn opgetreden die ervoor hebben gezorgd dat er een aantal dingen niet onderzocht zijn in deze scriptie. Ten eerste zorgde het programmeren van Hartenjagen zelf voor een begrenzing. Alhoewel dit makkelijker was voor het implementeren van de algoritmes was dit misschien niet een optimale tijdsbesteding. Dit zorgde uiteindelijk voor minder tijd die besteed kon worden aan belangrijkere dingen, zoals het implementeren van alle regels van het spel. Ten tweede zijn de MCTS algoritmes natuurlijk niet optimaal. Er zijn altijd verbeteringen die gedaan kunnen worden maar hier niet van toepassing zijn gekomen door deze tijdslimieten. Ten derde is subvraag 2 hier alleen onderzocht met betrekking tot een twee speler spel. Dit zorgde wel voor duidelijke resultaten maar dit is helaas niet hetzelfde als het traditionele Hartenjagen.

Wat betreft verder onderzoek is er enorm veel keuze. Met betrekking tot subvraag 2 is het mogelijk om de strategie "voor alle punten gaan" te onderzoeken voor een normaal vier speler spel. De algoritmes zelf kunnen ook nog altijd verder verbeterd worden. Een potentiële optie hiervoor is de policy in de simulaties veranderen. Op dit moment worden de zetten in de simulaties willekeurig gekozen, maar door deze keuze slimmer te maken zou het de prestatie tegen slimmere agenten en spelers kunnen verbeteren. In het kader van Hartenjagen is er ook nog veel te onderzoeken. MCTS is waarschijnlijk niet het beste algoritme om dit spel op te lossen. Er kan bijvoorbeeld gekeken worden naar het implementeren van een neuraal netwerk of een compleet ander algoritme. Als het gaat om agenten voor het spel Hartenjagen ligt de wereld nog wijd open.

References

[Ark] Arkadium. https://www.arkadium.com/blog/hearts-card-game-strategy/.

[Bax20] Freek Bax. Determinization with monte carlo tree search for the card game hearts. *University of Utrecht*, 2020.

[Bro18] Seth Brown. Shoot the moon in the game hearts. *The Spruce Crafts*, 24 Oct, 2018.

[Has16] Demis Hassabis. Alphago: using machine learning to master the ancient game of go. *Google The Keyword*, Jan 27, 2016.

[Rob20] Steve Roberts. The upper confidence bound (ucb) bandit algorithm. *Towards Data Science*, Oct 26, 2020.

[Rul] Official Game Rules. https://www.officialgamerules.org/card-games/hearts.

[SHSH18] David Silver, Thomas Hubert, Julian Schrittwieser, and Demis Hassabis. Alphazero: Shedding new light on chess, shogi, and go. *Google DeepMind*, December 6, 2018.

[Sol19] Meilan Solly. This poker-playing a.i. knows when to hold 'em and when to fold 'em. *Smithsonian*, July 15, 2019.

[Sto] Stockfish. https://stockfishchess.org.

[Stu08] Nathan Sturtevant. An analysis of uct in multi-player games. *Department of Computing Science University of Alberta*, 2008.

[SW06] Nathan R. Sturtevant and Adam M. White. Feature construction for reinforcement learning in hearts. *Department of Computer Science University of Alberta*, May, 2006.

[Ter23] John Terra. Exploring intelligent agents in artificial intelligence, Aug 2023.

[Teu17] Joris Teunisse. Agents for the card game of hearts. *Leiden Institute of Advanced Computer Science (LIACS)*, 2017.

[Wan21] Benjamin Wang. Monte carlo tree search: An introduction. *Towards Data Science*, Jan 10, 2021.

[Wik] Wikipedia. https://nl.wikipedia.org/wiki/Hartenjagen.

[WPC11] Daniel Whitehouse, Edward J. Powley, and Peter I. Cowling. Determinization and information set monte carlo tree search for the card game dou di zhu. *IEEE Xplore*, October 2011.