

Vierkantjes-makende agenten

Luc Schouten
s2660148

Dylan Macquiné
s2592991

6 november 2021

1 Inleiding

Dit verslag gaat over de tweede opdracht voor het vak kunstmatige intelligentie [4]. Voor deze opdracht was het de bedoeling om twee van dezelfde soort agenten op een bord van 18×18 te plaatsen die elkaar kunnen vinden om daarna samen één vierkant te vormen.

2 Uitleg probleem

Voor de opdracht moest gebruik worden gemaakt van het programma RoboCom [3]. In dit programma kunnen agenten worden geprogrammeerd. Voordat er verder op het probleem en de constraints wordt ingegaan is het belangrijk om eerst de gebruikte terminologie in dit verslag te behandelen. Met de term “kinderen” van een agent wordt bedoeld: “Alle agenten die zijn aangemaakt door één van de agenten die initieel op het bord worden geplaatst maar ook de kinderen daarvan en zo door”. Met de term “familie” wordt bedoeld: “alle kinderen van één initiële agent en die initiële agent zelf”.

De taak-omgeving van deze agenten kan worden gespecificeerd met behulp van PEAS 3:

Performance: Op het gebied van performance moet de agent zo worden geprogrammeerd dat als er twee van dezelfde soort agenten van verschillende families op het bord worden geplaatst dat deze dan zo vaak mogelijk op coöperatieve wijze één vierkant maken. Daarnaast is het de bedoeling dat de agenten dit binnen een zo kort mogelijke tijdsperiode doen. Een andere eis is dat het ontstane vierkant moet bestaan uit vier agenten waarvan twee van de agenten kinderen zijn van één van de initieel geplaatste agent, of die agent zelf moet zijn. De andere twee agenten uit het vierkant moeten de kinderen zijn van de andere initieel geplaatste agent, of wederom die agent zelf. Daarnaast moeten de twee agenten in het vierkant die van dezelfde familie komen, aan elkaar grenzen.

Environment: De environment van een agent bestaat uit een 18×18 vakjes torus-bord. Vakjes kunnen leeg zijn maar kunnen ook agenten bevatten.

Actuatoren: Een agent heeft verschillende actuatoren waarmee deze invloed op de omgeving kan uitoefenen, zo kan een agent een nieuwe agent aanmaken en code aan deze agent meegeven of nieuwe code injecteren in een al bestaande agent tevens kan een agent de staat van een andere agent veranderen. Daarnaast kunnen de agenten zichzelf draaien, zichzelf inactief maken of zichzelf vernietigen.

Sensoren: Er zijn enkele sensoren welke de agenten kunnen gebruiken om hun omgeving waar te nemen. Agenten kunnen kijken of het vakje voor hun leeg is. In het geval dat het vakje voor de agent niet leeg is kan de agent verschillende variabelen van de agent die voor hem staat vergelijken. Zo kan de agent de actieve staat van de andere agent voor hem met andere staten vergelijken en kan de agent vergelijken of de agent die voor hem staat uit dezelfde familie als hijzelf komt.

De omgeving kan ook nog verder worden toegelicht aan de hand van de 6 kenmerken van omgevingen 3. De omgeving van de agent is deels-observeerbaar; de agent is in staat om het vakje direct voor zich te scannen en te controleren maar heeft geen toegang tot de staten van alle andere vakjes. Verder bevinden

de agenten zich in een multi-agente omgeving aangezien er meerdere agenten aanwezig zijn. Deze agenten werken op coöperatieve wijze samen om het vierkantje te maken. Wat determinisme betreft is de omgeving van de agent deterministisch, behalve de handelingen van de agenten en de huidige staat is er niks anders bepalend voor de staat. De omgeving van de agent is verder episodisch; wat de ouders of de kinderen van de agent voor handelingen uitvoeren is niet van belang voor de handeling van de huidige agent en daarom zal de huidige agent onafhankelijk van het handelen van andere agenten opereren. De omgeving van de agent binnen de opdracht is een dynamische omgeving, terwijl de agent keuzes maakt kunnen andere agenten ook invloed uitoefenen op zijn omgeving en zelfs op de huidige staat van andere agenten. Verder is de omgeving van de agent een discrete omgeving. Het aantal staten waarin de omgeving zich kan bevinden, evenals het aantal handelingen en de waarnemingen van een agent zijn namelijk eindig.

Een ander punt dat meespeelt binnen het programma is dat wanneer de agenten op het bord geplaatst worden, deze op een random plek op het bord geplaatst worden en dat de agenten een random richting opkijken.

3 Relevant werk

Een van de relevante werken voor deze opdracht is PEAS [2], Aan de hand van PEAS kan voor een agent worden omschreven wat de mate van prestaties voor die agent is, hoe de omgeving van de agent eruit ziet, hoe de agent invloed kan uitoefenen op zijn omgeving en hoe de agent zijn omgeving kan waarnemen. Een ander werk dat relevant is voor deze opdracht is “de lijst van 6 kenmerken van omgevingen” [2], aan de hand van deze lijst met kenmerken kan beter gespecificeerd worden in wat voor een omgeving een agent precies werkt. PEAS samen met de lijst van 6 kenmerken van omgevingen geven een beter beeld van de beperkingen waarbinnen een agent moet werken en van de mogelijkheden die een agent heeft.

Ander relevant werk voor deze opdracht is RoboCom [3], dit is het programma dat gebruikt wordt binnen deze opdracht om met de bijbehorende taal agenten te maken en te simuleren. RoboCom is gebouwd door Dennis C. Bemann en is geïnspireerd op CoreWar. In eerste instantie als manier om robots elkaar in het programma te laten bevechten door Assembly-achtige code te schrijven. De gebruikte Windows versie van RoboCom bestaat sinds 1998 en de code is compleet herschreven in Delphi om dit mogelijk te maken. Bij het herschrijven van de code was er extra oog voor snelheid en geheugengebruik. De graphical interface van de windows-versie is in samenwerking met Jens Rupp van Erlangen gemaakt [1].

4 Aanpak

De manier waarop de agenten werken is als volgt: ten eerste zal een agent welke op het bord geplaatst wordt bepalen wat zijn taak is. De mogelijke taken zijn; het aanmaken van kinderen om de andere familie van agenten te vinden, het vierkant produceren wat wordt gedaan door de laatste agent (**laatste-agent/lastbot**) uit de familie en het opruimen van de andere agenten uit de eigen familie (**opruij-agent/cleanbot**). Indien de agent als taak heeft om kinderen aan te maken zal de agent kijken of het vakje voor hem leeg is. Als het vakje voor de agent leeg is zal de agent daar een nieuwe agent aanmaken welke als taak meekrijgt om ook weer nieuwe kinderen aan te maken. Nadat het nieuwe kind is aangemaakt en is geactiveerd zal de agent kijken of deze ondertussen niet zelf een nieuwe taak van een andere agent heeft gekregen. Indien de agent geen nieuwe taak heeft zal deze zichzelf uitzetten. Wanneer de agent wel een nieuwe taak heeft zal zij de nieuwe taak uit gaan voeren. Indien het vakje voor de agent in eerste instantie al helemaal niet leeg was, zal de agent kijken of op het vakje voor hem familie staat. Wanneer dit het geval is zal de huidige agent zichzelf deactiveren. Wanneer op het vakje geen familie stond zal de agent nogmaals kijken of deze ondertussen niet de taak van opruij-agent is toebedeeld. Wanneer dit wel het geval is zal de agent nu de taken van opruij-agent gaan uitvoeren. Wanneer dit niet het geval is, dan is de agent die in het vakje voor de agent staat een agent van de andere familie wat inhoudt dat deze huidige agent een laatste agent is dat als doel heeft om het vierkantje te maken. Wanneer de huidige agent niet al een laatste agent was zal deze de andere agent een laatste agent aanmaken en zorgen dat die de andere kant opdraait bij het maken van het vierkantje. Als de huidige agent wel al een laatste agent was zou deze anders namelijk beginnen met zijn eigen kinderen ook laatste agenten maken, en dat is niet de bedoeling. De huidige

agent maakt zichzelf hoe dan ook een laatste-agent.

De taak van de laatste agenten is zorgen dat de rest van de eigen familie zichzelf opruimt en wachten totdat het tijd is om gezamenlijk een vierkantje te maken. De laatste agenten draaien voor een bepaalde duur om zich heen om kinderen van zichzelf te zoeken en deze opruim-agenten te maken. Wanneer deze duur om is zal de agent de goede kant opdraaien om daar een nieuw kind neer te zetten, deze te deactiveren en daarna zichzelf te deactiveren. Wanneer de twee overgebleven laatste-agenten dit allebei gedaan hebben is het vierkantje ontstaan.

De taak van de opruim-agenten is het zoeken naar andere familieleden die geen laatste-agent zijn om van die familieleden ook opruim-agenten te maken. De opruim-agenten doen dit door om zich heen te kijken totdat zij een familielid tegen komen, deze zetten ze vervolgens opruimen. Hierna zal de opruim-agent zichzelf vernietigen. Wanneer een opruim-agent na om zijn as te hebben gedraaid geen andere familieleden meer kan vinden zal deze zichzelf ook vernietigen om zo te zorgen dat wanneer een opruim-agent aan het einde van een rij familieleden zit deze niet overblijft.

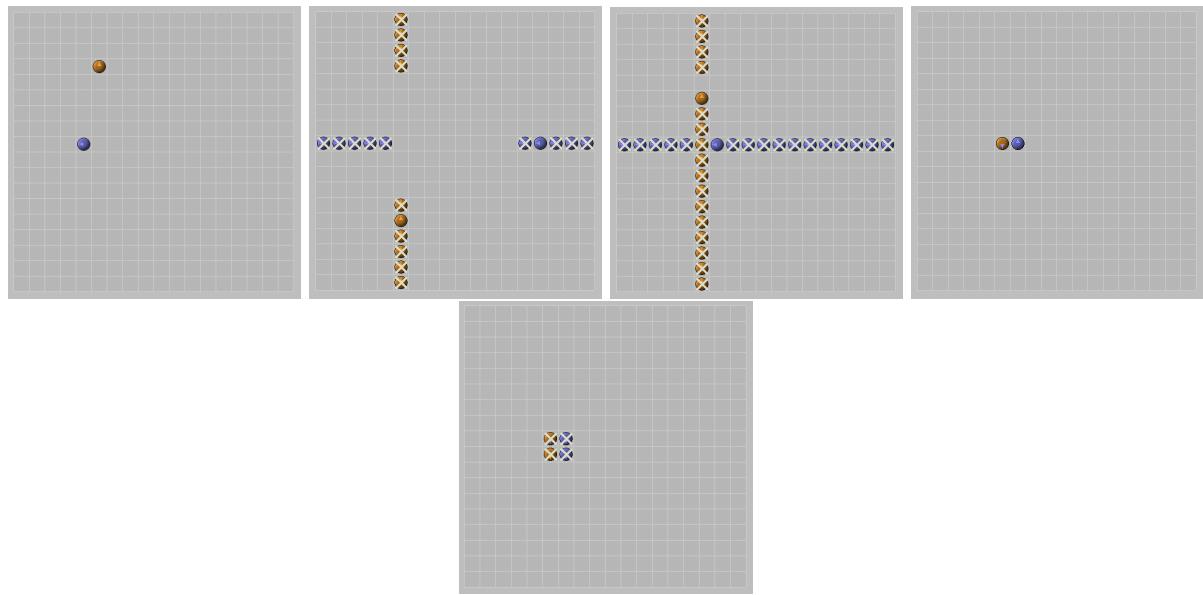
5 Implementatie

Voor de implementatie van de RoboCom-agenten is gebruik gemaakt van de bijbehorende RC300 taal. Bij de implementatie is gebruikgemaakt van één bank welke naar alle kinderen werd overgedragen. De bank is dan weer op te delen in drie stukken code welke de verschillende taken van de agenten beslaan. Vanuit het stukje code voor kinderen aan makende agenten kan een agent terechtkomen in de code voor een laatste-agent of de code voor een opruim-agent, waarna de agenten binnen de delen van de code voor de respectievelijke taken blijven en niet meer terug kunnen naar de complete code.

6 Experimenten

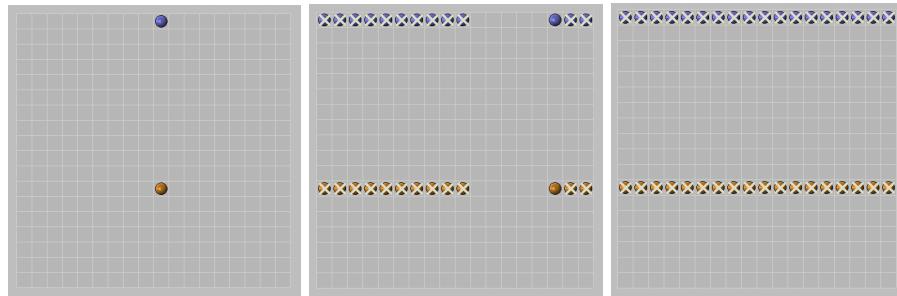
Om de prestaties van de gemaakte agenten te testen zijn op het bord $50 \times$ de twee familie verschillende agenten geplaatst en is gekeken naar het percentage van de gevallen waarin één vierkant werd gemaakt en de gemiddelde tijd die het duurde om een vierkant te maken wanneer een vierkant gemaakt werd.

Uit de resultaten van de experimenten blijkt dat van de 50 runs er 21 keer één vierkant kon worden gemaakt. Hierin zijn ook vierkanten gemaakt die aan de zijkant van het bord werden gemaakt en daardoor aan de andere kant van het bord weer uitkwamen. Van de 21 vierkanten zijn er twee waarbij dit het geval was. Het percentage van de keren dat één vierkant kan worden gemaakt bedraagt zo 42%. De gemiddelde tijd om een vierkant te maken bedroeg afgerond op duizendtallen 13000 RoboCom tijdseenheden. De tijden zijn tussentijds afgerond op duizendtallen omdat het op de stop knop klikken om de tijd vast te leggen niet heel precies is. Over de maximale tijd die binnen de experimenten nodig was om een vierkant te maken kan nog een kanttekening gemaakt worden aangezien het in ieder geval niet langer dan het op de stop knop drukken duurt om het vierkant te maken. De maximaal gedocumenteerde tijd om een vierkant te maken bedroeg 14346. De minimale tijd die het duurde om een vierkant te maken binnen de experimenten was 10818, echter zou dit in werkelijkheid nog iets lager kunnen liggen door de tijd die nodig was om op de stop-knop te drukken. De stappen van het maken van een vierkant zijn weergegeven in Figuur 1.



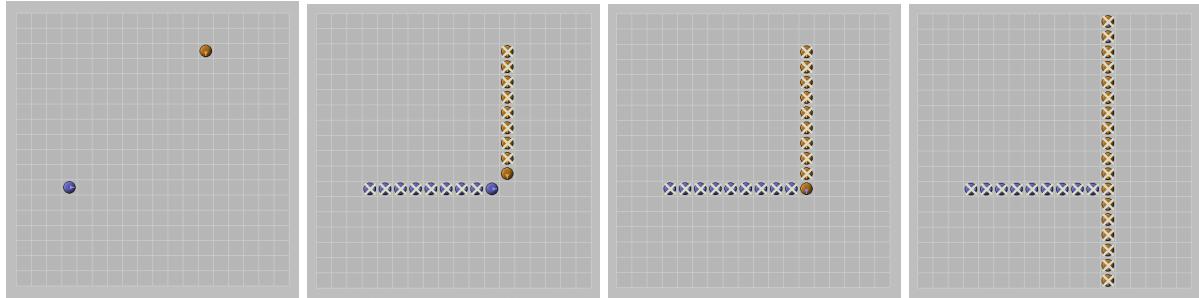
Figuur 1: Goed verloop

Er zijn ook verschillende gevallen waarin het niet mogelijk was om een vierkant te maken. De eerste en meest voorkomende hiervan is het geval waarin de agenten aan het begin zo geplaatst worden dat zij dezelfde kant opkijken of dat één van de twee agenten 180 graden de andere kant opkijkt. In dit geval zullen de agenten parallel aan elkaar zoeken waardoor zij nooit een lid van de andere familie tegen kunnen komen. Van de experimenten was het in 26 gevallen zo dat de agenten parallel aan elkaar zochten. Deze situatie is weergegeven in Figuur 2.



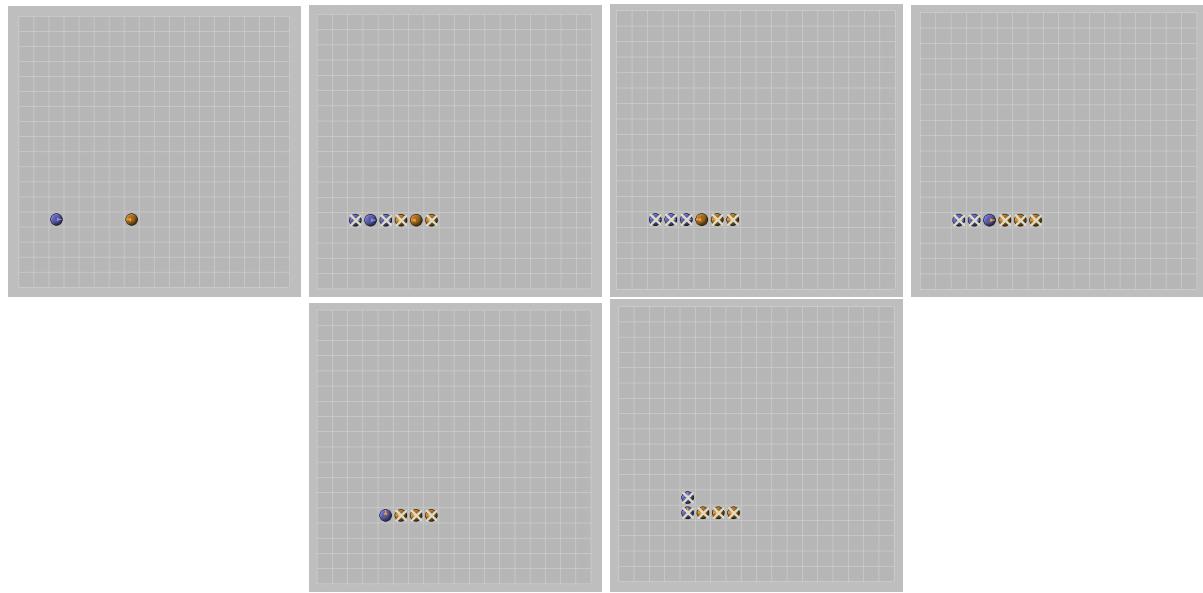
Figuur 2: Parallel verloop

Het tweede geval waarin geen vierkant gemaakt kon worden was wanneer de agenten elkaar wel zouden kruisen maar tegelijkertijd hetzelfde vakje scannen wat op dat moment nog leeg is. Beide agenten komen dan tot de conclusie dat zij op dit vakje een eigen kind kunnen plaatsen (een typisch voorbeeld dat de omgeving hier dynamisch is en dat dat tot problemen kan leiden). In dit geval zal één van de twee agenten (de oudste) een kind kunnen plaatsen welke als taak doorkrijgt het aanmaken van eigen kinderen. De andere agent zal zichzelf op inactief zetten omdat deze de code voor het maken van een kind denkt te hebben uitgevoerd. De nog actieve familie blijft kinderen aanmaken totdat die een familielid tegenkomt. Daarna wordt deze familie ook inactief. In de experimenten kwam deze situatie één maal voor. Een visuele weergave van dit geval is weergegeven in Figuur 3.



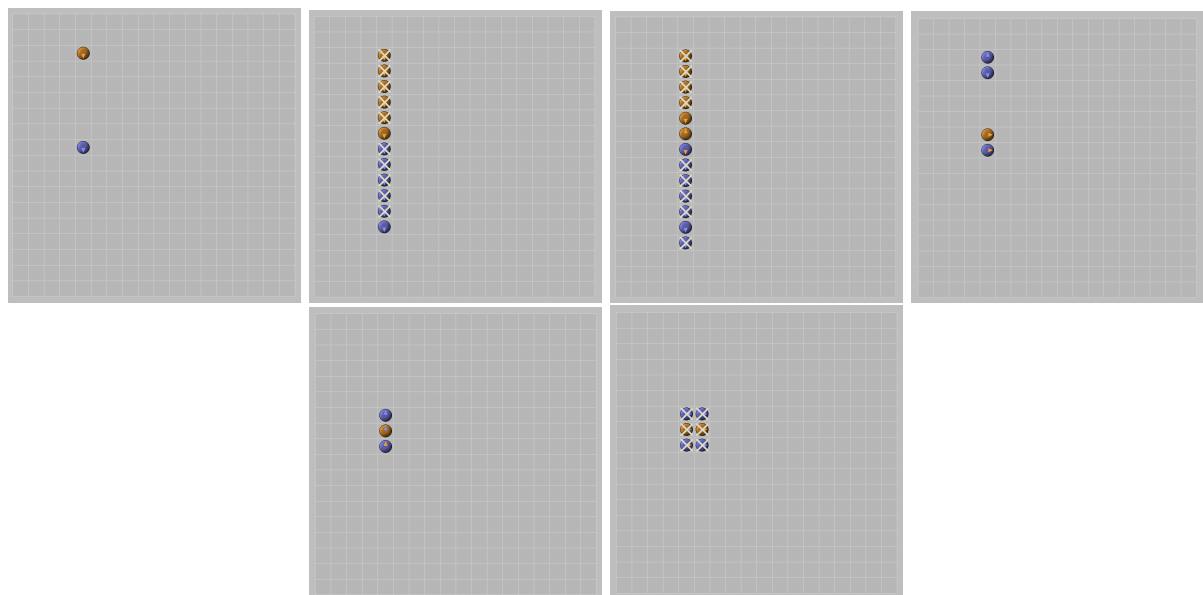
Figuur 3: Zelfde vakje scannen

Het derde geval waarin geen vierkant gemaakt kon worden was dat in de startsituatie de agenten elkaars richting op keken met een even aantal vakjes tussen de twee agenten. Op een gegeven moment zullen de agenten die in het normale geval over zouden blijven elkaar hier tegenkomen. De oudste agent zal in dit geval de ander agent uitzetten en de taak van laatste-agent meegeven. Deze laatste-agent zal hierdoor weer aanschieten en starten in de code waarin deze net bezig was en de andere agent uitzetten. De agent zal op een gegeven moment tot de conclusie komen dat hij een lid van de andere familie is tegengekomen maar aangezien hij ziet dat hij zelf al een laatste-agent is zal hij die agent van de andere familie niet de laatste-agent taak meegeven. Als gevolg zal alleen de agent die de laatste-agent taak heeft meegekregen haar kinderen opruimen en haar deel van het vierkant maken maar zal de andere familie door inactiviteit dit niet kunnen doen. De situatie waarbij dit zo was kwam één keer voor. Deze situatie is weergegeven in Figuur 4.



Figuur 4: Elkaar aankijken

Het laatste geval waarin geen vierkant gemaakt kon worden was het geval waarin de initiële agenten dezelfde richting opkeken en een kind van één van de agenten de andere agent tegenkomt. Deze agent zal nu van zichzelf en de andere agent een laatste-agent maken waarna deze twee laatste-agenten de taak aan hun kinderen doorgeven om zichzelf op te ruimen. Echter zal de agent die niet de andere agent tegenkwam ook nog doorgaan met het maken van kinderen en één van die kinderen zal dan op een gegeven moment bij de laatste-agent van de andere familie aankomen. Omdat deze agent een lid van een andere familie tegenkomt maakt deze agent zichzelf ook een laatste-agent waardoor er in totaal drie laatste-agenten overblijven omdat deze laatste-agent niet wordt opgeruimd omdat hij immers een laatste-agent is. De overgebleven agenten zijn er twee uit een familie en één van de andere familie. Op het einde zal hierdoor een rechthoek in plaats van een vierkant gemaakt worden. Deze situatie kwam ook eenmaal voor. Deze situatie is weergegeven in Figuur 5.



Figuur 5: Achtervolgende agenten

Wanneer het bord begint met meer dan twee agenten dan geldt dat wanneer alle agenten parallel aan elkaar zoeken de families van deze agenten alleen hun eigen familie tegen zullen komen waardoor er geen vierkant ontstaat. Wanneer agenten elkaar wel tegenkomen zal op die plek een vierkant ontstaan. Het kan bij meer dan twee agenten ook zo zijn dat er één vierkant ontstaat en dat andere families van agenten alleen een rechte lijn maken welke door de andere families nooit geraakt wordt, die lijn blijft dan bestaan omdat deze familie niet wordt aangezet tot opruimen. Het kan ook zo zijn dat alle families van agenten een andere familie tegenkomen waardoor alle families hun kinderen opruimen en waardoor er zo meerdere vierkanten ontstaan. Ook bij een initieel aantal agenten van groter dan twee gelden de eerder beschreven situaties waarin geen vierkant gevormd kon worden door agenten die elkaar al dan niet tegenkomen.

7 Conclusie

Voor de opdracht van RoboCom was het de bedoeling om een agent te maken welke, wanneer er twee van op het bord worden gezet, zo vaak mogelijk één vierkant maken. Zoals uit de experimenten blijkt, weet de ontwikkelde agent in circa 42% van de gevallen ook echt één vierkant te maken. Wanneer een vierkant gemaakt wordt duurt het ongeveer 13000 RoboCom tijdseenheden voordat een vierkant gemaakt is.

Een vervolgonderzoek kan zich richten op de tijd die de agenten erover doen om een vierkantje te maken. Binnen het programma bestaat een variabele voor de laatste-agenten die bepaald hoe lang zij wachten voordat zij beginnen met het maken van het vierkantje. In vervolgonderzoek zou gekeken kunnen worden naar een optimale (lagere) beginwaarde waarbij nog steeds altijd één vierkant gevormd kan worden zonder in de problemen te komen. Verder kan vervolgonderzoek zich richten op het verbeteren van het zo vaak mogelijk kunnen maken van één vierkantje. Hierbij zouden de initiële agenten ook opzij kunnen zoeken naar de andere familie zodat er altijd een kruising van elkaars wegen plaatsvindt. Tevens zou in het vervolg nog gekeken kunnen worden naar de overige edge cases die bij de experimenten besproken zijn en manieren om die te verhelpen.

Referenties

- [1] Bemann, D. C. (z.d.). RoboCom Documentation. RoboCom. Geraadpleegd op 23 maart 2021, van https://web.archive.org/web/20050105174954fw_//http://www.cyty.com/robocom/download/RobSci_E.html
- [2] S.J. Russel, P. Norvig, (2021), Artificial Intelligence A Modern Approach (Fourth edition, Vol. 1), Pearson Education.
- [3] The RoboCom Team. (z.d.). RoboCom - The Programming Game. RoboCom. Geraadpleegd op 22 maart 2021, van https://web.archive.org/web/20061017143906fw_//http://www.cyty.com/robocom/?area=main
- [4] W. Kosters,(z.d.), Kunstmatige intelligentie - Programmeeropgave 2 - Agenten & Robotica, Geraadpleegd op 22 Maart 2021, van <https://liacs.leidenuniv.nl/~kosterswa/AI/robot2021.html>.

Appendix: Code

Voor de code voor deze opdracht was geen code beschikbaar en derhalve is de gehele code zelfgeschreven code. Er is bij het schrijven van de code wel gebruik gemaakt van ideeën zoals die besproken zijn in de video's voor deze tweede programmeeropgave [4].

De code voor de RoboCom agent is als volgt:

```
Published Name      Squarer           ; Name of this program
Published Author    Dylan Macquine & Luc Schouten ; Name of author
Published EMail     s2592991/s2660148@vuw.leidenuniv.nl ; Author's e-mail address
Published Country   Netherlands        ; Author's home country
Published Comment   Robocom program for AI @ Liacs ; A comment on this prog
```

```

Secret      Password      YOUR_PASSWORD_HERE ; Password for competitons
Published   OpenSource    No                 ; This prog is not open source
Published   Language     RC300              ; Written in RC300 language
Published   OptionSet    RC3 Standards     ; Recommended OptionSet

Bank Main
SET #5, 5           ;Aantal keer dat een cleannrobot maximaal mag draaien voor dat hij
zichzelf op moet ruimen
SET #6, 300          ;Het aantal keer dat een lastbot moet draaien voordat hij begint met
het maken van een vierkant.
SET #7, 1            ;De kant die de lastbot op moet draaien.

;Alle code tussen deze en de volgende scheidslijn is code voor een "normale" bot welke
;of eigen kinderen aanmaakt en daarna doodgaat. Of zelf de lastbot wordt en de andere activeert.
;;;;;;;;
@Amlastbot
COMP #Active, 10    ;Ben ik de bot die over moet blijven
JUMP @Amcleanbot   ;Als ik niet de overblijfende bot ben, kijken of ik een cleanbot ben
JUMP @Lastbot       ;Als ik de laatste bot ben moet ik zorgen dat de rest zich op gaat ruimen

@Amcleanbot
COMP #Active, 9      ;Ben ik een cleanbot?
JUMP @Normalcourse ;Als ik niet de Cleanbot ben, normaal programmaverloop
JUMP @Cleanup        ;Als ik wel een cleanbot ben, ga verder met het programmaverloop van een cleanbot

@Normalcourse
COMP %Active, 10    ;Kijk of de bot die je tegen komt een lastbot is, als dat zo is, zet hem dan niet uit.
SET %Active,0        ;Zet voor op non active
SCAN #1              ;Kijk naar de bot voor je
COMP #1,0            ;Kijk of het vakje voor je leeg is
JUMP @Checkrelated  ;Als het vakje voor je niet leeg is, kijk of het vakje voor je familie is
Create 2, 1, 0        ;Als het vakje voor je leeg is: creeer een nieuwe bot
Trans 1, 1            ;Verplaats jouw bank naar zijn bank
SET %Active, 1        ;Zet de nieuwe aangemaakte robot aan
COMP #Active, 9        ;Ben je zelf in de tussentijd een Cleanbot geworden?
JUMP 2                ;Nee geen cleanbot, wel een lastbot?
JUMP @Cleanup         ;Ga dan naar de code executie voor een cleanbot.
COMP #Active, 10        ;Ben je zelf ondertussen een active 10 bot geworden?
SET #Active, 0          ;Zet jezelf uit
JUMP @Amlastbot      ;Ga naar de code executie voor een active 10 bot.

@Checkrelated
COMP #1, 2
JUMP @Amcleanbot2   ;Als het vakje voor je geen "familie" is, is het dan een Cleanbot?
SET #Active, 0        ;Als het vakje voor "familie" is, stop

@Amcleanbot2
COMP #Active, 9        ;Ben ik ondertussen wel zelf een opruim-robot?
JUMP @Amlastbot2    ;Dan ben ik zelf een lastbot.
JUMP @Cleanup         ;Ga naar Cleanup code

@Amlastbot2
TRANS 1, 1
COMP #Active, 10        ;Als je zelf een laatste robot bent, zet dan niet de robot voor jou
op active 10 want dat is dan een kind van jezelf.
SET %Active, 10
SET #Active, 10
SUB #7, 1               ;Zorg dat deze lastrobot robot de andere kant opdraait dan de andere
lastrobot bij het maken van het vierkant.
JUMP @Amlastbot

```

```

;Alle code vanaf deze scheidingslijn tot de volgende scheidingslijn is bedoeld voor de bots
die uiteindelijk over blijven.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
@Lastbot
SUB #6, 1           ;Verminder tijd totdat een vierkantje gemaakt wordt.
COMP #6, 0           ;Is het tijd om een vierkant te maken?
JUMP @Lastbotcontinue ;Nee, ga verder met lastbot
JUMP @Makesquare    ;Ja, maak het vierkantje.
@Lastbotcontinue
TURN 0               ;draai rondjes
SCAN #2
COMP #2, 0           ;Is het vakje leeg?
JUMP @Isenemy        ;Nee, is het dan een enemy?
JUMP @Lastbot        ;Als het vakje leeg is, draai verder
@Isenemy
COMP #2, 1           ;Is het vakje een enemy?
JUMP @Initiatecleanup ;Nee, het vakje is een aanverwante, tijd om op te laten ruimen.
JUMP @Lastbot        ;Als het vakje een enemy is, draai verder

JUMP @Lastbot

@Initiatecleanup
TRANS 1, 1           ;Transfer bank 1 naar bank 1 zodat de robot met het programmaverloop begint vanaf de e
SET %Active, 9        ;Zet de robot voor je op opruimen
JUMP @Lastbot

@Makesquare
TURN 0
SCAN #8
COMP #8, 1           ;Is het vakje voor je de andere lastbot?
JUMP @Makesquare    ;Nee? draai verder
TURN #7
CREATE 2, 1, 0        ;Creëer een nieuwe robot op de goede plek.
SET %Active, 0        ;Zet hem meteen uit
SET #Active, 0        ;Zet jezelf uit

;Alle code vanaf deze scheidslijn is enkel bedoeld voor bots in de cleanup modus.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
@Cleanup
TURN 0               ;Draai rond
SUB #5, 1           ;Haal 1 af van het aantal keren dat nog gedraaid mag worden voordat je jezelf opruimt
COMP #5, 0           ;Mag je nog verder draaien?
JUMP @Mayturn        ;Mag nog steeds verder draaien.
DIE                 ;Mag niet meer verder draaien, ga dood

@Mayturn
SCAN #3
COMP #3, 0           ;Is het vakje voor je leeg?
JUMP @Isinactive     ;Is ie niet leeg? is het dan misschien een inactieve bot
JUMP @Cleanup

@Isinactive
COMP %Active, 0      ;Is het een inactieve robot?
JUMP @Isactive1      ;Is het dan een Active 1 robot?
SET %Active, 9        ;Ja, het is een inactieve robot, zet hem op opruimen.
DIE

@Isactive1
COMP %Active, 1      ;Is het een Active 1 robot?
JUMP @Cleanup         ;Nee, draai verder
SET %Active, 9        ;Ja het is een Active 1 robot, zet hem op opruimen

```

DIE

;Taak volbracht, tijd om dood te gaan.