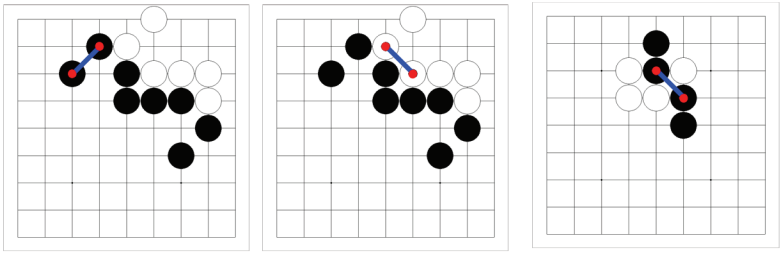
Computer leert connectiviteit van Go-expert

Leo Dorst (1 kyu), UD UvA 15 januari 2007
(leo@science.uva.nl)

Het was een unieke gelegenheid: één van Nederlands sterkste Go-spelers en meervoudig Nederlands kampioen Emil Nijhuis (6e dan) die op 15 december 2006 afstudeerde op zijn master thesis "Learning Patterns in the Game of Go" [1] aan de Universiteit van Amsterdam (in de opleiding Intelligent Systems). Het grootste deel van het publiek bestond uit Go-spelers, onder andere zijn leermeesteres Guo Juan, meervoudig Nederlands en Europees kampioen. Ik zat in de beoordelingscommissie, en omdat de thesis vooral ging over het leren van 'verbindingen' in Go leek een artikel voor "De Connectie" wel op zijn plaats.

Emil begint met een korte evaluatie van de 'state-of-the-art' in het programmeren van Go. In zijn bespreking van bestaande programma's is zijn focus niet hoe ze spelen, maar hoe ze *leren* spelen. Hij constateert dat er duidelijk sprake is van een enorme verspilling: ondanks het bestaan van talloze bestanden zoals Frank de Groot's Moyogo met een database van 450.000 partijen, is er nog geen manier gevonden om deze bestanden in een sterk spelend Go-programma om te zetten. Dit contrasteert met het feit dat een menselijke *insei*[1] op grond van het naspelen van 5000 partijen op professioneel *dan*-niveau[2] kan komen. Blijkbaar halen de programma's veel te weinig uit de structuren van de aangeboden voorbeeldpartijen. Emil's afstudeerwerk gaat over manieren om dat effectiever te maken.



Figuur 1. Respectievelijk sterk gebonden, gebonden en voorwaardelijk gebonden connectiviteit

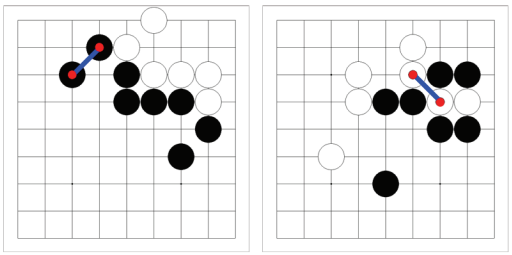
Trouwe lezers van *De Connectie* weten dat Go een fascinerend bordspel is, gebaseerd op simpele regels[3]. Bij Go wil je van een 19×19 bord dat aan het begin van het spel totaal leeg is, net iets meer in beslag te nemen dan de tegenstander. Je zet om beurten, stenen bewegen niet, maar toch is er een enorm duw- en trekwerk gaande, dat nooit verveelt.

Als je de Go-regels net hoort (zie bijvoorbeeld *www.gobase.org*) lijkt het dat 'omsingelen is slaan' de hoofdcomponent van het spel is, en beginners proberen dan ook voornamelijk elkaars stenen te vangen. Maar na een partij of 20 (dus na een week) besef je dat het omsingelen van stenen meer een manier is om te bewijzen dat je een stuk van het bord beheerst, en dan concentreer je je op het maken van gebied, dus 'oppervlakte' – je gaat eerst grof afpalen en dat later verfijnen. Nog wat later, zeg na een partij of 50 (dus na een maand), begin je te beseffen hoe belangrijk de contouren van het gebied zijn; daar vindt de actie plaats, het feitelijke vechten. Groepen stenen worden afgeknipt van de hoofdmoot en kunnen daardoor omsingeld raken, waardoor je zetten kwijt bent aan het beveiligen terwijl de ander er omheen danst en gebied maakt. Je zoekt een balans tussen snelheid en veiligheid, gebruikmakend van je locale tactieken, en ingegeven door globale strategische overwegingen. Reuze spannend allemaal, en eeuwig boeiend.

Emil noemt Go een *battle for territory*. Veel computerprogramma's focusseren op het *territory* aspect en proberen de waarde van zetten te geven aan de hand van de hoeveelheid punten die zetten voor een gebied waard zouden kunnen zijn. Ze hebben ongeveer het niveau van een mens die een jaar intensief speelt. Emil vindt het *battle* aspect belangrijker in hoe een Go-speler feitelijk denkt over zijn zetten, en wil programma's helpen beter lokaal te vechten. Hij gelooft dat we daarbij allereerst moeten proberen om bepaalde aspecten automatisch te leren evalueren; de basisconcepten van Go zoals connectiviteit, zwaktes, invloed, levenskansen van groepen, en uiteraard ook gebiedsschattingen. Hij maakt daarmee een begin door *connectiviteit* bij de horens te nemen. Deze methode van aanpak is volgens hem een voorbeeld van hoe je andere concepten ook zou moeten behandelen.

Connectiviteit lijkt een simpele zaak: stenen zijn wel of niet met elkaar verbonden; als ze dat wel zijn noemen we ze een *keten*. Iedere Go-speler weet: *connected stones share a common fate*; ze worden als geheel geslagen (of niet). We moeten dus altijd denken in termen van ketens; een enkele steen is gewoon de kortst mogelijke keten. Niet-triviale connectiviteit gaat dus tussen ketens: die kunnen verbonden of geknipt worden door middel van zetten. Voor de meeste spelsituaties hoef je daarbij lokaal maar één zet vooruit te kijken; maar als er een *ko*[4] op het bord is wordt het belangrijk of twee opeenvolgende zetten een verbinding of knip kunnen bewerkstel-

1. Een officiële Go-student die ernaar streeft professional te worden. (Red.)
2. Een graad van speelsterkte. (Red.)
3. Zie het artikel over Go in De Connectie van April 2005. (Red.)
4. Een spelsituatie waarbij een oneindige herhaling van zetten zou kunnen ontstaan, ware het niet dat de Ko-regel dit verbiedt. (Red.)



Figuur 2. Respectievelijk gescheiden en sterk gescheiden connectiviteit

ligen (zie de regels op *www.gobase.org*). Daarom onderscheidt Emil verschillende maten van verbondenheid:

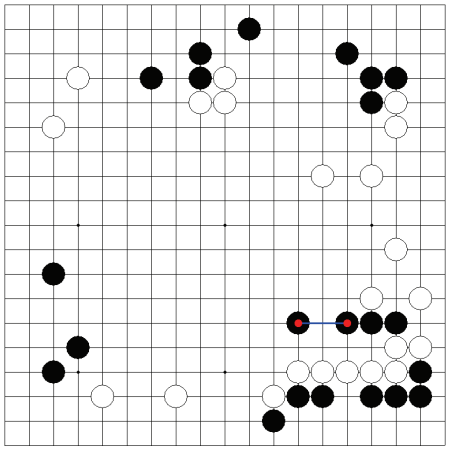
1. sterk verbonden: geen twee opeenvolgende zetten kunnen de verbinding verbreken

2. verbonden: alleen twee opeenvolgende zetten kunnen de verbinding verbreken
3. voorwaardelijk verbonden: alleen verbonden als er nog een zet aan wordt besteed

4. gescheiden: alleen met twee zetten zijn de ketens te verbinden
5. sterk gescheiden: alleen met drie zetten te verbinden

Voorbeelden van die klassen zijn te vinden in Figuren 1 en 2. Als je over de verbondenheid van willekeurige ketens wil praten is het handig nog een zesde klasse in te voeren, waarin ketens zover van elkaar verwijderd zijn dat hun connectiviteit niet
relevant is.

Dit verrijkte concept van connectiviteit van ketens is wat Emil wil laten leren door de computer, omdat dat tot een lokaal spelinzicht leidt die de spelkwaliteit potentieel kan verbeteren. Hij laat het programma direct op grond van lokale patronen in
een omgeving een voorspelling doen over het soort connectiviteit, in plaats van door een uitgebreide analyse. In het voorbeeld van



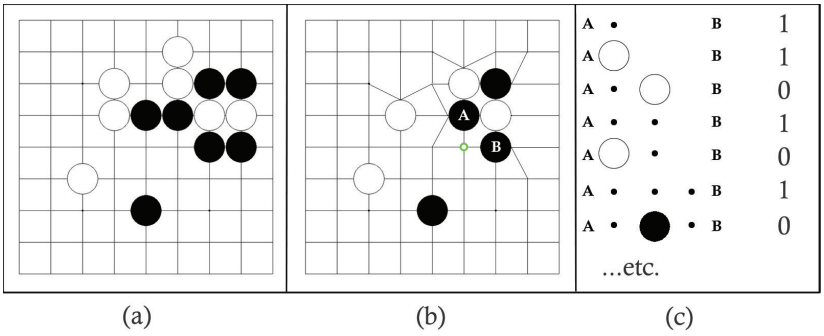
Figuur 3. Een voorbeeld van een moeilik te bepalen connectiviteit

Figuur 3 is al te zien dat zo'n analyse wel eens verder zal gaan lopen dan de lokale omgeving; dit kost teveel
tijd. Voor de gewenste patroonherkenning gebruikt hij AI-technieken uit de *expert learning systems*: een expert geeft van een groot aantal situaties aan wat zijn oordeel is over de mate van connectiviteit, wat leidt tot een gelabelde database. De computer karakteriseert iedere situatie op de een of andere manier (meestal vereenvoudigd) en probeert met handige keuzes van parameters in een programma zo dicht mogelijk bij de voorspelling van de expert te komen voor alle beschouwde situaties. Je wilt daarbij wel een zekere 'generalisatie' bereiken, waarbij nog nooit geziene situaties ook een goede kans hebben om correct voorspeld worden.

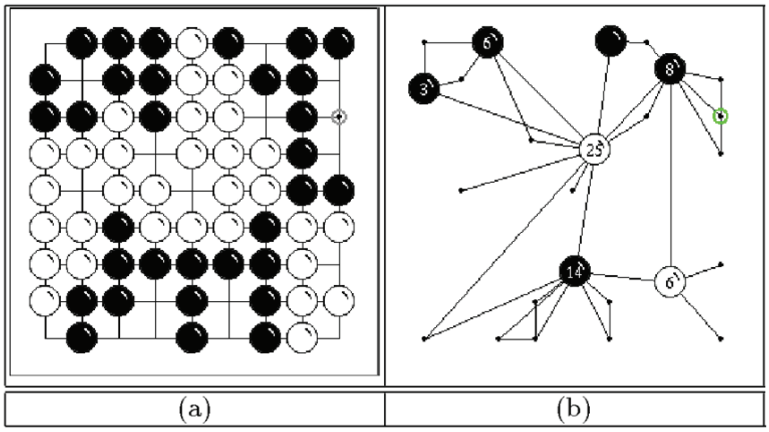
Emil heeft voor het leren van het connectiviteitsbegrip een *tesuji*-encyclopedie genomen van optimale zetten in lokale problemen, uitgegeven door de Nihon Ki-in (de grootste Go-bond in Japan). Daarbij is het wel zo dat die verzameling voor een groot deel bestaat uit gewiekste zetten waar een amateur niet zomaar opkomt; de zetten zijn niet geheel typerend voor normaal spel. Hij heeft alle relevante connectiviteiten voor 300 stellingen expliciet gelabeld. Je zou voor een deel van deze taak ook een automatische analyse kunnen laten uitvoeren; omdat dit een *on-line* stap is mag dit best wat rekentijd kosten. Maar niet voor alle Go-begrippen zal dit kunnen, en labeling door een expert zoals Emil is dan een praktische ontwerpkeuze. Hoe dan ook, je krijgt een gelabelde database voor de bestudeerde aspecten.

Vervolgens komt het leren door de computer. De karakterisering van de lokale situatie door de computer moet vaak en snel gebeuren in iedere partij, daarom moet deze dus relatief simpel zijn. Voor connectiviteit moet je duidelijk in termen van ketens denken, niet alleen maar in een vaste 'omgeving' van een punt (zoals veel programma's naïef doen). Emil volgt daarin Thore Graepel [2]: hij transformeert de bordrepresentatie naar een representatie die volledig in termen van ketens is, niet van stenen. Alleen de lege bordpunten geven nog een idee van de feitelijke geometrie van de situatie. Dit proces is geschetst in Figuur 4. Iedere keten (a) is vervangen in (b) door een enkele knoop in een graaf die de bordsituatie voorstelt. Die knopen zijn aangegeven met enkele stenen, maar nu met een vreemd aantal vrijheden (dat zijn lege bordpunten naast de keten). Die graaf is de essentie van die situatie. Als we dit doen voor een meer uitgespeelde stelling, zoals in Figuur 5, dan herken je linksboven duidelijk een groep met twee ogen die bestaat uit twee ketens. Doordat de lege punten op hun plaats blijven bevat deze vereenvoudigde graaf toch nog de essentiële spelinformatie.

Vervolgens moet dit patroon omgezet worden in getallen die iets zeggen over deze omgeving. Als Emil de omgeving van een keten wil karakteriseren doet hij dat door te tellen hoeveel paden er gemaakt kunnen worden die lopen over zwart, wit of blanco ketens (bordpunten). Dat is aangegeven in figuur 4(c). Je moet ergens stoppen; Emil stopt bij paden van lengte 8. Dat geeft zo'n 1188 getallen per keten. Dat lijkt mij overigens wel wat veel, ze zullen



Figuur 4. (a) de algemene bordrepresentatie, (b) Common Fate Graph, (c) de karakterisering van een verbinding



Figuur 4. (a) de algemene bordrepresentatie, (b) Common Fate Graph, (c) de karakterisering van een verbinding

niet allemaal relevant zijn.

De rest is standaard AI: op grond van een vector van die 1188 getallen probeert een *Support Vector Machine*[5] een oordeel te geven dat lijkt op die van de expert, voor die bepaalde situatie. Je traint zoiets op een leerset, in dit geval van zo'n 6000 voorbeelden van verbindingstypen (misschien wat weinig), en evalueert het resultaat op een testset. Als de karakterisering redelijk goed klopte voor dit probleem zou een redelijk resultaat behaald moeten worden.

Emil's eerste resultaten van deze experimenten staan in Figuur 6, waarin is aangegeven welk percentage van ieder type van connectiviteit wordt voorspeld op grond van de lokale omgeving en wat het feitelijke type was. Het programma heeft duidelijk iets geleerd over connectiviteit. Het voorspelt de correcte klasse in 68,4% van de gevallen. Dat lijkt wat laag, maar in feite is het al voldoende als het de verbondenheid kon voorspellen zonder precies te weten of het nu 'sterk' verbonden was of niet. Dat doet het met 85% nauwkeurigheid. Op deze verzameling problemen is dat behoorlijk goed, want de tesuji's in het boek betreffen nou juist situaties waarin de verbinding van ketens een echt probleem kan zijn. Op gewone partijsituaties zou het programma aanzienlijk beter kunnen zijn, maar dat moet toekomstig onderzoek uitwijzen.

	voorspellingen				
expert labels	1	2	3	4	5
1	50.5%	46.2%	3.2%	0%	0%
2	6.9%	79.9%	13.1%	0%	0%
3	0%	28.3%	63.0%	7.6%	1.1%
4	0%	2.1%	54.2%	35.4%	8.3%
5	0%	0%	13.3%	20.0%	66.7%

Figuur 6. Voor de vijf types van verbondenheid is de gemeten voorspelkans van Emil's software aangegeven in deze 'confusion matrix'

Emil is ook nog iets verder gegaan in richting van automatisch spelen: hij probeert het 'zettype' te voorspellen van de oplossing van ieder tesuji-probleem. Dat zettype wordt gekarakteriseerd als een van 43 voor Go-spelers herkenbare standaardtypen (zoals 'diagonaal', '1-stap sprong', 'paardensprong', en dergelijke). Dat lukt in 65% van de gevallen, voor tesuji-problemen lijkt mij dit een goed begin. Ook hier zal het resultaat in gewone partijen beter zijn.

Tijdens Emil's verdediging kwamen er veel vragen uit de zaal en van de commissie,
die hij overtuigend beantwoordde. Daarbij bleek ook dat hij al verder is dan in de thesis staat; hij vindt nu bijvoorbeeld dat er ook categorieën toegevoegd zouden moeten worden met conditionele verbindingen (afhankelijk van 'ladders').

Dit is pas het begin van een nieuwe benadering voor het programmeren van Go,
en het roept al veel behandelbare vragen op, onder andere: "Hoe simpel kunnen we de lokale representatie maken zonder verlies van resultaat?" en "Is deze representatie overdraagbaar op andere concepten?". Interessant is ook dat de gekozen benadering ruimte geeft voor een interactie tussen mens en computer, omdat het programma ook weet wanneer er twijfels zijn bij het voorspellen van een concept. Emil ambieert zijn onderzoek voort te zetten op de University of Cambridge, in de groep van Graepel. ø

Referenties:

• *[1] Emil Nijhuis, Learning Patterns in the Game of Go, http://www.science.uva.nl/research/ias/alumni/m.sc.theses/#Afgestudeerd06*
• *[2] Thore Graepel, Mike Goutrie, Marco Krauger, and Ralf Herbrich, "Learning on graphs in the game of Go", International Conference on Articial Neural Networks, Vienna, Austria, 2001*

5.Supervised learning-methode voor classificatie. (Red.)