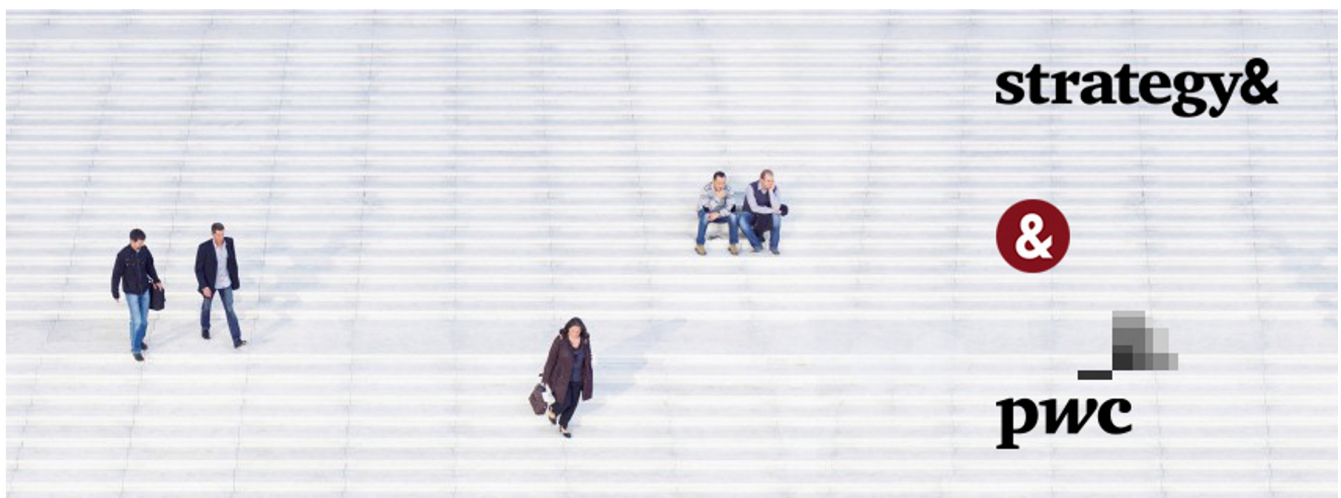


StudeerSnel.nl

201205_met_antw.pdf

Tentamen 16 mei 2012, vragen en antwoorden (toets deel 1)

Rijksuniversiteit Groningen | Kunstmatige Intelligentie | Kunstmatige intelligentie 1(oud)



strategy&
Formerly Booz & Company

Booz & Company zal verder gaan onder de naam **Strategy&**. Aan ons werk, onze mensen en de kantoorlocatie zal niets zal veranderen, we zien alleen maar meer mogelijkheden en kijken ernaar uit om onderdeel te zijn van het wereldwijde netwerk van PwC. Voor meer informatie: www.strategyand.pwc.com/nl

Uitwerking Toets 1: Kunstmatige Intelligentie I (KI1)

16 mei 2012, 9:00-11:00 uur

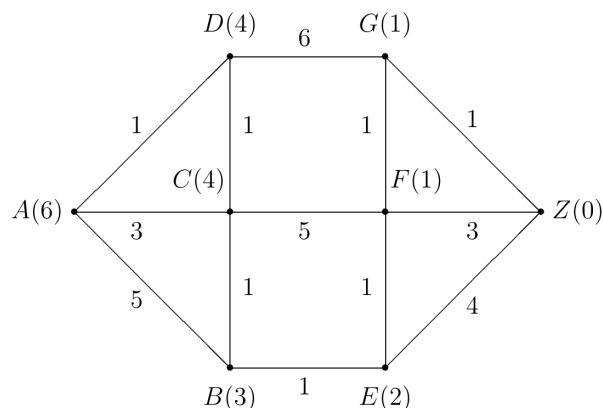
- De toets bestaat uit 4 opgaven. Opgave 1 is 15 punten waard. De opgaven 2, 3 en 4 zijn 25 punten waard. Je krijgt 10 punten cadeau. Iedere opgave heeft deelvragen. De punten voor de opgave worden gelijkmatig verdeeld over deze deelvragen.

Opgave 1: Kennisvragen

- Wat is het verschil tussen de *agent function* en het *agent program* van een agent?
- Een agent wordt wel gespecificeerd volgens een PEAS beschrijving. Waarvoor staat PEAS? Geef een voorbeeld van een PEAS beschrijving.
- Wat wordt verstaan onder een *episodic environment*?
- Wat wordt verstaan onder *conformant problems* en *belief states*?
- Wat is het verschil tussen een *uninformed* en een *informed search problem*?

Opgave 2: Problem solving by searching

We beschouwen de onderstaande ongerichte graaf. Beginknoop is A, doelknoop is Z. De kostenfunctie staat naast de kanten (zijden) in de graaf. Bij de knopen staat tussen haakjes de waarden van een admissible heuristic function h .



- Laat zien dat een zoekstrategie (voor het zoeken van een pad van A naar Z) die gebaseerd is op *Depth-First Search* faalt. Hierbij gaan we er vanuit dat het algoritme niet administret welke knopen reeds bezocht zijn. Ga er bij gelijkwaardige keuzes vanuit dat knopen alfabetisch geordend zijn (dus bij gelijkheid gaat A voor B, etc.).

DFS raakt gemakkelijk in een loop als er niet bijgehouden wordt welke nodes reeds bezocht zijn. In dit geval is de eenvoudige loop $A \rightarrow D \rightarrow A \rightarrow D \rightarrow \dots$ duidelijk.

- In welke volgorde worden de knopen langs gelopen bij het zoeken van een route van A naar Z met behulp van *Breadth-First Search*? Let op, ook nu gebruiken we de alfabetische tie-breaker

regel.

BFS maakt gebruik van een fringe die bijgehouden wordt in een FIFO-queue.

De bedoeling van de opgave was een greedy keuze, waarbij de buurnodes van een knoop in de volgorde van oplopende edge-cost in de queue geplaatst zouden worden. Bij meerdere keuzes met een gelijke cost wordt de alfabetische naam van de knopen als tie-breaker gebruikt.

Bijvoorbeeld: D heeft de burens A (cost 1), C (cost 1) en G(cost 6). Wegens de tiebreaker-rule plaatsen we dan eerst A, vervolgens C en tenslotte G op de queue. De meeste studenten hebben de opgave inderdaad ook zo opgevat. Helaas staat in de opgavetekst niets over de greedy keuzeregels. Oplossingen die alleen de alfabetische tiebreaker gebruiken zijn daarom ook goed gerekend. Er zijn dus 2 antwoorden goed:

1. Greedy: A D C B A C G B D A F C E A D C B B D A F F Z

2. Alfabetisch: A B C D A C E A B D F A C G B C D A B D F B F Z

(c) In welke volgorde worden de knopen langs gelopen bij het zoeken van een route van A naar Z met behulp van *Uniform-Cost Search*? Let op, ook nu gebruiken we de alfabetische tie-breaker regel.

UCS expandeert steeds het pad waarvan de som van de kosten het kleinst is. De fringe wordt daarom bijgehouden in een priority-queue. Uiteraard heeft het geen zin om reeds bereikte nodes weer in de queue te plaatsen met een grotere afstand dan reeds eerder bereikt.

stap	keuze(afstand)	priority queue
1.	A(0)	D(1), C(3), B(5)
2.	D(1)	C(2), C(3), B(5), G(7)
3.	C(2)	B(3), C(3), B(5), F(7), G(7)
4.	B(3)	C(3), E(4), B(5), F(7), G(7)
5.	C	E(4), B(5), F(7), G(7)
6.	E(4)	B(5), F(5), F(7), G(7), Z(8)
7.	B	F(5), F(7), G(7), Z(8)
8.	F(5)	G(6), F(7), G(7), Z(8), Z(8)
9.	G(6)	F(7), G(7), Z(7), Z(8), Z(8)
10.	F(7)	G(7), Z(7), Z(8), Z(8)
11.	G(7)	Z(7), Z(8), Z(8)
10.	Z(7)	×

Ook bij deze vraag zijn twee antwoorden goed gerekend:

1. A, D, C, B, C, E, Z (tot aan de streep in tabel: pad gevonden, niet kortste pad)

2. A, D, C, B, C, E, B, F, G, F, G, Z (kortste pad gevonden).

(d) Wat betekent het dat de heuristic function *admissible* (Nederlands: *toelaatbaar*) is?

De heuristic schat de afstand tot het doel altijd optimistisch in, d.w.z. \leq de echte afstand.

(e) Laat zien hoe het A*-algoritme gebruik maakt van de heuristische function om de de kortste route van A naar Z te zoeken.

A expandeert steeds de node n waarvoor $f(n) = g(n) + h(n)$ het kleinst is. Hierbij is $g(n)$ de echte minimale cost van A naar n , en $h(n)$ de heuristische inschatting van de cost van n naar Z. De fringe wordt ook nu bijgehouden in een priority-queue (met priority $f(n)$). Ook nu heeft het geen zin om reeds bereikte nodes weer in de queue te plaatsen met een echte (!) grotere afstand dan reeds eerder bereikt.*

stap	n ($g(n)$)	priority queue
1.	A(0)	D(5=1+4), C(7=3+4), B(8=5+3)
2.	D(1)	C(6=2+4), C(7=3+4), G(8=7+1), B(8=5+3)
3.	C(2)	B(6=3+3), C(7=3+4), F(8=7+1), G(8=7+1), B(8=5+3)
4.	B(3)	E(6=4+2), C(7=3+4), F(8=7+1), G(8=7+1), B(8=5+3)
5.	E(4)	F(6=5+1), C(7=3+4), F(8=7+1), G(8=7+1), B(8=5+3), Z(8=8+0)
6.	F(5)	C(7=3+4), G(7=6+1), F(8=7+1), G(8=7+1), B(8=5+3), Z(8=8+0), Z(8=8+0)
7.	C	G(7=6+1), F(8=7+1), G(8=7+1), B(8=5+3), Z(8=8+0), Z(8=8+0)
8.	G(6)	Z(7=7+0), F(8=7+1), G(8=7+1), B(8=5+3), Z(8=8+0), Z(8=8+0)
9.	Z(7)	×

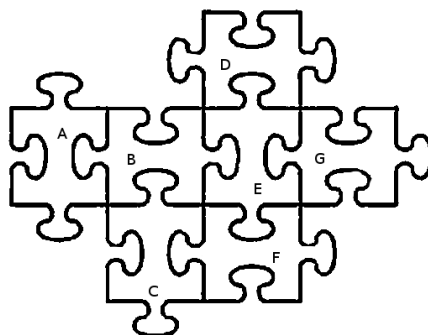
Ook bij deze vraag zijn twee antwoorden goed gerekend:

1. A, D, C, B, E, Z (tot aan de streep in tabel: pad gevonden, niet kortste pad)
2. A, D, C, B, E, F, C, G, Z (kortste pad gevonden).

Opgave 3: Constraint Satisfaction Problems

(a) Geef de formele definitie van een CSP (Constraint Satisfaction Problem).
oplossing

We beschouwen de onderstaande gelabelde puzzelstukjes.



(b) We willen de puzzelstukjes met vier kleuren (W=Wit, Z=Zwart, P=Paars en R=Rood) zodanig kleuren dat buurstukjes verschillend gekleurd zijn. Beschrijf dit als een standaard CSP.

Variabelen: A, B, C, D, E, F, G

Domeinen: $D_A = D_B = \dots = D_G = \{W, Z, P, R\}$

Constraints: $A \neq B, B \neq C, B \neq E, C \neq F, D \neq E, E \neq F, E \neq G$

(c) Los het CSP uit onderdeel (b) handmatig op alsof je een computeralgoritme simuleert. Leg uit hoe je de oplossing vindt. Maak gebruik van de volgende heuristieken: *degree heuristic*, *minimum remaining values* en *forward checking*.

Het proces ziet er stap voor stap als volgt uit:

1. *Vanwege de degree heuristiek beginnen we met E. We kiezen een kleur, bijv. E=W. Met forward checking (FC) updaten we de domeinen: $D_B = D_F = D_D = D_G = \{Z, P, R\}$*
2. *De domeinen van B, D, F en G bevatten nu 3 elementen. Er is dus geen voorkeur via Minimum Remaining Values (MRV), dus we gebruik opnieuw de degree heuristiek. De keuze valt dan op B. We kiezen (bijvoorbeeld) B=Z. Met FC vinden we $D_A = D_C = \{W, P, R\}$.*
3. *Alle resterende domeinen bevatten 3 elementen. We kiezen nu opnieuw via de degree heuristiek voor C. We kiezen (bijvoorbeeld) C = R. Na FC vinden we $D_F = \{Z, P\}$.*
4. *Het domeinen D_F bevat twee elementen, de overige bevatten 3 elementen. Via MRV valt de keuze op F, en we kiezen (bijvoorbeeld) F=Z. Omdat E en C al bekend zijn, heeft FC geen effect.*
5. *De resterende drie variabelen (A, D en G) hebben geen onderlinge constraints. We kunnen we ze een willekeurige waarde uit hun domeinen toekennen: A=P, D=P, G=R.*

(d) Beschouw de volgende Boolean expressie in de Boolese variabelen x_0, x_1, \dots, x_5 .

$$(x_0 \vee x_2) \wedge (x_0 \vee \neg x_3) \wedge (x_1 \vee \neg x_3) \wedge (x_1 \vee \neg x_4) \wedge (x_2 \vee \neg x_4) \wedge \\ (x_0 \vee \neg x_5) \wedge (x_1 \vee \neg x_5) \wedge (x_2 \vee \neg x_5)$$

We beschouwen het probleem dat bestaat uit het vinden van een *assignment* van Boolese waarden aan de variabelen zodanig dat de bovenstaande expressie tot true evalueert.

We gaan nu uit van een random assignment: random toekenning van de waarden 0 en 1 (false en true) aan de variabelen x_0 tot en met x_5 . Bedenk een geschikte heuristiek function die voor de bovenstaande expressie en een gegeven assignment een getal (waardering) oplevert. Wat is de waarde van de heuristiek functie voor een *solution assignment*? Leg uit hoe deze heuristiek gebruikt kan worden in combinatie met een local hillclimbing algoritme om een oplossing te vinden. Leg ook uit wat te doen als het algoritme vastloopt in een lokaal extremum.

De formule is een conjunctie van disjuncties. Ieder van de conjuncten moet true zijn, wil de gehele formule true zijn. Als heuristiek kiezen we daarom het aantal conjuncten dat evalueert tot true bij een gegeven assignment. Omdat de formule uit 8 conjuncties bestaat is de waarde van de heuristiek voor een solution assignment dan natuurlijk 8.

Hillclimbing gaat dan als volgt. Laat s een assignment met heuristiek waarde $h(s)$ zijn.

We beschouwen nu alle mogelijke buurstates n die verkregen worden uit s door een van de waarden van de variabelen te veranderen (flip). Van al die buurstates bepaal je de heuristiek waarde en kiest de grootste. Als deze waarde groter is dan die van s dan 'klimmen' we naar deze nieuwe state. Vanuit de nieuwe toestand proberen we dit proces te herhalen. We zijn klaar als de heuristiek de waarde 8 bereikt heeft. Als we vastzitten in een lokaal maximum, dan kunnen we het gehele proces herhalen met een willekeurige nieuwe begintoestand.

(e) Beschrijf een genetisch algoritme voor het probleem uit onderdeel (d).

Je begint met een populatie van N random begintoestanden. Vervolgens doe je in een loop het volgende proces:

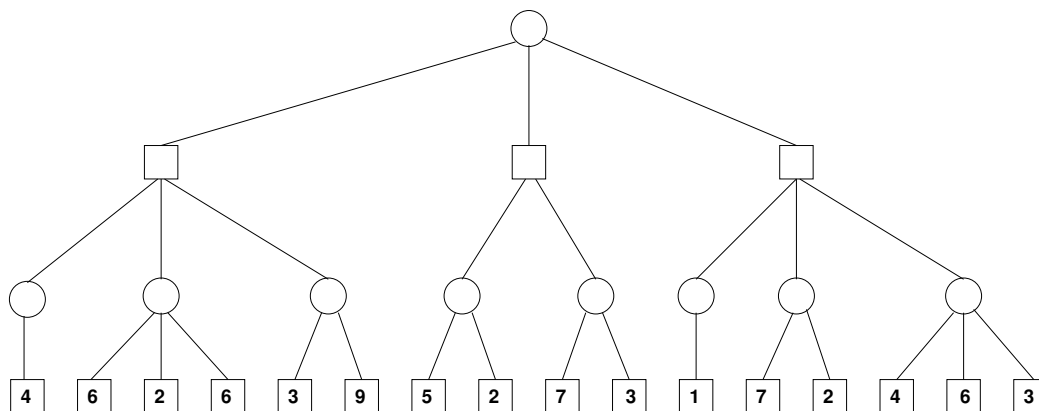
1. *Fitness*: Sorteer de N toestanden op afnemende heuristiek waarde (dus beste vooraan, slechtste achteraan).
2. *Selection+crossover*: Genereer N nieuwe toestanden door de vorige toestanden te kruisen met een kansverdeling die proportioneel is met de ranking: toestanden vooraan de lijst hebben een grotere kans bijdragen te leveren aan 'offspring' dan de toestanden achteraan de lijst. Bij een crossover wordt de locatie van de crossing random bepaald. (Voorbeeld: 010111 en 110000 levert 010000 en 110111 bij crossover na locatie 3).
3. *Mutaties*: Laat random mutaties met een kleine kans toe (voorbeeld: 010000 wordt met een kleine kans 010010).

Uiteraard stoppen we na stap 1 als de beste heuristiek waarde 8 is (oplossing gevonden).

Opgave 4: Games

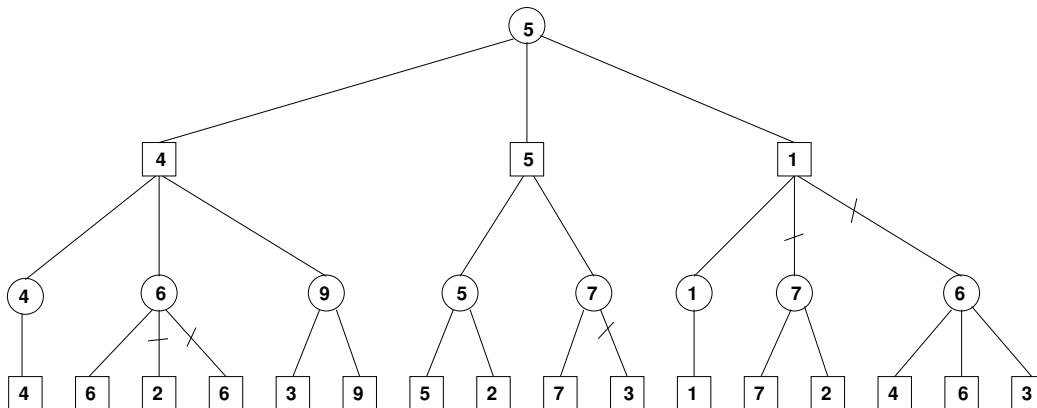
De onderstaande figuur is een volledige variantenboom van een spel waarbij beide partijen volledige informatie over de toestand van het spel hebben en precies weten welke keuzes beide spelers op welk moment kunnen maken.

Er zijn twee spelers: Min en Max. Bij het spel zijn de spelers om de beurt aan zet. Max begint het spel en probeert een zo hoog mogelijke score te bereiken terwijl Min probeert een zo laag mogelijke score te bereiken. De scores die bereikt worden aan het einde van het spel staan in de bladeren van de boom. De knopen van de boom waarin Max aan de beurt is zijn aangegeven met een cirkel, terwijl knopen waarin Min aan de beurt is zijn aangegeven met een vierkantje.



(a) Teken de boom over en vul voor de interne knopen (knopen die nog niet voorzien zijn van een waarde) de evaluaties in die door het *minimax* algoritme worden bepaald. Wat is de beste score die Max kan behalen onder de aanname dat beide partijen optimaal spelen?

De beste score die Max kan bereiken bij optimaal spel van beide spelers is 5, zoals blijkt uit de onderstaande minimax-boom:



(b) Het minimax algoritme bezoekt alle 28 nodes van de boom. Echter, met toepassing van de *alpha-beta pruning* methode zullen minder nodes bezocht worden. Leg uit hoe *alpha-beta pruning* werkt, en geef tevens aan welke nodes van de boom niet bezocht zullen worden als we deze techniek toepassen.

De effectiviteit van *alpha-beta* pruning is sterk afhankelijk van de volgorde waarin in iedere knoop de alternatieven worden langs gelopen.

- (c) Teken de boom waarin de effectiviteit van *alpha-beta* pruning optimaal is. Hoeveel nodes worden in deze situatie bezocht?
- (d) Teken de boom waarin de effectiviteit van *alpha-beta* pruning minimaal is. Hoeveel nodes worden in deze situatie bezocht?
- (e) Als het minimax algoritme, eventueel met pruning, wordt gebruikt voor de implementatie van een programma dat kan schaken (of andere spelen, zoals dammen, go, othello, ...) dan blijkt vaak dat de speelsterkte van het programma sterk wordt beperkt door het zogenaamde *horizon effect*. Wat wordt hiermee bedoeld?