Lineair programmeren kent u wellicht uit de schoolmethodes: zoek een optimum van een doelfunctie onder bepaalde vastliggende voorwaarden. Stochastisch dynamisch programmeren gaat een aantal stappen verder. De voorwaarden zelf treden op met een bepaalde kans en er moet keer op keer een optimum worden gezocht. **Ger Koole** legt uit hoe dat in zijn werk gaat aan de hand van het spel Risk.

Stochastisch dynamisch programmeren

Inleiding

Veel problemen in het dagelijks leven zijn zowel *stochastisch* als *dynamisch* van aard. Stochastisch, omdat we de loop van veel zaken niet kunnen voorspellen. Wiskundig wordt deze onzekerheid gemodelleerd met kansrekening, ofwel stochastiek. Dynamisch, omdat veel beslissingen niet eenmalig zijn, maar omdat er meerdere beslissingsmomenten zijn. Een voorbeeld is het afsluiten van een hypotheek. De beste keuze hangt niet alleen van de huidige, maar ook van de toekomstige financiële ontwikkelingen af. Een verlaging van de rente kan leiden tot vervroegd aflossen of van hypotheek veranderen, om zo de maandelijkse lasten zo laag mogelijk te houden. En dit is een typisch stochastisch dynamisch programmeringsprobleem. Dynamisch vanwege de, zeg jaarlijkse, beslissing om vervroegd af te lossen of oversluiten, stochastisch vanwege de veranderende economische en persoonlijke omstandigheden. Het woord *programmering* wordt om historische redenen gebruikt, beter zou zijn stochastisch dynamisch *optimaliserings*probleem.

Het modelleren

De theorie van het stochastisch dynamisch programmeren levert methoden om optimale beslissingen te nemen in situaties met onzekerheid en herhaalde beslissingen. Maar dan moet 'optimaal' eerst gedefiniëerd worden. Om verschillende beslissingsstrategieën te kunnen vergelijken wordt aan iedere strategie een getal toegekend. Er worden kosten toegekend aan verschillende beslissingen en situaties. (Opbrengsten zijn in dit geval negatieve kosten.) Daarna moet er besloten worden hoe de kosten op elk tijdstip vertaald kunnen worden naar één getal voor de hele beslissingsperiode. Bijvoorbeeld de gemiddelde kosten. Maar vanwege het onvoorspelbare karakter van het probleem is dit echter ook een stochastische grootheid. Om tot één getal te komen wordt daarom in het algemeen de verwachtingswaarde gekozen. Met behulp van deze verwachting kunnen de verschillende strategieën vergeleken worden.
Hoe wordt nu 'de meest geschikte hypotheek' vertaald in wiskundige termen? Ga uit van een maandelijks budget (dat variabel is) en een looptijd van n jaar. De definitie van 'de meest geschikte hypotheek' wordt nu het dusdanig aanwenden van het maandelijkse budget zodat het vermogen na n jaar maximaal is.
Het centrale begrip bij het stochastisch dynamisch programmeren is de *toestand*. Kennis van de huidige toestand geeft alle mogelijke informatie over het toekomstig gedrag van het systeem; alle informatie over het verleden is er in samengebracht. Deze eigenschap heet de *Markov eigenschap,* naar de beroemde Russische wiskundige (1856-1922). De Markov eigenschap zorgt ervoor dat beslissingen alleen van de huidige toestand afhangen en niet van de voorgaande toestanden, de geschiedenis.
Een tweede eigenschap waar de toestand aan moet voldoen is dat deze waarneembaar moet zijn voor de beslisser: in het geval van de hypotheek moeten bijvoorbeeld het beschikbare budget, de rente, de soort hypotheek en het reeds opgebouwde vermogen bekend zijn.
De laatste stap in de specificatie van het probleem is het vaststellen van de wijze waarop het systeem van toestand verandert. Er worden kansen gedefiniëerd die de waarschijnlijkheid van een overgang naar een nieuwe toestand weergeven. In het hypotheekprobleem is zo'n overgangskans samengesteld uit een overgangskans voor de rente en een overgangskans voor het beschikbare budget.
Samen met de te nemen beslissing vormen ze voor elke toestand een kansverdeling op de toestanden voor volgend jaar.

Een optimalisatie algoritme

Tijd voor de wiskundige notaties. Laat de ruimte van alle toestanden gegeven worden door X en de ruimte van mogelijke beslissingen of acties door A. Als we ons bevinden in een toestand $x \in X$ en we nemen een actie $a \in A$, dan is de kans dat er een overgang naar $y \in X$ plaatsvindt gelijk aan $p(x, a, y)$. Aan een toestand-actie combinatie (x, a) kennen we meestal de directe kosten $c(x, a)$ toe. Met deze notatie kan een algoritme bestudeerd worden dat in staat is de optimale strategie te vinden.
Een centraal concept in dit algoritme is de grootheid

$V_t(x)$: de verwachte minimale kosten als er nog t tijdseenheden te gaan zijn. Het interessante van V_t is het feit dat V_{t+1} zich eenvoudig laat uitdrukken in V_t. Stel namelijk dat je in x, met nog $t+1$ beslissingen te gaan, actie a kiest. Dan krijg je meteen kosten $c(x, a)$, en daarna ga je naar toestand y met kans $p(x, a, y)$. Vanuit y heb je dan nog t tijdseenheden te gaan, met kosten $V_t(y)$. De *verwachte* kosten vanuit x, bij actie a, zijn dus:

$$c(x, a) + \sum_{y \in X} p(x, a, y) V_t(y)$$

De verwachte *minimale* kosten zijn:

$$V_{t+1}(x) = \min_{a \in A} \left\{ c(x, a) + \sum_{y \in X} p(x, a, y) V_t(y) \right\}$$

De kosten $V_0(x)$ worden voor elke x bekend verondersteld: er zijn per slot van rekening geen beslissingen meer te nemen. Voor de hypotheek is dit dus het tegengestelde van het eindbedrag. Vanuit V_0 kan nu V_1 berekend worden, en V_2, enzovoorts. Met x als huidige toestand vertelt $V_n(x)$ ons wat we na n jaar als eindvermogen hebben. Het woord 'verwachten' moet hier wel strikt wiskundig worden geïnterpreteerd: het algoritme zegt niets over de mogelijk enorme variantie! Dit principe van achterwaartse recursie heet dynamisch programmeren of waardeïteratie. Het kan voor veel problemen van bescheiden omvang de optimale strategie berekenen.

De Curse of Dimensionality

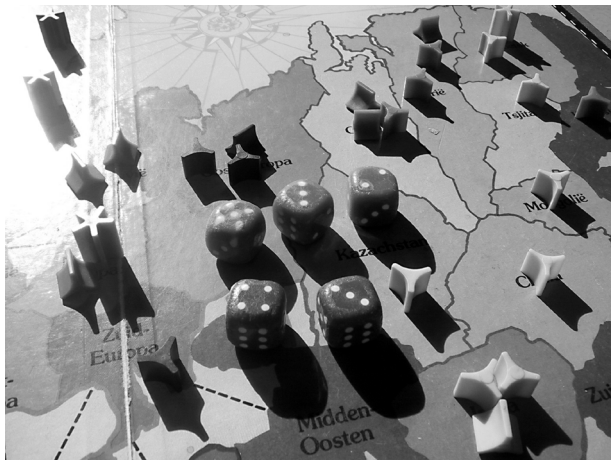
Op het eerste gezicht lijkt dit dynamisch programmeren een wondermiddel voor veel praktische problemen. Maar: de rekentijden om elke toestand op elke tijdseenheid door te rekenen kunnen enorm worden. Zeker als de toestandsruimte uit meerdere componenten bestaat, ofwel meer-dimensionaal is. Het hypotheekprobleem bijvoorbeeld was al vier-dimensionaal (budget, rente, soort hypotheek en opgebouwde vermogen). Bij modellen van productieprocessen kan het aantal dimensies oplopen tot tientallen of zelfs honderden.
En stel dat er D dimensies zijn met elk N mogelijke toestanden, dan zijn er al totaal N^D toestanden. Dit aantal wordt al bij relatief lage D zó groot, dat het praktisch onmogelijk wordt deze allemaal door te rekenen. Dit verschijnsel heet de *Curse of Dimensionality*, een naam bedacht door R. Bellman (Bellman, 1961), een van de pioniers op het gebied van het stochastisch dynamisch programmeren. De laatste jaren is er een hernieuwde belangstelling voor het ontwikkelen van benaderingsalgoritmes die *niet* te lijden hebben onder de Curse of Dimensionality.

Een optimale strategie voor Risk

Risk is een bordspel waarbij de wereld opgesplitst is in 42 territoria. Iedere speler houdt zijn territoria met legertjes bezet.

Het spel Risk

Naburige territoria kunnen worden veroverd met dobbelsteengevechten. De aanvaller mag met maximaal drie dobbelstenen aanvallen, de verdediger mag met één of met twee dobbelstenen verdedigen. Dit herhaalt zich tot de verdediger zijn legertjes verspeeld heeft of de aanvaller zijn aanval staakt.
Nadat de aanvaller heeft gegooid, wordt de worp op aflopende volgorde gelegd, bijvoorbeeld 5 3 2. Besluit de verdediger met één dobbelsteen te gooien, dan wordt deze met de 5 vergeleken. De verdediger wint als hij hoger of gelijk gooit, een 5 of een 6 dus. Gooit de verdediger met twee stenen, dan wordt zijn hoogste worp met de 5 vergeleken en zijn laagste worp met de 3 en geldt dezelfde winst- of verliesregel. Als 5 3 2 bijvoorbeeld gepareerd wordt met 4 3, dan verliezen beide spelers een legertje, gooit de aanvaller 4 4 1 en de verdediger 5 4 dan verliest de aanvaller twee legers en bij 6 3 3 tegen 5 2 verliest de verdediger er twee.



Beiden verliezen hun legertje

De vraag die met dynamisch programmeren op te lossen is: met hoeveel stenen moet de verdediger gooien bij elke aanvalsworp?
Allereerst moet de te minimaliseren doelfunctie worden

opgesteld. We gaan er vanuit dat beide spelers veel legertjes op hun territoria hebben staan. Het doel is dat de aanvaller zoveel mogelijk legertjes verliest.
Belangrijk is dat een verdedigingsworp met twee stenen tweemaal zo zwaar weegt als een worp met één steen, want er gaan altijd evenveel legertjes van het bord als de verdediger stenen gooit. Een goede doelfunctie is het gemiddeld verlies per door de verdediger ingezet legertje. Een speelwijze die deze functie minimaliseert, maximaliseert ook het gemiddeld verlies van de aanvaller.
Een tijdscomponent wordt geïntroduceerd door aan te nemen dat per tijdseenheid één legertje wordt ingezet. Als er dus twee legertjes worden ingezet dan gaan we van t naar $t-2$.
Een verkeerde doelfunctie zou zijn: het gemiddelde verlies per *worp*. Dit verlies wordt geminimaliseerd door iedere keer met één steen te verdedigen. Maar op aanvalsworpen als 3 2 1 of zelfs 6 1 1 wil je met twee stenen verdedigen om het verlies van de aanvaller zo groot mogelijk te maken.
Notaties: laat $u(x, y)$ de kans op een aanvalsworp $x\ y$ zijn met $x \geq y$. Dus in $u(5, 3)$ tellen we de kansen op worpen als 5 5 3 , 2 5 3 en 3 1 5 bij elkaar.
Evenzo, schrijf $q^2(r, s)$ voor de kans op een verdedigingsworp van $r\ s$ met $r \geq s$. Voor het gemak noteren we ook $q^1(r)$ voor de worp met één steen. Al deze getallen kunnen eenvoudig worden uitgerekend en in tabellen worden opgeslagen.
Als toestand kiezen we de mogelijke worpen van de aanvaller, als actie het aantal door de verdediger geworpen dobbelstenen en als kosten het verwachte aantal verloren legertjes. De enige afwijking van het besproken algoritme is dat we bij actie 2 in één keer twee tijdseenheden verspringen. De door het algoritme gebruikte grootheden zijn als volgt:

$$X = \{(x, y) \mid 1 \leq y \leq x \leq 6\}$$
$$A = \{1, 2\}$$
$$p((x, y), a, (r, s)) = u(r, s)$$

$$c((x,y), 1) = \sum_{1 \leq r < x} q^1(r)$$
$$c((x,y), 2) = \sum_{1 \leq r \leq s < 6} q^2(r,s)[I\{r < x\} + I\{s < y\}]$$

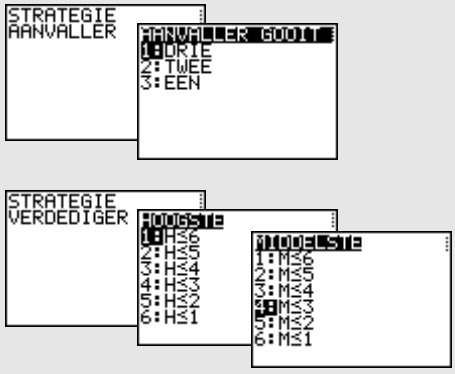
De getallen $c((x, y), a)$ zijn het verwachte verlies aan legertjes bij aanvalsworp $x\ y$ en een verdedigingsworp met a dobbelstenen. De notatie $I\{\cdot\}$ is hier gebruikt voor de indicatorfunctie: $I\{B\} = 1$ als B waar is en 0 als B niet waar is.
De recursie voor het verwachte verlies aan legertjes wordt nu:

$$V_{t+1}(x, y) =$$
$$min\left\{ c((x,y),1) + \sum_{(r,s) \in X} p((x,y),1,(r,s)) V_t((r,s)), \right.$$
$$\left. c((x,y),2) + \sum_{(r,s) \in X} p((x,y),2,(r,s)) V_{t-1}(r,s) \right\}$$

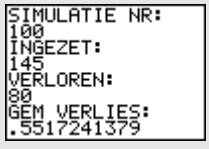
voor alle $t > 0$.
Met $V_0(x, y) = 0$ en $V_1(x, y) = c((x, y),1)$ kan per computer V_n voor $n = 0, 1, 2, \ldots$ berekend worden. De resultaten van de berekeningen zijn als volgt samen te vatten: het is optimaal met twee dobbelstenen te verdedigen alleen wanneer de middelste worp 1, 2 of 3 is. Dus tegen 6 4 moet met één steen gegooid worden, tegen 6 3 met twee. Voor grote n nadert V_n / n tot het gemiddeld verlies. Dit blijkt na afronding gelijk te zijn aan 0,499743.
Dus bij optimaal verdedigen is de verdediger iets in het voordeel; het gemiddeld verlies van de aanvaller is immers 1– 0,499743 = 0,500257.

Simuleren op de TI-83

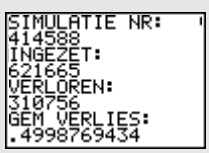
Met behulp van het dynamisch programmeren kunnen de optimale strategieën gevonden worden voor het verdedigen bij het spel Risk. Is een strategie eenmaal bekend, dan kan deze met een programma op de TI-83 gesimuleerd worden. Het programma is te downloaden vanaf www.fi.uu.nl/wiskrant
Het werkt als volgt:

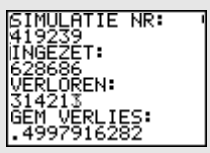


Na het starten van het programma Risk volgt het keuzemenu voor de strategie van de aanvaller: hij kan met één, twee of drie stenen aanvallen. De strategie van de verdediger wordt bepaald door het stellen van bovengrenzen. In het bovenstaande voorbeeld gooit de verdediger met twee stenen wanneer de middelste worp drie of minder is. Nadat vervolgens het aantal simulaties is opgegeven, geeft het programma de volgende output:



Om het verwachte verlies per legertje van 0,499743 te krijgen moet er echter wel heel vaak gesimuleerd worden:





Tom Goris

Had dit ook eenvoudiger gekund, bijvoorbeeld zoals Wierda & Helmich (1992) deden: iedere worp apart bekijken? Een eenvoudige analyse laat zien dat dit niet verstandig is.
Bereken bijvoorbeeld het verwachte verlies voor de aanvalsworp 5 3. Verdedig je met één steen, dan verlies je in vier van de zes mogelijke worpen een legertje. Het gemiddeld verlies per ingezet legertje is dan $\frac{2}{3}$.
Gooi je met twee dobbelstenen, dan zijn er 36 mogelijkheden, die elk met een kans van $\frac{1}{36}$ voorkomen. Merk op dat de meeste worpen dubbel voorkomen. Het is eenvoudig na te rekenen dat de kans om 0, 1 of 2 legers te verliezen elk precies $\frac{1}{3}$ is. Dus het gemiddeld verlies over de worp is $\frac{1}{3}(0 + 1 + 2) = 1$, per ingezet legertje precies $\frac{1}{2}$!
Het spreekt voor zich dat je tegen 5 3 met twee stenen verdedigt. Echter, laten we dezelfde berekening eens maken voor 6 5. Bij een worp met één steen is $\frac{5}{6}$ het gemiddeld verlies . Met twee stenen tegen 6 5 is het verlies echter $\frac{1}{12}0 + \frac{1}{4}1 + \frac{2}{3}2 = \frac{19}{12}$. Per ingezet legertje dus $\frac{19}{24}$, en dat is minder dan het verlies met één steen: $\frac{5}{6}$.
Moeten we nu concluderen dat het tegen 6 5 beter is om met twee stenen te verdedigen? De denkfout die hier gemaakt wordt is dat een aanvaller niet steeds 5 3 of 6 5 gooit. Het is dus verstandig tegen 6 5 met één steen te verdedigen om het andere leger te gebruiken voor een latere, gunstigere worp. De aanvalsworpen kunnen niet apart bestudeerd worden; de winstkansen bij de ene worp hebben invloed op de beslissingen bij andere worpen. Vandaar de noodzaak om met behulp van het stochastisch dynamisch programmeren ook toekomstige worpen in de huidige beslissing te betrekken.

Mijn dank gaat uit naar Richard Boucherie, die mij door consequent doorvragen motiveerde een stelling bij mijn proefschrift uit te werken tot een artikel (Koole, 1994). De sectie over de dobbelsteenduels bij Risk is daarop gebaseerd.

Ger Koole, VU, Amsterdam

Literatuur

Bellman, R. (1961). *Adaptive Control Processes: A guided tour.* Princeton: Princeton University Press.

Wierda, S.J. & J.P Helmich (1992). De aanvaller en de verdediger in het spel Risk. *Nieuwe Wiskrant, 12*(1), 19-24.

Koole, G.(1994). An optimal dice rolling policy for Risk. *Nieuw Archief voor Wiskunde*, *12*(1&2), 49-52.

WisWeb-site vernieuwd!

Sinds 1 februari 2002 is de geheel vernieuwde WisWeb-site bij het Freudenthal Instituut in de lucht. De site ziet er niet alleen anders uit, maar er is ook veel nieuw materiaal (onder andere applets, software, lesmateriaal bij de applets en software) te vinden. Om het u makkelijk te maken is het WisWeb nu ook te vinden onder de url www.wisweb.nl.

Zie voor meer informatie over het WisWeb-project, ontwikkelde materialen en ervaringen, het artikel van Martin van Reeuwijk over de WisWeb-studiedag elders in deze aflevering van de *Nieuwe Wiskrant*.
In het WisWeb-project hebben ook de webmasters van een aantal interessante websites zich voor het Nederlandse wiskundeonderwijs georganiseerd. Deze wiskundewebmasters komen een paar keer per jaar bij elkaar om elkaar te informeren en afspraken te maken over wie wat doet.
Het WisWeb-project krijgt vanaf de zomer van 2002 een vervolg in het WELP-project, waarin samen met de educatieve uitgevers gewerkt wordt aan de integratie van de applets in de verschillende methoden. In het WELP-project zal daarnaast verder onderzocht en ontwikkeld worden hoe een digitale algebraleerlijn – gebaseerd op de applets en boekvervangend – eruit kan gaan zien.
Meer informatie over WisWeb en WELP is te vinden op www.wisweb.nl en zal geregeld worden bijgewerkt en uitgebreid. We heten u van harte welkom.