Technische Universiteit Delft
Faculteit Elektrotechniek, Wiskunde en Informatica
Delft Institute of Applied Mathematics

Oplossingen voor Dreshers hoger lager spel voor $N \leq 980$

(Engelse titel: Solutions to Dresher's guessing game for $N \leq 980$)

Verslag ten behoeve van het
Delft Institute of Applied Mathematics
als onderdeel ter verkrijging

van de graad van

BACHELOR OF SCIENCE
in
TECHNISCHE WISKUNDE

door

Misha Stassen

Delft, Nederland
Mei 2018

BSc verslag TECHNISCHE WISKUNDE

" Oplossingen voor Dreshers hoger lager spel voor $N \leq 980$ (Engelse titel: Solutions to Dresher's guessing game for $N \leq 980$) "

Misha Stassen

Technische Universiteit Delft

Begeleider

Dr. R.J. Fokkink

Overige commissieleden

Dr. D.C. Gijswijt

Dr. B. van den Dries

Mei, 2018

Delft

Inhoudsopgave

1 Hoger lager spel **7**

1.1 Inleiding speltheorie . . . 7
1.2 Beschrijving hoger lager spel . . . 7
1.2.1 Voorbeeld spelverloop . . . 8
1.3 Overzicht . . . 8
1.4 Definities & Notatie . . . 8
1.5 Aantal zoekbomen voor Bob . . . 10
1.6 Van spel naar LP probleem . . . 11
1.6.1 In standaardvorm . . . 12
1.7 LP probleem oplossen . . . 13
1.8 Simplexmethode algemeen . . . 13
1.9 Minimax-stelling . . . 14
1.10 De simplexmethode van Melvin Dresher . . . 14
1.11 Pseudocode algoritme Dresher . . . 20
1.12 Symmetrie . . . 21
1.13 Dreshers algoritme voor N=4 . . . 21

2 Efficiënt de beste zoekboom vinden **29**

2.1 Delayed column generation . . . 29
2.2 Beste gok subroutine . . . 29
2.3 Beste deelrij subroutine . . . 30
2.3.1 Definities . . . 30
2.3.2 Dynamisch programmeren . . . 31
2.3.3 Optimale zoekboom extraheren . . . 32
2.3.4 Voorbeeld . . . 33
2.3.5 Niet valide bomen . . . 34
2.4 Sneller convergeren . . . 34

3 Resultaten **37**

3.1 Correctheid resultaten . . . 37
3.2 Optimale strategie Alice . . . 37
3.3 Berekeningen . . . 42
3.4 Conclusie . . . 46

Abstract

Melvin Dresher beschouwde in zijn boek uit 1961 over speltheorie een getallenraadspel over N getallen. Hij liet zien hoe de optimale strategieën van beide spelers kon worden gevonden met behulp van lineair programmeren. Later toonde Selmer Johnson oplossingen voor $N \leq 11$ en merkte op dat de berekeningen steeds complexer werden bij toenemende N.
Deze thesis beschrijft technieken om het spel op te lossen voor $N \leq 980$. Het algoritme van Dresher vormt nog steeds de basis, maar er zijn enkele aanpassingen. Niet alle voorwaarden worden bijvoorbeeld in het begin gegenereerd. Voorwaarden worden alleen op het moment dat ze nodig zijn gevonden met een subroutine. Dit is een vorm van delayed column generation. Er wordt ook gebruik gemaakt van heuristieken, zoals het schatten van de optimale kansverdeling van één van de spelers.
Dit werk verschaft meer inzicht in het patroon van de oplossingen van dit spel en of enkele eerder geformuleerde vermoedens van Selmer Johnson en Edgar Gilbert over dit spel kloppen.

Hoofdstuk 1

Hoger lager spel

1.1 Inleiding speltheorie

De theorie van strategische spellen kan beschouwd worden als een mathematische theorie van besluitvorming door deelnemers in een competitieve omgeving. De deelnemers worden spelers genoemd en hebben normaal gesproken een bepaalde invloed op de uitkomst van een zekere gebeurtenis. Geen deelnemer of puur toeval alleen bepaalt de uitkomst volledig. Sudoku, Conway's game of life en ganzenbord vallen dus buiten dit domein. De theorie buigt zich over het probleem welke acties optimaal zijn om de doelen van de spelers te bereiken.
Voorbeelden zijn tafelspellen zoals poker, mens-erger-je-niet en schaken, militaire spellen zoals de verdediging van doelwitten tegen aanvallen, maar ook economische spellen zoals de prijscompetitie tussen enkele verkopers. Sommige spellen hebben een kanselement erin zitten. Poker bijvoorbeeld. De uitkomst wordt dan niet alleen door de acties van de spelers bepaald, maar ook door welke kaarten worden gedeeld. Daar hebben de spelers geen invloed op. Ook zijn er spellen die worden gespeeld door meer dan twee spelers. Bijvoorbeeld de prijscompetitie tussen verkopers. Dit soort spellen zal ik verder niet behandelen. Ik richt me in deze thesis op een spel met twee spelers die zonder kanselement van buitenaf de uitkomst van het spel bepalen. De spelers kunnen wel ervoor kiezen om met een bepaalde kans een actie uit te voeren.

1.2 Beschrijving hoger lager spel

Het hoger lager spel is een spel voor 2 spelers en is als eerste beschreven door Melvin Dresher. Beide spelers spreken eerst een getal N af, met $N \in \mathbb{N}$. De ene speler, de verstopper, kiest een geheim getal. Dat getal moet geheel zijn en in het interval $[1, N]$ zitten. De andere speler, de zoeker, moet dan het getal raden en aanvankelijk weet hij alleen het interval waarin het getal is gekozen. Elke keer als de zoeker raadt, moet de verstopper eerlijk antwoorden of de zoeker het correct geraden heeft of dat hij hoger of lager dan zijn gok verder moet zoeken. De zoeker gaat door totdat hij het geheime getal heeft genoemd. (Dus ook als er nog slechts één mogelijkheid is voor dat getal, dan moet hij dat toch nog noemen.)
Het spel draait om het aantal keer raden dat de zoeker nodig heeft om het geheime getal te noemen. De zoeker betaalt dit aantal aan de verstopper. De zoeker wil het aantal keer raden dus minimaliseren, terwijl de verstopper dit juist wil maximaliseren. Beide spelers gaan uit van optimaal spel van hun tegenstander tegen hun eigen strategie.
In het vervolg van dit verslag noemen we de verstoppende speler Alice en de zoekende speler Bob.

1.2.1 Voorbeeld spelverloop

Alice en Bob spreken eerst af dat ze het spel met $N = 10$ gaan spelen.
Alice kiest 6 als haar geheime getal.
Bob raadt 5.
Alice zegt “hoger”.
Bob raadt 8.
Alice zegt “lager”.
Bob raadt 7.
Alice zegt “lager”.
Bob raadt 6.
Alice zegt “correct”.
Bob betaalt 4 aan Alice.

1.3 Overzicht

Het spel oplossen is het vinden van een optimale strategie voor beide spelers. Hieruit volgt de waarde van het spel.
Melvin Dresher [2] gebruikte dit spel als voorbeeldspel voor zijn lineair programmeer algoritme en werkte een oplossing voor $N = 3$ uit. Selmer Johnson [4] lukte het om het spel voor grotere waarden van N op te lossen en vond alle oplossingen voor $1 \leq N \leq 11$. Johnson rekende dit met de hand uit en merkte op dat het voor $N = 11$ al een stuk moeilijker werd en dat voor grotere N een computer gebruikt kon worden. Met een standaard computer is $N \leq 20$ uit te rekenen, maar het aantal strategieën en de rekentijd stijgen explosief. Met het algoritme van Dresher als uitgangspunt en enkele cruciale aanpassingen zijn nu de oplossingen voor $N \leq 980$ gevonden op enkele uitzonderingen na.
In deze thesis wordt eerst getoond dat het oplossen van het spel het oplossen van een lineair programmeer probleem is. Daarna wordt Dreshers algoritme beschreven en aansluitend mijn aanpassingen. Ik heb een vorm van delayed column generation gebruikt en een inschatting van een optimale speelwijze voor Alice. Vervolgens komen de resultaten aan bod. Edgar Gilbert [3] heeft net als Johnson aan het spel gewerkt en zij hadden enkele vermoedens over de optimale strategie van Alice. In het laatste hoofdstuk worden die getoetst.

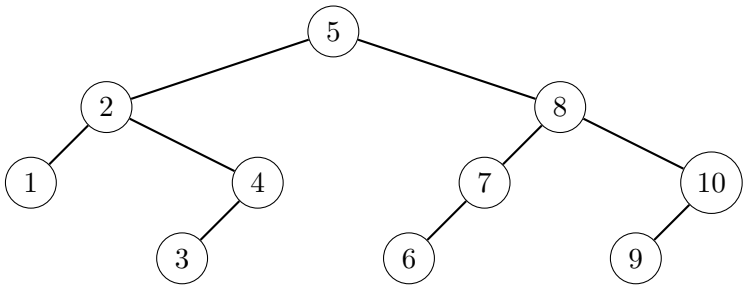
1.4 Definities & Notatie

Voor Bob lijkt op het eerste gezicht een bisectiestrategie optimaal. Steeds wordt het overgebleven interval in twee (bijna) gelijke stukken gedeeld. Op het eind blijft er 1 getal over en het geheime getal wordt dan in maximaal $\lfloor \lg(N) \rfloor + 1$ ronden geraden, waarbij lg staat voor de logaritme met grondtal 2.
Maar als Bob sowieso deze strategie speelt, dan kan Alice daarop inspelen door een getal te kiezen dat in precies $\lfloor \lg(N) \rfloor + 1$ beurten wordt geraden door Bob. Als Bob het verwachte aantal keer raden wil verminderen, dan moet Bob variëren met zijn strategieën. Alice moet eveneens variëren met haar strategieën, zodat Bob er onmogelijk van kan profiteren, mocht hij de strategie van Alice te weten komen. We dienen hierbij onderscheid te maken tussen de volgende soorten strategieën.

Definitie 1. *Een* pure strategie *is een handelingsvoorschrift die in elk mogelijk te bereiken situatie voor een speler de te spelen zet bepaalt.*

Definitie 2. *Een* gemengde strategie *is een kansverdeling over de pure strategieën.*

De pure strategieën van Alice zijn simpelweg de getallen 1 tot en met N die ze als geheim getal kan kiezen. Voor Bob ligt dat anders. Hij lijkt op meerdere momenten in het spel een keuze te moeten maken. Hij kan dit echter terugbrengen tot 1 keuze in het begin van het spel. Dan denkt Bob meerdere stappen vooruit en bepaalt hij welk getal hij verder raadt, als Alice lager zegt of hoger. Eigenlijk maakt hij dan een zoekboom zoals in fig. 1.1:



Figuur 1.1: Voorbeeld binaire boom

Als Bob deze zoekboom speelt en Alice heeft 6 als geheim getal gekozen, dan verloopt het spel, zoals in eerdergenoemd voorbeeld. Bob hoeft dan tussendoor geen nieuwe beslissingen meer te nemen.

Definitie 3. Binaire zoekbomen *zijn gewortelde bomen, waarbij elke knoop een sleutel heeft en maximaal 2 subbomen: De linkersubboom en de rechtersubboom. De boom voldoet ook aan de eigenschap dat de sleutel in elke knoop groter is dan de sleutels van de knopen in de linkersubboom en kleiner is dan de sleutels van de knopen in de rechterboom.*

De pure strategieën van Bob in het hoger lager spel zijn alle binaire zoekbomen met N knopen.

Definitie 4. *Een* twee-speler nulsomspel *is een spel met twee deelnemers, waarbij de winst van de één gelijk is aan het verlies van de ander. Hun nettowinst opgeteld geeft dus nul.*

Het hoger lager spel is dus een twee-speler nulsomspel. Er zijn twee spelers en Bob betaalt evenveel als Alice krijgt. In het voorbeeld "wint" Bob -4 en wint Alice 4. De som daarvan is nul.

Definitie 5. *De strategische vorm van een twee-speler nulsomspel is gegeven door een triple* $(\mathcal{A}, \mathcal{B}, M)$*, waarbij:*

1. $\mathcal{A}$ *is een niet-lege verzameling, de verzameling van strategieën van speler I*
2. $\mathcal{B}$ *is een niet-lege verzameling, de verzameling van strategieën van speler II*
3. M *is een reëelwaardige functie op* $\mathcal{A} \times \mathcal{B}$

Als beide spelers een strategie geselecteerd hebben, dan ligt het spelverloop vast en dus ook de uitkomst van het spel.

Definitie 6. *Voor een gegeven verzameling pure strategieën* $\mathcal{A}$*, zijn de bijbehorende* gemengde strategieën *de discrete kansvariabelen met domein* $\mathcal{A}$*, dwz., het zijn de functies* f *zdd.* $\sum_{a \in \mathcal{A}} f(a) = 1$ *en* $f : \mathcal{A} \to [0, 1]$ *positief.*

De uitkomst van een spel bij twee gegeven pure strategieën wordt gegeven door het spel M, en bij twee gegeven gemengde strategieën wordt dit gedefinieerd als

Definitie 7. *Het verwachte aantal keer raden bij de gemengde strategieën a en b wordt gedefinieerd als*

$$waarde(M, a, b) = \mathbb{E}(M(a, b)) = \sum_{\tilde{a} \in a} \sum_{\tilde{b} \in b} M(a, b) P(\tilde{a} = a) P(\tilde{b} = b)$$

Definitie 8. *Optimaal spel volgens het minimax principe is zodanig spelen dat je de maximale verwachte winst van de tegenstander minimaliseert.*

Dit houdt in dat je tegenstander geen voordeel kan halen uit kennis van je gemengde strategie. Je kunt dus eerst je strategie bepalen en je tegenstander vertellen wat je gaat doen. Ook al kiest je tegenstander daarna zijn strategie, dan nog kan hij niet profiteren van de kennis van jouw strategie.

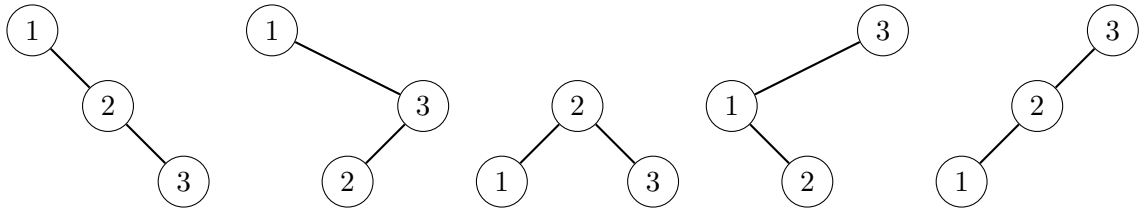
Definitie 9. *De waarde van een spel is de verwachte uitkomst, als beide spelers optimaal spelen.*

Definitie 10. *Het* hoger lager spel met getallen tot en met N *is het spel gegeven door* M : $([1, N]) \times \{$*zoekbomen met* N *knopen*$\}$ *met*

$$M(k, F) = \textit{de diepte van } k \textit{ in zoekboom } F$$

voor $k \in [1, N]$ *en* F *een binaire zoekboom met* N *knopen.*

Laten we als voorbeeld alle strategieën van Bob op een rijtje zetten voor $N = 3$. Dit zijn dus alle binaire zoekbomen met 3 knopen:



Bob begint bij de knoop die het hoogst is en raadt het getal dat die knoop bevat. Daarna stopt hij als hij het goed heeft geraden en anders gaat hij links of rechts naar beneden afhankelijk van het antwoord van Alice. Nu kunnen we de uitkomst van elke strategie van Bob tegen elke strategie van Alice bekijken en in een matrix zetten. We noemen dit de payoffmatrix:

$$\begin{bmatrix} 1 & 1 & 2 & 2 & 3 \\ 2 & 3 & 1 & 3 & 2 \\ 3 & 2 & 2 & 1 & 1 \end{bmatrix} \tag{1.1}$$

Elke kolom in de matrix is een pure strategie van Bob. Alice kiest een rij in de matrix. Op plaats (i, j) in de matrix staat het aantal keer raden, oftewel de payoff, als Alice de strategie behorend bij rij i kiest en Bob kolom j kiest. Als Alice 2 als geheim getal kiest en Bob kiest de middelste zoekboom, dan raadt Bob het dus in 1 keer.

1.5 Aantal zoekbomen voor Bob

Voor $N = 3$ hebben we gezien dat Bob 5 strategieën heeft. Dat is ook het derde Catalan-getal en dat is geen toeval. Het is bekend dat het aantal gewortelde binaire bomen met N knopen gelijk is aan het N^{de} Catalan-getal. Elke zoekboom van Bob is geworteld en binair. Er is namelijk een beginpunt en er zijn telkens hooguit 2 afsplitsingen. En er is precies 1 manier om de getallen 1 tot en met N in te vullen in de knopen van zo'n boom. Er geldt dus:

Lemma 1. *Het aantal zoekbomen met N knopen is*

$$\text{Catalan}(N) = \frac{1}{n+1}\binom{2n}{n}.$$

We kunnen het aantal zoekbomen in een formule vatten door het recursief te bekijken. Laat a de plaats van de eerste gok zijn met $1 \leq a \leq N$. Dan is er links van a een subboom ter grootte $a-1$ en rechts een subboom ter grootte $N-a$. Een formule voor het aantal zoekbomen voor Bob is dan:

$$aantal(N) = \sum_{1=a}^{N} aantal(a-1) \cdot aantal(N-a) \tag{1.2}$$

Dit is een andere formule voor de Catalangetallen met Catalan$(0) = 1$ en Catalan$(1) = 1$. Deze getallen komen overeen met de aantallen zoekbomen voor Bob voor $N = 0$ (de lege boom) en $N = 1$.

1.6 Van spel naar LP probleem

Beide spelers kunnen hun optimale strategieën vinden door een Lineair Programmeer (LP) probleem op te stellen. Elk LP probleem heeft variabelen, een lineaire doelfunctie en voorwaarden in de vorm van lineaire ongelijkheden. Een typisch LP probleem ziet er als volgt uit:

$$\begin{aligned} \text{maximize} \quad & \mathbf{c}^T \cdot \mathbf{x} \\ \text{subject to} \quad & \mathbf{A}\mathbf{x} \leq \mathbf{b}, \\ & x_i \geq 0 \end{aligned} \tag{1.3}$$

waarbij $\mathbf{x} = (x_1, \ldots, x_N)$ de variabelen van het probleem zijn. $\mathbf{c} = (c_1, \ldots, c_N)$ zijn de coëfficiënten van de doelfunctie. $\mathbf{A}$ is een $p \times N$ matrix en $\mathbf{b} = (b_1, \ldots, b_p)$ zijn constanten met $b_j \geq 0$. Elk LP probleem kan in deze standaardvorm worden gegoten.
Gevraagd is de combinatie van variabelen $\mathbf{x} = (x_1, \ldots, x_N)$ te vinden die de doelfunctie optimaliseert, maar geen enkele voorwaarde overtreedt.
Laten we het spel vanuit Alice bekijken. Alice wil het verwachte aantal keer raden maximaliseren. Laten we daarvoor de variabele v introduceren. De doelfunctie van Alice wordt dan het maximaliseren van v. De variabelen waar Alice invloed op heeft zijn de kansen α_i waarmee zij geheim getal i kiest. Nu moet Alice die variabelen zodanig kiezen dat Bob geen strategie/zoekboom tot zijn beschikking heeft, waarmee hij minder dan v keer hoeft te raden. Anders kan Bob net die zoekboom spelen. Beide spelers spelen volgens het minimax principe en gaan dus uit van het sterkst mogelijke tegenspel. Zo wordt elke strategie van Bob een lineaire ongelijkheid voor Alice. Het verwachte aantal keer raden voor strategie S_k van Bob moet dan groter of gelijk zijn aan v. Het verwachte aantal keer raden voor strategie S_k is het inproduct van S_k en de gemengde strategie van Alice. De lineaire ongelijkheid wordt dan:

$$\sum_{i=1}^{N} \alpha_i S_{ki} \geq v$$

Voor Alice ziet het gehele LP probleem er dan als volgt uit:

$$\begin{aligned} \text{maximize} \quad & v \\ \text{subject to} \quad & \forall k : \sum_{i=1}^{N} \alpha_i S_{k_i} \geq v, \\ & \alpha_i \geq 0, \\ & \sum_{i=1}^{N} \alpha_i = 1. \end{aligned} \tag{1.4}$$

In het geval van $N = 3$ wordt dat dan:

$$\begin{aligned} \text{maximize} \quad & v \\ \text{subject to} \quad & \alpha_1 \cdot 1 + \alpha_2 \cdot 2 + \alpha_3 \cdot 3 \geq v, \\ & \alpha_1 \cdot 1 + \alpha_2 \cdot 3 + \alpha_3 \cdot 2 \geq v, \\ & \alpha_1 \cdot 2 + \alpha_2 \cdot 1 + \alpha_3 \cdot 2 \geq v, \\ & \alpha_1 \cdot 2 + \alpha_2 \cdot 3 + \alpha_3 \cdot 1 \geq v, \\ & \alpha_1 \cdot 3 + \alpha_2 \cdot 2 + \alpha_3 \cdot 1 \geq v, \\ & \alpha_1, \alpha_2, \alpha_3 \geq 0, \\ & \alpha_1 + \alpha_2 + \alpha_3 = 1. \end{aligned} \tag{1.5}$$

1.6.1 In standaardvorm

Ook dit LP-probleem is naar standaardvorm om te schrijven. Herformuleer eerst de voorwaarden naar:

$$\begin{aligned} \sum_{i=1}^{N} -\alpha_i S_{k_i} + v \leq 0 \qquad & \text{voor } 1 \leq k \leq \text{Catalan}(N) \\ \alpha_i \geq 0 \qquad & \text{voor } 1 \leq i \leq N \\ \sum_{i=1}^{N} \alpha_i \cdot 1 + v \cdot 0 \leq 1 & \end{aligned}$$

Als we de overbodige eis $v \geq 0$ toevoegen, dan kunnen we het volgende definiëren om tot de standaardvorm te komen:

$$\begin{aligned} \mathbf{x} &= (\alpha_1, \ldots, \alpha_N, v) \\ \mathbf{c} &= (0, \ldots, 0, 1) \text{ met } N \text{ nullen} \\ \mathbf{b} &= (0, \ldots, 0, 1) \text{ met Catalan}(N) \text{ nullen} \end{aligned}$$

En matrix $\mathbf{A} = (a_{ij})$ kunnen we definiëren met:

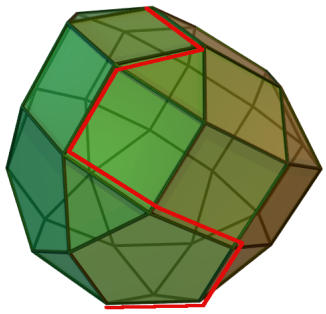
$$a_i = \begin{cases} (-S_{i_1}, \ldots, -S_{i_N}, 1) & \text{als } 1 \leq i \leq \text{Catalan}(N) \\ (1, \ldots, 1, 0) & \text{als } i = \text{Catalan}(N) + 1 \end{cases}$$

1.7 LP probleem oplossen

LP problemen zijn veel onderzochte problemen en veel mathematische softwarepakketten hebben solvers hiervoor. We hoeven dan alleen de doelfunctie en de voorwaarden in te voeren. Als oplossing krijgen we dan de variabelen die de doelfunctie optimaliseren en de waarde van de doelfunctie. De solvers maken gebruik van bijvoorbeeld de simplexmethode. Dit werkt prima voor $N \leq 11$, maar voor grotere N werkt het steeds trager. Daarom zal ik de simplexmethode van Melvin Dresher hier uitleggen, zodat we kunnen onderzoeken welke onderdelen lang duren. Vervolgens zal ik een aanpassing tonen, die voor een grote versnelling zorgt.

1.8 Simplexmethode algemeen

Er bestaan verschillende varianten van de simplexmethode, maar in de basis zijn ze hetzelfde als de methode die in de jaren 40 is ontworpen. In grote lijnen ziet het er als volgt uit.
Als we N variabelen hebben, dan hebben we over het algemeen een N-dimensionale ruimte, waarin de doelfunctie een vector is. We willen een punt zo ver mogelijk in die richting vinden, maar we moeten rekening houden met de randvoorwaarden. Die randvoorwaarden vormen halfruimtes met grensvlakken van $N-1$ dimensies. De deelverzameling van punten die aan alle randvoorwaarden voldoen is een polytoop. Nu zoeken we een punt binnen die polytoop die de doelfunctie optimaliseert. Dit zijn over het algemeen oneindig veel punten, maar het kan ook gebeuren dat er nul of één mogelijkheid is.
Merk op dat alle randvoorwaarden lineair zijn en dat we dus alleen alle punten op de rand hoeven af te gaan. Als we op een grensvlak op de rand zitten, dan kunnen we dezelfde redenering gebruiken, zodat we alleen hoeven te kijken, waar verschillende grensvlakken elkaar snijden. Hoeveel variabelen en dus dimensies het probleem ook heeft, uiteindelijk kunnen we concluderen dat we alleen de hoekpunten van het polytoop af hoeven te gaan. Dit is slechts een eindig aantal punten, waar over het algemeen N grensvlakken elkaar snijden. Alleen in gedegenereerde gevallen, bijvoorbeeld als er een randvoorwaarde loodrecht op de doelfunctie staat, kunnen dat minder dan N grensvlakken zijn. In fig. 1.2 zien we een voorbeeld van een 3-dimensionale ruimte met een groene polytoop, die de punten bevat die voldoen aan de randvoorwaarden.



Figuur 1.2: Grafische weergave van de simplex methode, [1]

De rode lijntjes tonen de speurtocht naar het punt met de optimale oplossing. Het aantal hoekpunten kan exponentieel in N zijn, dus de simplexmethode gaat ze (meestal) niet één voor één af. In de eerste fase van het algoritme wordt een willekeurig hoekpunt van de polytoop gezocht. Als zo'n punt bestaat, wordt vervolgens een verbetering gezocht en naar een naburig hoekpunt gesprongen met een betere waarde volgens de doelfunctie. Merk op dat beide hoekpunten veel grensvlakken gemeen hebben. Eigenlijk wordt slechts één grensvlak ingewisseld tegen

een ander grensvlak. (In het plaatje lijken soms twee grensvlakken tegelijk te worden gewisseld, maar dat is omdat er vier grensvlakken in één punt uitkomen. Er zit dan een stap met lengte nul tussen.)
Zo wordt een aantal keer een beter hoekpunt gevonden, totdat er geen buren met een betere waarde zijn. Hier stopt het algoritme in een lokaal optimum. Is dit ook een globaal optimum? Ja, want alle randvoorwaarden zijn lineair. Zodra we dus in een lokaal optimum zitten, weten we zeker dat we een punt met het globaal optimum te pakken hebben.
Door met een bepaald criterium een goede buur te selecteren gaat het simplexalgoritme meestal een beperkt aantal hoekpunten af om een optimum te vinden. Toch bestaan er LP-problemen, waarbij het simplexalgoritme exponentieel veel hoekpunten bezoekt.

1.9 Minimax-stelling

De minimax-stelling is de hoofdstelling van de speltheorie. Voor elke matrix $A = (a_{ij})$ geldt:

Stelling 1. *Laat $A = (a_{ij})$ een $n \times m$ matrix zijn en X de verzameling van alle gemengde strategieën over n pure strategieën en Y de verzameling van alle gemengde strategieën over m pure strategieën. Dan geldt:*

$$\max_X \min_Y \sum_{j=1}^{n} \sum_{i=1}^{m} a_{ij} x_i y_j = \min_Y \max_X \sum_{j=1}^{n} \sum_{i=1}^{m} a_{ij} x_i y_j = v\,.$$

Uit deze stelling volgt dat elk eindig twee-speler nulsom spel optimale gemengde strategieën heeft. Deze stelling kent verschillende bewijzen. Melvin Dresher bewijst [2, Hoofdstuk 2] het door een methode te beschrijven, die de optimale gemengde strategieën ook daadwerkelijk vindt. De volgende paragraaf geeft zijn methode weer.

1.10 De simplexmethode van Melvin Dresher

We definiëren eerst l en u voor ondergrens en bovengrens:

Definitie 11.

$$l(X) = \min_Y \sum_{j=1}^{n} \sum_{i=1}^{m} a_{ij} x_i y_j = \min_j \sum_{i=1}^{m} a_{ij} x_i$$
$$u(Y) = \max_X \sum_{j=1}^{n} \sum_{i=1}^{m} a_{ij} x_i y_j = \max_i \sum_{j=1}^{n} a_{ij} y_j$$

De minimaxstelling is equivalent aan de volgende bewering: Er bestaat een m-component vector X^* en een n-component vector Y^*, met componenten x_i en y_i, zodanig dat aan de volgende 7 voorwaarden wordt voldaan:

$$
\begin{aligned}
&\text{I.} && x_i \geq 0, && i = 1, 2, \cdots, m; \\
&\text{II.} && \sum_{i=1}^{m} x_i = 1; && \\
&\text{III.} && l(X^*) \leq \sum_{i=1}^{m} a_{ij} x_i, && j = 1, 2, \cdots, n; \\
&\text{IV.} && y_j \geq 0, && j = 1, 2, \cdots, n; \\
&\text{V.} && \sum_{j=1}^{n} y_j = 1; && \\
&\text{VI.} && \sum_{j=1}^{n} a_{ij} y_j \leq u(Y^*), && i = 1, 2, \cdots, m; \\
&\text{VII.} && l(X^*) = u(Y^*)\,. &&
\end{aligned}
\tag{1.6}
$$

Dreshers bewijs bestaat uit het construeren van X^* en Y^*, die aan de 7 vergelijkingen hierboven voldoen. We beginnen het bewijs met de definitie van de geaugmenteerde matrix G van de spelmatrix $A = (a_{ij})$ als volgt:

$$
G = \begin{array}{c}
\begin{matrix} P_0 & P_1 & P_2 & \cdots & P_n & P_{n+1} & P_{n+2} & \cdots & P_{n+m} \end{matrix} \\
\left[\begin{array}{ccccccccc}
0 & 1 & 1 & \cdots & 1 & 0 & 0 & \cdots & 0 \\
-1 & a_{11} & a_{12} & \cdots & a_{1n} & 1 & 0 & \cdots & 0 \\
-1 & a_{21} & a_{22} & \cdots & a_{2n} & 0 & 1 & \cdots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
-1 & a_{m1} & a_{m2} & \cdots & a_{mn} & 0 & 0 & \cdots & 1
\end{array}\right]
\end{array}
\tag{1.7}
$$

Deze matrix heeft $m + 1$ rijen en $n + m + 1$ kolommen. Geef de eerste rij van G het label $i = 0$. Noem de kolommen van links naar rechts

$$
P_0; P_1, \ldots, P_n; P_{n+1} = U_1, U_2, \ldots, U_m = P_{n+m}\,,
$$

waar U_i vectors zijn met een 1 als i^{de} component en voor de rest nullen. Neem aan dat de rijen van A geordend zijn zodanig dat

$$
a_{m1} = \max_{i \leq m} a_{i1}\,. \tag{1.8}
$$

Laat ons nu $m + 1$ kolommen kiezen van G. Deze vormen een $m + 1$ vierkante matrix B. We zullen B een *basis* noemen als B aan de volgende voorwaarden voldoet:

B_1. P_0 is de eerste kolom van B;

B_2. B is niet-singulier, dat wil zeggen, B heeft een inverse B^{-1};

B_3. Voor elke rij van B^{-1}, behalve mogelijk de eerste rij die we gelabeld hebben als $i = 0$, is de eerste component ongelijk aan nul positief.

Een voorbeeld van een basis is de volgende matrix:

$$B^0 = (P_0, P_1, U_1, \ldots, U_{m-1}) = \begin{bmatrix} 0 & 1 & 0 & 0 & \cdots & 0 \\ -1 & a_{11} & 1 & 0 & \cdots & 0 \\ -1 & a_{21} & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & 0 & \ddots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \cdots & 1 \\ -1 & a_{m1} & 0 & 0 & \cdots & 0 \end{bmatrix} . \tag{1.9}$$

Het is makkelijk in te zien dat B^0 niet-singulier is en dat de inverse wordt gegeven door:

$$(B^0)^{-1} = \begin{bmatrix} a_{m1} & 0 & \cdots & \cdots & 0 & -1 \\ 1 & 0 & \cdots & \cdots & 0 & 0 \\ b_1 & 1 & \cdots & \cdots & 0 & -1 \\ b_2 & 0 & 1 & & 0 & -1 \\ \vdots & \vdots & & \ddots & \vdots & \vdots \\ b_{m-1} & 0 & \cdots & \cdots & 1 & -1 \end{bmatrix} = \begin{bmatrix} R_0^0 \\ R_1^0 \\ R_2^0 \\ R_3^0 \\ \vdots \\ R_m^0 \end{bmatrix} , \tag{1.10}$$

waar $b_i = a_{m1} - a_{i1}$. We kunnen de inverse matrix ook uitdrukken in termen van de rijvectoren R_i^0. Uit verg. (1.8) volgt dat $b_i \geq 0$; dus voor elke rij van $\left(B^0\right)^{-1}$, behalve mogelijk voor $i = 0$, is de eerst component ongelijk aan nul postief.

De vectoren ordenen Voorwaarde B_3 in de definitie van een basis kan worden uitgedrukt als een vergelijking van vectoren met de nulvector. Later in het bewijs zal het handig worden twee willekeurige vectoren te kunnen vergelijken, dat wil zeggen, gegeven twee ongelijke vectoren, willen we een regel om bijvoorbeeld de grootste uit te kiezen. Een geschikte ordening van vectoren zal daarvoor worden gedefinieerd.

Laat ons eerst gelijkheid van vectoren definiëren. Twee vectoren $A = (a_i)$ en $B = (b_i)$ zijn gelijk dan en slechts dan als $a_i - b_i = 0$ voor elke i. In andere woorden: A is gelijk aan B (voortaan genoteerd als $A = B$) dan en slechts dan als de vector $A - B$ de nulvector 0 is. Als twee vectoren A en B niet gelijk zijn, bevat de vector $A - B$ tenminste één component ongelijk aan nul. Als de eerste component ongelijk aan nul positief is, dan noemen we A groter dan B en schrijven we $A \succ B$.

Als de rijen van de inverse van een basis B worden genoteerd met $R_0, R_1, \ldots, R_m$, dan zegt voorwaarde B_3 uit de definitie van een basis dat

$$R_i \succ 0 , \qquad i = 1, 2, \ldots, m .$$

Uit deze methode om vectoren te ordenen volgt dat de kleinste van een reeks van vectoren de vector is met de kleinste eerste component; Als meerdere vectoren het minimum delen, dan wordt van die vectoren de tweede component vergeleken, etc.

Optimale basis Laat

$$B = (P_0, P_{j_1}, P_{j_2}, \ldots, P_{j_m})$$

een basis zijn met inverse

$$B^{-1} = \begin{bmatrix} R_0 \\ R_1 \\ \vdots \\ R_m \end{bmatrix} = (C_0, C_1, \ldots, C_m) ,$$

waar de R-en rijvectoren zijn en de C's kolomvectoren. Dan volgt uit het feit dat

$$B^{-1}B = I$$

waar I de eenheidsmatrix is, dat

$$\begin{aligned} R_k P_{j_i} &= 0 && \text{voor } i \neq k\,, \\ R_i P_{j_i} &= 1 && i = k = 0, 1, 2, \ldots, m\,, \end{aligned}$$

waar $j_0 = 0$. In het bijzonder hebben we

$$R_0 P_{j_i} = 0 \qquad \text{voor } j_i = j_1, j_2, \ldots, j_m\,. \tag{1.11}$$

Beschouw nu de inwendige producten met $n + m$ kolommen van de matrix G:

$$R_0 P_j\,, \qquad j = 1, 2, \ldots, n + m\,.$$

Uit verg. (1.11) volgt dat tenminste m van deze producten nul zullen zijn. Als de overige n inwendige producten niet groter zijn dan nul, dan zullen we B een *optimale basis* noemen. Er zal worden bewezen dat een optimale basis optimale gemengde strategiën geeft voor beide spelers. In andere woorden, als de basis B zo is dat

$$R_0 P_j \leq 0, \qquad j = 1, 2, \ldots, n + m\,,$$

dan is B een optimale basis.

Optimale strategiën Laat B een optimale basis zijn. Noem de componenten van de 0-rij en de 0-kolom van B^{-1} als volgt:

$$R_0 = (l, -x_1, -x_2, \ldots, -x_m)\,,$$

$$C_0 = (u, y_{j_1}, y_{j_2}, \ldots, y_{j_m})\,. \tag{1.12}$$

We zullen laten zien dat een optimale gemengde strategie X^* voor Alice uit rij R_0 is te halen door te stellen:

$$X^* = (x_1, x_2, \ldots x_m)\,.$$

Een optimale gemengde strategie $Y^* = (y_1, y_2, \ldots, y_n)$ voor Bob is te halen uit kolom C_0:

$$\begin{aligned} y_j &= y_{j_i} && \text{voor } j_i \leq n, \\ &= 0 && \text{voor alle andere } j. \end{aligned}$$

We zullen bewijzen dat X^* en Y^* optimale gemengde strategiën zijn van het originele spel, door te laten zien dat de hierboven gedefinieerde X^* en Y^* aan de voorwaarden I-VII voldoen van verg. (1.6).

Neem aan dat we een optimale basis hebben. Laat ons de producten $R_0 P_j$ uitrekenen voor verschillende waarden van j.

Als $j = 0$ hebben we

$$1 = R_0 P_0 = \sum_{i=1}^{m} x_i\,,$$

dus aan voorwaarde II is voldaan.

Als $1 \leq j \leq n$ hebben we

$$0 \geq R_0 P_j = l - \sum_{i=1}^{m} a_{ij} x_i \,,$$

dus aan voorwaarde III is ook voldaan.

Als $n+1 \leq j \leq n+m$, dan geldt

$$0 \geq R_0 P_j = R_0 U_i = -x_i \qquad \text{voor } 1 \leq i \leq m,$$

waardoor ook aan voorwaarde I is voldaan.

Omdat B een basis is, geldt

$$R_i \succ 0, \qquad i = 1, 2, \ldots, m \,.$$

In het bijzonder is de eerste component van elke R_i niet-negatief. De eerste component van elke R_i is y_i volgens verg. (1.12), dus aan IV is voldaan.

Omdat $BB^{-1} = I$, hebben we in het bijzonder

$$BC_0 = U_0 \,.$$

Uit deze matrixvergelijking krijgen we $m+1$ vergelijkingen met $m+1$ variabelen $(u, y_{j_1}, y_{j_2}, \ldots, y_{j_m})$. De eerste van deze vergelijkingen is

$$\sum_{j_i \leq n} y_{j_i} = \sum_{j=1}^{n} y_j = 1 \,,$$

omdat de eerste component van elke P_{j_i} gelijk aan 1 is voor $1 \leq j_i \leq n$ en anders 0 is. Dus aan V is voldaan. Nu kunnen de overgebleven m vergelijkingen geschreven worden als:

$$-u + \sum_{j_k \leq n} a_{ij_k} y_{j_k} + \sum_{j_k > n} \delta_{j_k} y_{j_k} = 0, \qquad i = 1, 2, \ldots, m \,,$$

waarbij $\delta_{j_k} \geq 0$. Hieruit volgt

$$\sum_{j=1}^{n} a_{ij} y_j \leq u, \qquad i = 1, 2, \ldots, m \,,$$

waarmee aan VI is voldaan. Tot slot wordt aan VII ook voldaan, omdat u en l door hetzelfde element in B^{-1} worden gedefinieerd.

Een optimale basis construeren We hebben nu het probleem van het vinden van optimale strategieën gereduceerd tot het vinden van een optimale basis. De inverse van zo'n basis geeft optimale strategieën. Nu volgt een iteratieve procedure om een optimale basis en zijn inverse te construeren.

De iteratieve procedure start met een basis, bijvoorbeeld zoals B^0 gedefinieerd is in verg. (1.9). Als B^0 niet optimaal is, dan construeren we op basis van B^0 een nieuwe basis, B^1, die met precies 1 kolom van B^0 verschilt. Dat gebeurt op zodanige wijze, dat als R_0^1 de nulde rij is van $(B^1)^{-1}$, de inverse van B^1, de volgende eigenschap geldt

$$R_0^0 \succ R_0^1 \,. \tag{1.13}$$

Als B^1 niet optimaal is, dan herhalen we het voorgaande algoritme voor B^1, etc. Dit proces genereert een rij van bases. Uit verg. (1.13) volgt dat elke basis hooguit 1 keer voorkomt.

Verder kan het aantal bases niet uitstijgen boven het aantal manieren waarop m kolommen uit de $n+m$ kolommen van matrix G kunnen worden gekozen. Het proces eindigt, wanneer we op een optimale basis uitkomen, wat altijd zal gebeuren.

Veronderstel dat de basis B^0 niet optimaal is, dan bestaat er een P_j in G zodanig dat $R_0^0 P_j > 0$. Om B^1 van B^0 te construeren laten we P_s een kolom van B_0 vervangen, waarbij P_s bepaald is door

$$R_0^0 P_s = \max_j R_0^0 P_j > 0\,, \qquad j = 1, 2, \ldots, n+m\,. \tag{1.14}$$

P_s is dus een pure strategie van de kolomspeler die het beste werkt tegen de huidige gemengde strategie van de rijspeler. In het geval P_s nog niet uniek bepaald is, kies dan P_s met de kleinste index.

Bereken vervolgens de kolomvector $V = (v_0, v_1, \ldots, v_m)$ die voldoet aan de vergelijking $B^0 V = P_s$. Dus hebben we

$$v_i = R_i^0 P_s\,, \qquad i = 0, 1, \ldots, m\,. \tag{1.15}$$

In het bijzonder hebben we

$$v_0 = R_0^0 P_s > 0\,.$$

Verder is er een $i \neq 0$ met $v_i > 0$. Stel immers dat $v_i \leq 0$ voor alle $i > 0$, dan volgt uit de relatie

$$P_s = B^0 V = v_0 P_0 + \sum_{i=1}^{m} v_i P_{j_i}\,,$$

het volgende:

$$P_0 = \frac{1}{v_0} P_s - \sum_{i=1}^{m} \frac{v_i}{v_0} P_{j_i}\,.$$

De kolom P_0 kan dan worden geschreven als een positieve lineaire combinatie van $m+1$ kolommen van G. Uit de definitie van G blijkt echter dat P_0 niet als een positieve lineaire combinatie van andere kolommen kan worden geschreven, dus hebben we een tegenspraak.

Nu kiezen we ervoor om kolom P_{j_r} uit B^0 te verwijderen, zó dat

$$\min_i \frac{R_i^0}{v_i} = \frac{R_r^0}{v_r} \qquad \text{voor } v_i > 0\,,\ v_r > 0\,,\ i \neq 0\,. \tag{1.16}$$

Hieruit volgt dat

$$R_i^0 - \frac{v_i}{v_r} R_r^0 \succ 0 \qquad \text{voor } v_i > 0\,. \tag{1.17}$$

We hebben B^1 uit B^0 geconstrueerd door kolom P_{j_r} te vervangen door kolom P_s, zoals hierboven gedefinieerd. We moeten nog aantonen dat B^1 een basis is. Dat doen we door $(B^1)^{-1}$ uit $(B^0)^{-1}$ te construeren.

Laat R_i^1 de rijen zijn van $(B^1)^{-1}$ voor $i = 0, 1, \ldots, m$. We zullen nu verifiëren dat we R_i^1 verkrijgen uit R_i^0 en v_i met

$$\begin{aligned} R_i^1 &= R_i^0 - \frac{v_i}{v_r} R_r^0 \qquad \text{voor } i \neq r\,, \\ R_r^1 &= \frac{v_i}{v_r} R_r^0\,. \end{aligned} \tag{1.18}$$

Met deze definitie van $(B^1)^{-1}$ zullen we nu nagaan dat $(B^1)^{-1}B^1 = I$. Eerst voor $i \neq r$ en $j_i \neq s$,

$$R^1_k P_{j_i} = R^0_k P_{j_i} - \frac{v_k}{v_r} R^0_r P_{j_i} = 0 \qquad \text{voor } i \neq k\,,$$
$$R^1_k P_{j_i} = R^0_k P_{j_k} - \frac{v_k}{v_r} R^0_r P_{j_k} = 1 - 0 = 1 \qquad \text{voor } i = k\,.$$

Voor $i \neq r$ en $j_i = s$ hebben we

$$R^1_i P_s = R^0_i P_s - \frac{v_i}{v_r} R^0_r P_s = v_i - \frac{v_i}{v_r} v_r = 0\,.$$

Voor $i = r$ en $j_i \neq s$ hebben we

$$R^1_r P_{j_i} = \frac{1}{v_r} R^0_r P_{j_i} = 0\,.$$

En voor $i = r$ en $j_i = s$ hebben we

$$R^1_r P_s = \frac{R^0_r}{v_r} P_s = \frac{v_r}{v_r} = 1\,.$$

Dus verg. (1.18) geeft de inverse van de matrix B^1.

Om het bewijs dat B een basis is te voltooien, moeten we nog aantonen dat $R^1_i \succ 0$ voor $i = 1, 2, \ldots, m$. Voor $i = r$ hebben we $R^0_r \succ 0$ en $v_r > 0$, dus

$$R^1_r = \frac{1}{v_r} R^0_r \succ 0\,.$$

Als $i \neq r$ en $v_i \leq 0$, dan

$$R^1_i = R^0_i - \frac{v_i}{v_r} R^0_r \succ 0\,.$$

Als $i \neq r$ en $v_i > 0$, dan zien we met verg. (1.17)

$$R^1_i = R^0_i - \frac{v_i}{v_r} R^0_r \succ 0\,.$$

B^1 is dus daadwerkelijk een basis en evenzo B^2, B^3 etc.

Tot slot moeten we nog aantonen dat geen basis kan worden herhaald in dit proces. Het is voldoende om verg. (1.13) te bewijzen, oftewel $R^1_0 \prec R^0_0$. Dit volgt uit het feit dat R^0_r een rij is van een niet-singuliere matrix en dus minstens één component moet bezitten die niet nul is. De eerste daarvan moet positief zijn. Omdat $v_0 > 0$ en $v_r > 0$, hebben we

$$R^1_0 P_{j_r} = R^0_0 P_{j_r} - \frac{v_0}{v_r} R^0_r P_{j_r} = 0 - \frac{v_0}{v_r} < 0\,.$$

Ook hebben we

$$R^1_0 P_j = 0 \qquad \text{voor } j = j_1, j_2, \ldots, s, \ldots, j_m\,.$$

Dus minstens $m+1$ kolommen van de $m+n$ kolommen hebben nu de eigenschap $R^1_0 P_j \leq 0$. Dit in vergelijking met minstens m kolommen met deze eigenschap in de voorgaande basis. Verder kan P_{j_r} niet terugkeren in de basis bij de volgende iteratie, omdat een kandidaat s voor de basis moet voldoen aan $R_0 P_s > 0$. Dit voltooit het bewijs.

1.11 Pseudocode algoritme Dresher

In grote lijnen ziet het algoritme van Dresher er als volgt uit:

Algorithm 1 Dreshers algoritme voor LP problemen

```
Creëer een initiele basis B
Bereken α, β en V bij deze basis
b ← boom met kleinste inproduct met α
while b · α < V do
    Bepaal de strategie die uit de basis moet
    Wissel in de basis b om met de strategie die eruit moet
    Update α, β en V
    b ← boom met kleinste inproduct met α
end while
```

1.12 Symmetrie

We kunnen de grootte van het probleem reduceren door gebruik te maken van de symmetrie rond $\frac{N+1}{2}$. Voor Alice is de symmetrische tegenhanger van i het getal $N+1-i$ met $i \in [1, N]$. Elke zoekboom b van Bob heeft een symmetrische tegenhanger b' met $b_i = b'_{N+1-i}$. Merk op dat in sommige gevallen $b = b'$ geldt.

Lemma 2. *Alice kan haar zoekruimte beperken tot symmetrische gemengde strategieën zonder slechter te gaan spelen.*

Stel Alice heeft een optimale gemengde strategie α, waarbij er minimaal één i is met $\alpha_i \neq \alpha_{N+1-i}$. Dan bestaat er een α' met $\alpha_i = \alpha'_{N+1-i}$. α' zal dezelfde payoff hebben als α. Bob kan namelijk minimaal even sterk tegenspel bieden tegen α' als tegen α door zijn bomen te spiegelen. Omgekeerd kan Bob minimaal even sterk spelen tegen α als tegen α' met dezelfde redenering, dus α en α' hebben dezelfde payoff. Beschouw nu $\frac{\alpha+\alpha'}{2}$. Dit is symmetrisch rond $\frac{N+1}{2}$ en een lineaire combinatie van optimale strategieën is ook een optimale strategie. De verzameling van optimale strategieën is immers convex.

Lemma 3. *Bob kan zijn zoekruimte beperken tot symmetrische gemengde strategieën zonder slechter te gaan spelen.*

Het bewijs is identiek aan het bewijs van het vorige lemma.
Door deze reductie is het aantal dimensies ruwweg gehalveerd. Ook het aantal te beschouwen constraints is gehalveerd, omdat we nu alleen naar "pure"strategieën $\frac{b_i+b'_i}{2}$ kijken.

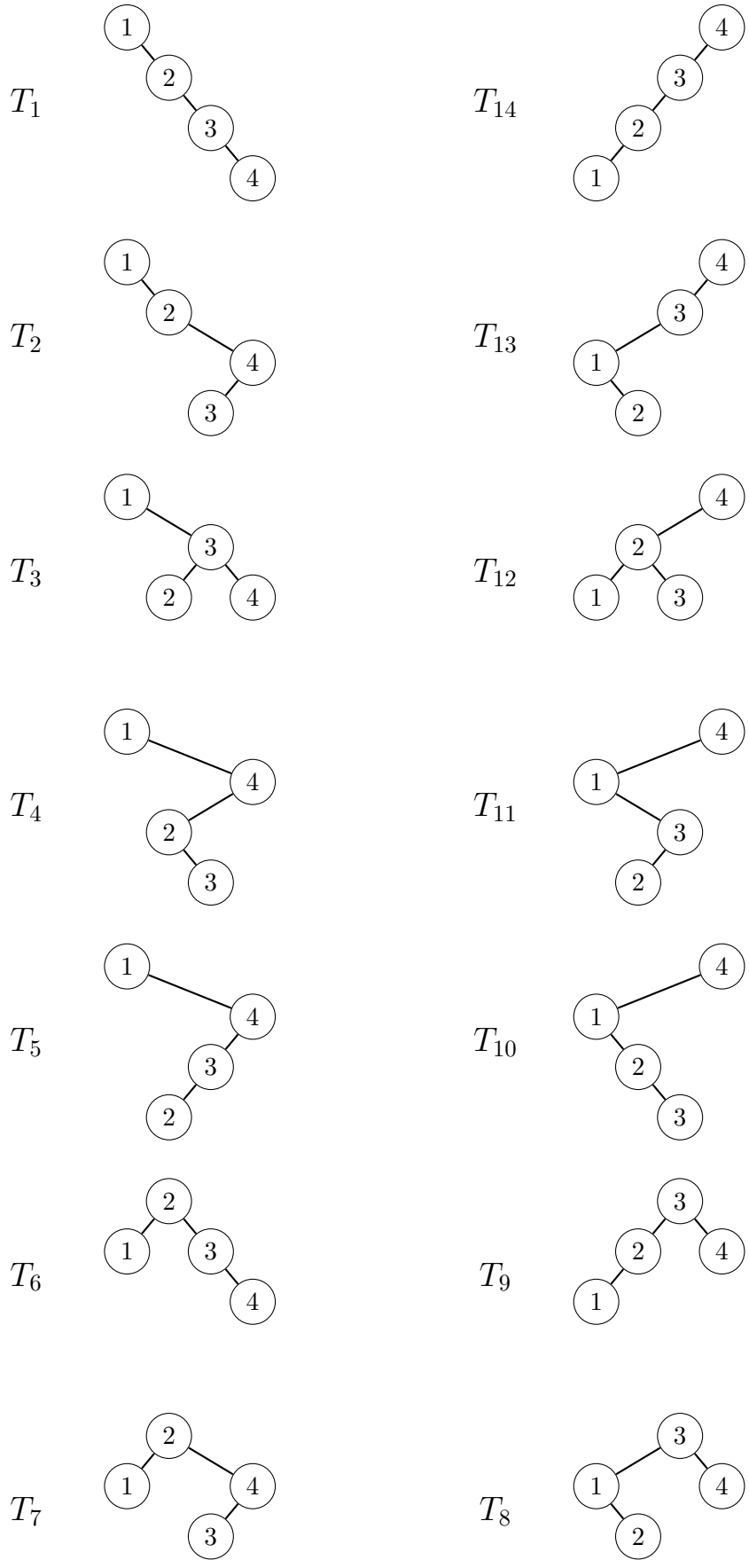
1.13 Dreshers algoritme voor N=4

In deze sectie passen we Dreshers algoritme toe op het hoger lager spel met N=4. We maken ook gebruik van de symmetrie in het spel, waardoor er minder rekenwerk is en we het makkelijker grafisch kunnen weergeven.

Eerst moeten we alle zoekbomen vinden. Ze staan in fig. 1.3. Ze zijn zo gegroepeerd dat ze naast hun symmetrische tegenhanger staan.

Deze zoekbomen leiden tot spelmatrix A:

$$A = \begin{array}{c} \begin{matrix} T_1 & T_2 & T_3 & T_4 & T_5 & T_6 & T_7 & T_8 & T_9 & T_{10} & T_{11} & T_{12} & T_{13} & T_{14} \end{matrix} \\ \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 2 & 2 & 2 & 3 & 2 & 2 & 3 & 3 & 4 \\ 2 & 2 & 3 & 3 & 4 & 1 & 1 & 3 & 2 & 3 & 4 & 2 & 4 & 3 \\ 3 & 4 & 2 & 4 & 3 & 2 & 3 & 1 & 1 & 4 & 3 & 3 & 2 & 2 \\ 4 & 3 & 3 & 2 & 2 & 3 & 2 & 2 & 2 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \end{array} \tag{1.19}$$



Figuur 1.3: Zoekbomen voor N=4

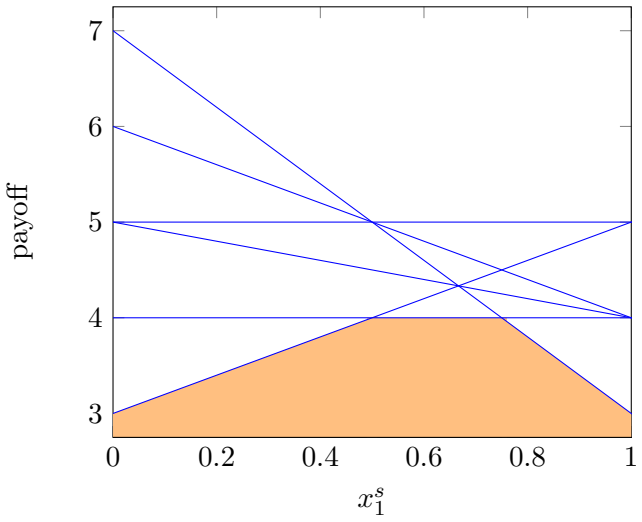
Bob speelt de kolommen en Alice de rijen. Door gebruik te maken van de symmetrische eigenschap van het spel kunnen we zowel het aantal kolommen als het aantal rijen ongestraft halveren. Alice kan bijvoorbeeld optimaal spelen met $x_1 = x_4$ en $x_2 = x_3$. Dit leidt tot de matrix A^s na reductie door symmetrie:

$$A^s = \begin{bmatrix} 5 & 4 & 4 & 3 & 3 & 5 & 4 \\ 5 & 6 & 5 & 7 & 7 & 3 & 4 \end{bmatrix} \tag{1.20}$$

Alles is met 2 vermenigvuldigd voor de leesbaarheid, dus er geldt dat $\mathcal{V}(4) = \frac{1}{2}\mathcal{V}^s(4)$. Hierbij is $\mathcal{V}^s$ de waarde van het spel van A^s. In A^s is elke kolom een lineaire ongelijkheid in de variabelen x_1^s en x_2^s, waarbij $x_1^s = x_1 + x_4$ en $x_2^s = x_2 + x_3$. Voor de tweede kolom geldt bijvoorbeeld:

$$\begin{aligned} 4x_1^s + 6x_2^s &\geq \mathcal{V}^s(4) \\ 4x_1^s + 6(1 - x_1^s) &\geq \mathcal{V}^s(4) \\ -2x_1^s + 6 &\geq \mathcal{V}^s(4) \end{aligned}$$

Bob legt zo met elke zoekboom een maximum op aan de waarde van het spel. We kunnen dit plotten op het domein $[0, 1]$. Als we dat voor elke kolom doen, dan krijgen we:



Het oranje gebied is de verzameling punten die voldoen aan alle ongelijkheden. Zo op het oog zien we dat voor Alice bijvoorbeeld $x_1^s = 0.6$ optimaal is. Alice wil immers maximaliseren en bij $x_1^s = 0.6$ is het minimum van Bob maximaal. Eigenlijk is het hele interval $[0.5, 0.75]$ optimaal voor Alice en zien we dat $\mathcal{V}^s(4) = 4$. We gaan nu bekijken hoe het algoritme van Dresher deze waarde vindt.

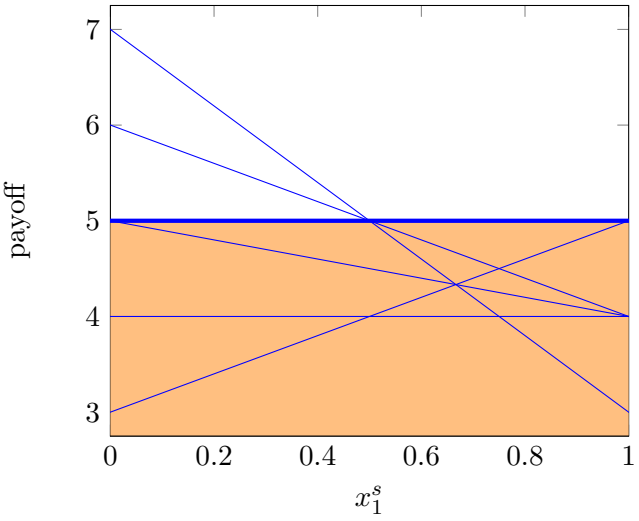
Door A^s te augmenteren verkrijgen we matrix G uit het algoritme van Dresher:

$$G = \begin{array}{c} \begin{matrix} P_0 & P_1 & P_2 & P_3 & P_4 & P_5 & P_6 & P_7 & P_8 & P_9 \end{matrix} \\ \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ -1 & 5 & 4 & 4 & 3 & 3 & 5 & 4 & 1 & 0 \\ -1 & 5 & 6 & 5 & 7 & 7 & 3 & 4 & 0 & 1 \end{bmatrix} \end{array} \tag{1.21}$$

Iteratie 1 De eerste basis B^0 bestaat uit (P_0, P_1, P_8), oftewel

$$B^0 = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 5 & 1 \\ -1 & 5 & 0 \end{bmatrix} \tag{1.22}$$

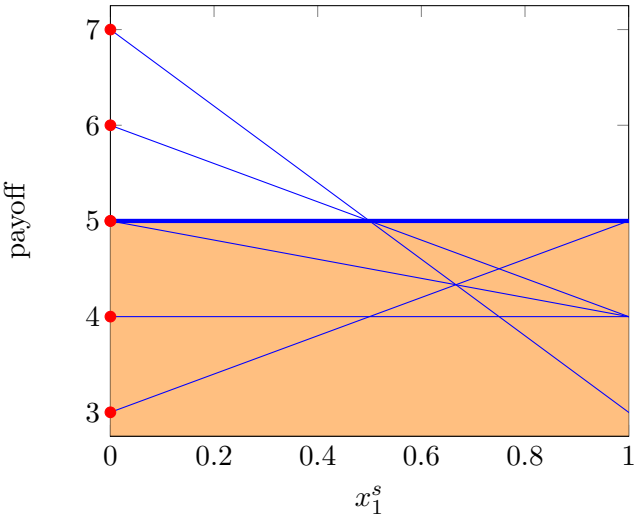
In de grafiek ziet dat er zo uit:



De inverse $(B^0)^{-1}$ is te berekenen met verg. (1.10):

$$(B^0)^{-1} = \begin{bmatrix} 5 & 0 & -1 \\ 1 & 0 & 0 \\ 0 & 1 & -1 \end{bmatrix} = \begin{bmatrix} R_0^0 \\ R_1^0 \\ R_2^0 \end{bmatrix}$$

De waarde van het spel is voorlopig 5, zoals we aan het eerste element van R_0^0 kunnen zien. Dat stemt overeen met de grafiek erboven. Verder zien we in R_0^0 dat $x_1^s = 0$ en $x_2^s = 1$. Nu moeten we het maximum uitrekenen van $R_0^0 P_j$ voor $j = 1, \ldots, 9$. Zo kunnen we het beste tegenspel van Bob tegen deze gemengde strategie van Alice vinden. Zie de rode punten in de grafiek:



$$R_0^0(P_1, \ldots, P_9) = \begin{bmatrix} 0 & -1 & 0 & -2 & -2 & 2 & 1 & 0 & -1 \end{bmatrix}$$

In deze stap wordt dus eigenlijk het verschil genomen tussen de voorlopige waarde van het spel en de waarde van elke zoekboom tegen de huidige gemengde strategie van Alice. Als het verschil positief is, dan is er nog verbetering mogelijk voor Bob. Het maximum zien we bij P_6, $R_0^0 P_6 = 2$. Dus we voegen P_6 toe aan de basis.

Nu moeten we de kolomvector V uitrekenen om te zien welke strategie de basis gaat verlaten.

$$V = (B^0)^{-1} P_6 = \begin{bmatrix} 2 \\ 1 \\ 2 \end{bmatrix}$$

Vervolgens berekenen we het minimum van $\frac{R_i^0}{v_i}$ voor $v_i > 0$ en $i \neq 0$ om te bepalen welke kolom weggaat:

$$\frac{R_1^0}{v_1} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$$
$$\frac{R_2^0}{v_2} = \begin{bmatrix} 0 & \frac{1}{2} & -\frac{1}{2} \end{bmatrix}$$

We dienen dus de laatste kolom van de basis te vervangen, aangezien $\frac{R_1^0}{v_1} \succ \frac{R_2^0}{v_2}$. Dit betekent dat P_6 in de plaats komt van P_8.

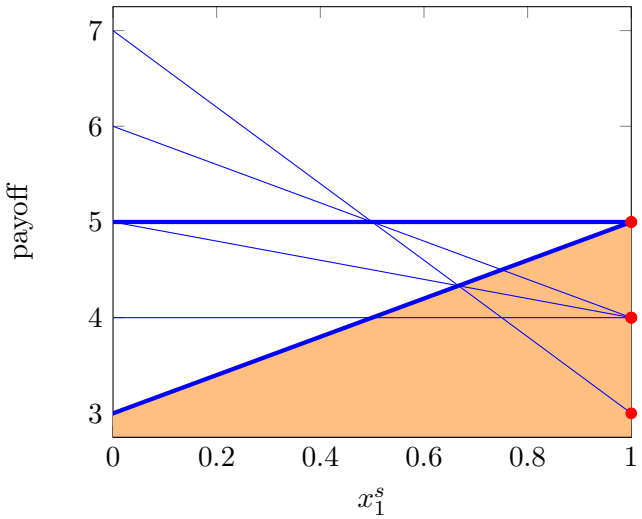
Iteratie 2

$$B^1 = (P_0, P_1, P_6) = \begin{bmatrix} 0 & 1 & 1 \\ -1 & 5 & 5 \\ -1 & 5 & 3 \end{bmatrix}$$

De inverse vinden we nu met de update-regels van Dreshers algoritme:

$$(B^1)^{-1} = \begin{bmatrix} 5 & -1 & 0 \\ 1 & -1/2 & 1/2 \\ 0 & 1/2 & -1/2 \end{bmatrix}$$

In R_0^0 zien we dat de voorlopige waarde van het spel nog steeds precies 5 is. Toch geldt $R_0^0 \succ R_0^1$, omdat de volgende elementen zijn veranderd. Nu geldt dat $x_1^s = 1$ en $x_2^s = 0$.



$$R_0^1(P_1, \ldots, P_9) = \begin{bmatrix} 0 & 1 & 1 & 2 & 2 & 0 & 1 & -1 & 0 \end{bmatrix}$$

Er zijn 2 kolommen die maximaal scoren, namelijk P_4 en P_5. Bij gelijke stand geeft de kleinste index de doorslag, dus P_4 zal de basis betreden.

$$V = (B^0)^{-1} P_4 = \begin{bmatrix} 2 \\ 3 \\ -2 \end{bmatrix}$$

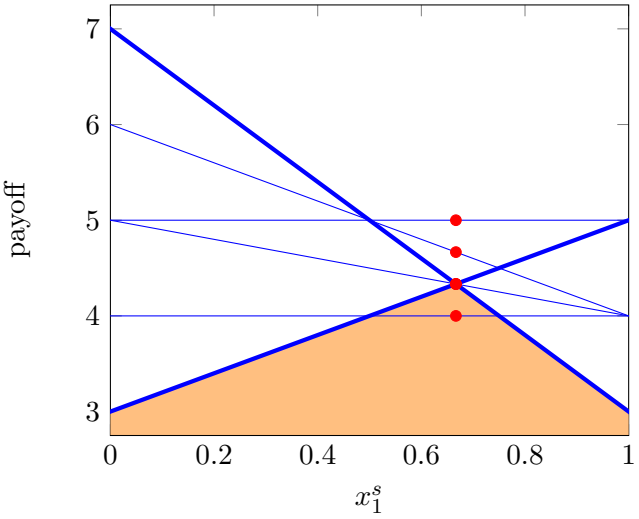
We zien dat $v_2 < 0$, dus we moeten kolom 1 uit de basis verwijderen. P_4 vervangt dan P_1.

Iteratie 3

$$B^2 = (P_0, P_4, P_6) = \begin{bmatrix} 0 & 1 & 1 \\ -1 & 3 & 5 \\ -1 & 7 & 3 \end{bmatrix}$$

$$(B^2)^{-1} = \begin{bmatrix} 13/3 & -2/3 & -1/3 \\ 1/3 & -1/6 & 1/6 \\ 2/3 & 1/6 & -1/6 \end{bmatrix}$$

Nu is de waarde van het spel eindelijk omlaag gegaan naar $\frac{13}{3}$ en is het voor Alice het beste om $x_1^s = \frac{2}{3}$ en $x_2^s = \frac{1}{3}$ te spelen. Wat kan Bob daar tegenin brengen?



$$R_0^2(P_1, \ldots, P_9) = \begin{bmatrix} -2/3 & -1/3 & 0 & 0 & 0 & 0 & 1/3 & -2/3 & -1/3 \end{bmatrix}$$

We zien dat er maar 1 kolom verbetering voor Bob kan brengen, aangezien alleen P_7 tot een positief getal leidt. P_7 gaat dus de basis in.

$$V = (B^0)^{-1} P_7 = \begin{bmatrix} 1/3 \\ 1/3 \\ 2/3 \end{bmatrix}$$

$$\frac{R_1^2}{v_1} = \begin{bmatrix} 1 & -\frac{1}{2} & \frac{1}{2} \end{bmatrix}$$

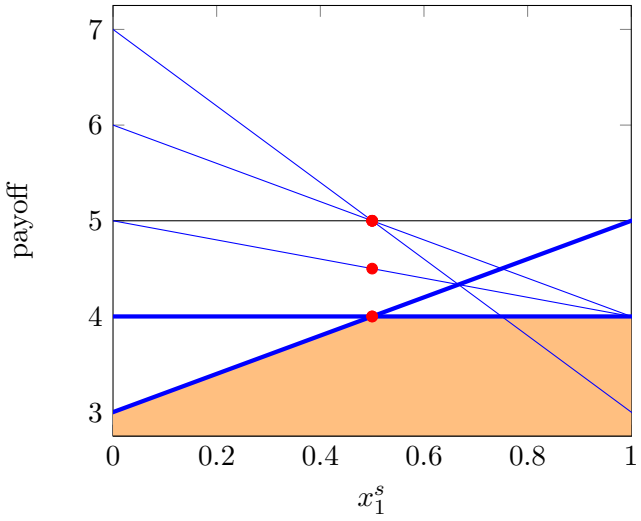
$$\frac{R_2^2}{v_2} = \begin{bmatrix} 1 & \frac{1}{4} & -\frac{1}{4} \end{bmatrix}$$

$R_1^2 \prec R_2^2$, dus kolom 1 moet alweer uit de basis. P_7 vervangt dus P_4.

Iteratie 4

$$B^3 = (P_0, P_7, P_6) = \begin{bmatrix} 0 & 1 & 1 \\ -1 & 4 & 5 \\ -1 & 4 & 3 \end{bmatrix}$$

$$(B^3)^{-1} = \begin{bmatrix} 4 & -1/2 & -1/2 \\ 1 & -1/2 & 1/2 \\ 0 & 1/2 & -1/2 \end{bmatrix}$$



$$R_0^3(P_1, \ldots, P_9) = \begin{bmatrix} -1 & -1 & -1/2 & -1 & -1 & 0 & 0 & -1/2 & -1/2 \end{bmatrix}$$

Geen enkel element is positief, dus hier stopt het algoritme. Bob kan niet beter dan dit en dat is ook in de grafiek te zien. De optimale tactieken voor beide spelers zijn nu uit de inverse af te lezen.

Oplossing Voor het spel met A^s speelt Alice $x_1^s = x_2^s = 0.5$ en Bob speelt altijd P_7. Voor de waarde van dit spel geldt dan $\mathcal{V}^s(4) = 4$. Voor het oorspronkelijke spel geldt nu:

$$\begin{aligned} \mathcal{V}(4) &= 2 \\ X^* &= \begin{bmatrix} 1/4 & 1/4 & 1/4 & 1/4 \end{bmatrix} \\ Y^* &= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1/2 & 1/2 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{aligned}$$

Hoofdstuk 2

Efficiënt de beste zoekboom vinden

2.1 Delayed column generation

In elke iteratie van de simplexmethode is er een stap, waarin we de beste kolom zoeken om toe te voegen aan de basis. Dat betekent dat we een zoekboom moeten zoeken, die optimaal is tegen de gemengde strategie van Alice volgens de huidige basis. En optimaal voor Bob is een zoekboom waarvoor het inproduct met de strategie van Alice minimaal is. Hiervoor gaan we in de standaard simplexmethode alle kolommen af, maar omdat dat aantal gelijk is aan Catalan(N) duurt dat lang. Daarom vervang ik die stap door een sneller algoritme dat in een polynomiaal aantal stappen werkt.
De input van het algoritme is dus de gemengde strategie van Alice over haar N pure strategiëen. Deze gemengde strategie verandert niet tijdens de uitvoer van het algoritme. Pas na het updaten van de basis kan de strategie van Alice weer veranderen.
De output van het algoritme is de waarde van een zoekboom van Bob die optimaal is tegen die specifieke gemengde strategie.
Ik presenteer twee subroutines. De eerste heeft tijdscomplexiteit $\mathcal{O}(N^3)$. De tweede heeft tijdscomplexiteit $\mathcal{O}(N^2)$, maar vindt niet gegarandeerd in elke situatie de optimale oplossing. Ook is de constante een stuk hoger.

2.2 Beste gok subroutine

Stelling 2. *Voor een gegeven gemengde strategie α van Alice kan Bob een zoekboom met minimale payoff tegen α vinden in $\mathcal{O}(N^3)$ iteraties.*

Het algoritme werkt als volgt. We beschouwen $E[a,b]$ met $a,b \in [1,N]$ als het minimaal verwachte aantal keer raden van Bob, wanneer hij na een aantal keer raden het zoekdomein tot het interval $[a,b]$ heeft teruggebracht. $E[1,N]$ is dus de verwachting van het aantal keer raden van de beste zoekboom van Bob tegen de huidige gemengde strategie van Alice. $E[a,a] = 1$ en $E[a,b] = 0$ als $a > b$ zijn simpele randgevallen. We willen $E[1,N]$ minimaliseren en daarvoor kunnen we de volgende recurrente betrekking gebruiken:

$$E[a,b] = \min_{a \le i \le b} p(a,i-1|a,b) \cdot E[a,i-1] + p(i,i|a,b) \cdot E[i,i] + p(i+1,b|a,b) \cdot E[i+1,b] \quad (2.1)$$

waarbij $p(c,d|a,b)$ de kans is dat het geheime getal zich in interval $[c,d]$ bevindt, gegeven dat het getal in interval $[a,b]$ zit.
Merk op dat als we $E[1,N]$ naïef uitrekenen, we veel subproblemen meerdere keren berekenen. Als we elke keer dat we iets hebben uitgerekend de uitkomst in een tabel schrijven en elke

Algorithm 2 Bereken de minimale payoff van een boom tegen gegeven α

Creëer een $N * N$ matrix W
$W(i,j) \leftarrow \begin{cases} 0, & \text{if } i = j \\ \infty, & \text{otherwise} \end{cases}$
Creëer een $N * N$ matrix SUM
SUM(i, j) $\leftarrow \sum_{k=i}^{j} \alpha_k$
for $len = 1$ **to** N **do**
  for $first = 1$ **to** $N + 1 - len$ **do**
    $last \leftarrow first + len$
    $W(len, start) \leftarrow min_i\{W(len, i-1) + W(i+1, last) + \text{SUM}(first, last)\}$
    for $i = start$ to $last$ **do**
      $W(len, start) \leftarrow \min\{W(len, start), W(len, i-1) + W(i+1, last) + \text{SUM}(first, last)\}$
    end for
  end for
end for
return $W(1, N)$

keer voordat we iets gaan uitrekenen in de tabel kijken of het al is uitgerekend, dan wordt geen enkel subprobleem meerdere keren berekend. Deze methode wordt ook wel dynamisch programmeren genoemd. Er zijn $\mathcal{O}(N^2)$ subproblemen, omdat er zoveel subintervallen van $[1, N]$ zijn. Per subprobleem worden er $\mathcal{O}(N)$ berekeningen gedaan, dus het totaal aantal iteraties om de minimale payoff te vinden is $\mathcal{O}(N^3)$. De bijbehorende zoekboom kan in $\mathcal{O}(N^2)$ stappen uit de tabel worden gehaald.

2.3 Beste deelrij subroutine

Er bestaat nog een andere manier om een goede zoekboom tegen een specifieke gemengde strategie van Alice te vinden in korte tijd. In de volgende methode wordt een optimale zoekboom van maximale hoogte h gevonden in $\mathcal{O}(N \cdot 2^h)$ iteraties. Dit is traag, als h groot genoeg is om elke zoekboom te overwegen. Maar als we h beperken tot $\lceil lg(N) \rceil$, dan krijgen we ineens een algoritme van orde $\mathcal{O}(N^2)$. We vinden dan een redelijk gebalanceerde zoekboom en in de praktijk blijken we met deze maximale hoogte optimale strategieën voor Bob te kunnen vinden. De optimaliteit van Bobs gehele oplossing is achteraf te controleren met de eerder genoemde, tragere, subroutine.

2.3.1 Definities

In het volgende stuk komt terminologie voor over bomen. Laten we de gebruikelijke definities aanhouden:

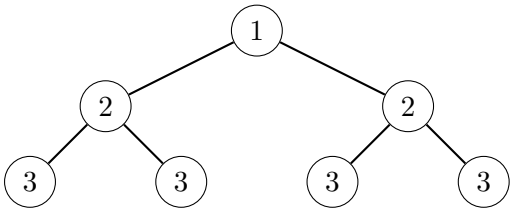
Definitie 12. *De hoogte van een knoop in een boom is het aantal takken op het langste pad tussen die knoop en een blad.*

Definitie 13. *De hoogte van een boom is de hoogte van de wortel van de boom.*

Definitie 14. *De diepte van een knoop is het aantal takken van de wortel tot die knoop.*

Laat F_h een gewortelde, binaire boom van hoogte $h-1$ zijn met een maximaal aantal knopen. De hoogte van de bladeren is dus 0 en de hoogte van de wortel is $h - 1$. Er zijn $2^h - 1$ knopen in de boom. Laat $F_h(i)$ het i^{de} element zijn van de boom in infixnotatie. De waarde van een

knoop is de diepte van de knoop plus 1, omdat dit het aantal keer raden is. $F_3(6)$ is dan gelijk aan 2 bijvoorbeeld, zie hieronder.



$F_3 = [3, 2, 3, 1, 3, 2, 3]$

Merk op dat elke zoekboom met hoogte $\leq h - 1$ een subboom is van F_h. We zoeken dus een deelrij van F_h van lengte N. Elke valide strategie is een deelrij, maar niet elke deelrij is een valide strategie voor Bob. Als in een deelrij bijvoorbeeld het element 1 ontbreekt, kan het geen valide strategie representeren. We bekijken later of het algoritme mogelijk illegale bomen oplevert.
De gemengde strategie van Alice noteren we met $\alpha = \{\alpha_1, \ldots, \alpha_N\}$. De waarde van een zoekboom is het inproduct van α met de deelrij van F_h die hoort bij die zoekboom. We zoeken dus de deelrij die dit inproduct minimaliseert.

Laat $d = \{d_1, \ldots, d_N\}$ een deelrij zijn van F_h. Er geldt $d_1 \geq 1$, $d_N \leq |F_h|$ en $d_i < d_{i+1}$. Beschouw nu de functie $W(i, j)$, die de minimum verwachte waarde weergeeft van de eerste i kansen van Alice optimaal verdeeld over de eerste j elementen van F_h. Dat is dus het minimum inproduct van $\{\alpha_1, \ldots, \alpha_i\}$ met $\{d_1, \ldots, d_i\}$, waarbij $d_i \leq j$. $W(N, |F_h|)$ is dus de waarde van de optimale zoekboom tegen α.

2.3.2 Dynamisch programmeren

De waarde van $W(N, |F_h|)$ is nu uit te rekenen met behulp van dynamisch programmeren. We stellen een tabel op voor alle mogelijke $W(i, j)$. Als we de tabel hebben ingevuld met behulp van een recursieve formule, dan kunnen we daaruit de beste zoekboom van Bob halen.
Voor de recursieve formule is het belangrijk om in te zien dat $W(i, j-1) \geq W(i, j)$. Elke deelrij d van i elementen met $d_i \leq j - 1$ is immers ook een deelrij met $d_i \leq j$. De enige mogelijkheid voor $W(i, j-1) > W(i, j)$ is dat het j^{de} element als i^{de} wordt gekozen. Alice speelt die strategie met kans α_i en Bob raadt het dan in $F_h(j)$ keer. De eerste $i - 1$ elementen moeten in dat geval al voor het j^{de} element gekozen zijn en de beste manier daarvoor is $W(i-1, j-1)$. Nu kunnen we de volgende recursieve formule opstellen:

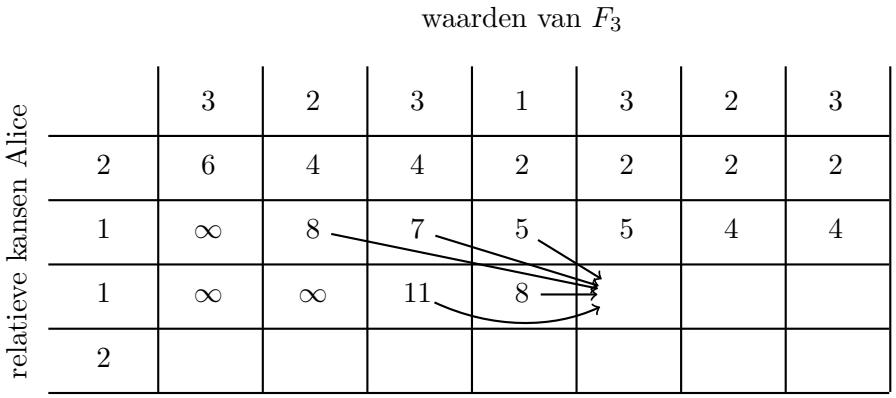
$$W(i, j) = \min\{W(i, j-1), W(i-1, j-1) + \alpha_i \cdot F_h(j)\} \tag{2.2}$$

$$W(i, j) = W(i, j-1) \wedge (W(i-1, j-1) + \alpha_i \cdot F_h(j)) \tag{2.3}$$

De basisgevallen zijn eenvoudig te bepalen:

$$\begin{aligned} \forall j < i : W(i, j) &= \infty \\ W(1, j) &= \min\{W(1, j-1), \alpha_1 \cdot F_h(j)\} \\ W(1, 1) &= \alpha_1 * F_h(1) \end{aligned}$$

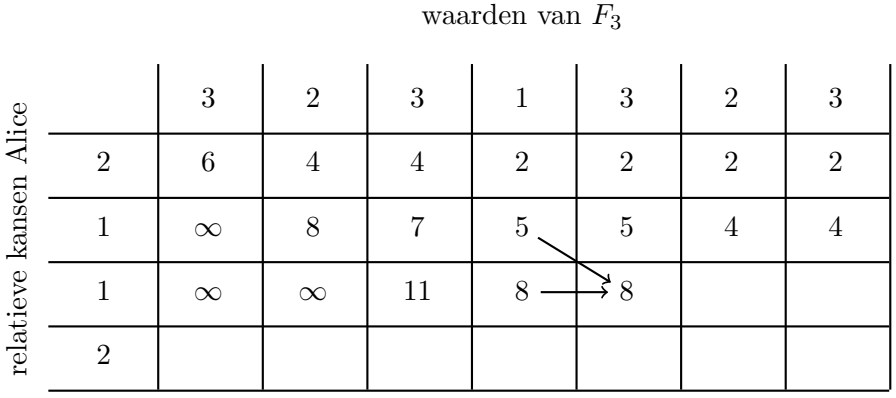
Laten we een voorbeeld bekijken. Neem $N = 4$ en $h = 3$. De tabel wordt rij voor rij ingevuld en we dienen W(3,5) uit te rekenen. Zie tabel 2.1. In de header staan de elementen van F_3 en in de linkerkolom staan de relatieve kansen waarmee Alice haar strategieën speelt.



	3	2	3	1	3	2	3
2	6	4	4	2	2	2	2
1	∞	8	7	5	5	4	4
1	∞	∞	11	8			
2							

Tabel 2.1: functie W voor $N = 4$

De pijlen in de figuur geven aan welke mogelijkheden er zijn om op $W(3, 5)$. Langs de pijlen uit dezelfde rij hoeven we niets op te tellen, omdat het derde element dan al geplaatst is. Langs de pijlen van een rij erboven moeten we nog $\alpha_3 F_h(5)$ optellen, omdat we het derde getal van Alice op plaats 5 zetten.
Het zou echter naïef zijn om al deze pijlen te beschouwen. Merk op dat in elke rij de getallen dalend zijn. Logisch, want $W(i, j-1) \leq W(i, j)$. Dus van beide rijen hoeven we eigenlijk alleen de meest rechter te beschouwen die nog op $W(3, 5)$ uitkomt. Wat we langs de pijlen optellen is immers constant voor elke rij. Er zijn dus voor elke cel hooguit 2 pijlen nodig. Zie tabel 2.2.



	3	2	3	1	3	2	3
2	6	4	4	2	2	2	2
1	∞	8	7	5	5	4	4
1	∞	∞	11	8	8		
2							

Tabel 2.2: functie W voor $N = 4$

Op deze wijze is de tabel efficient rij voor rij in te vullen vanaf $W(1, 1)$ t/m $W(N, |F_h|)$. In totaal worden $\mathcal{O}(N^2)$ cellen ingevuld en er worden voor elke cel twee termen vergeleken, dus is de tijdscomplexiteit $\mathcal{O}(N^2)$.

2.3.3 Optimale zoekboom extraheren

Als de tabel is ingevuld, dan kunnen we direct de minimale payoff aflezen in $W(N, |F_h|)$. Een optimale boom hebben we dan nog niet, maar die kunnen we vinden door in de tabel te starten bij $W(N, |F_h|)$. Van daaruit kunnen we met behulp van de recursieve formule richting de eerste

Algorithm 3 Bereken de minimale payoff van een gebalanceerde boom tegen gegeven α

$h \leftarrow 1+$ minimumdiepte boom met N knopen
$m \leftarrow 2^h - 1$
Creëer een $|F_h| * N$ matrix W

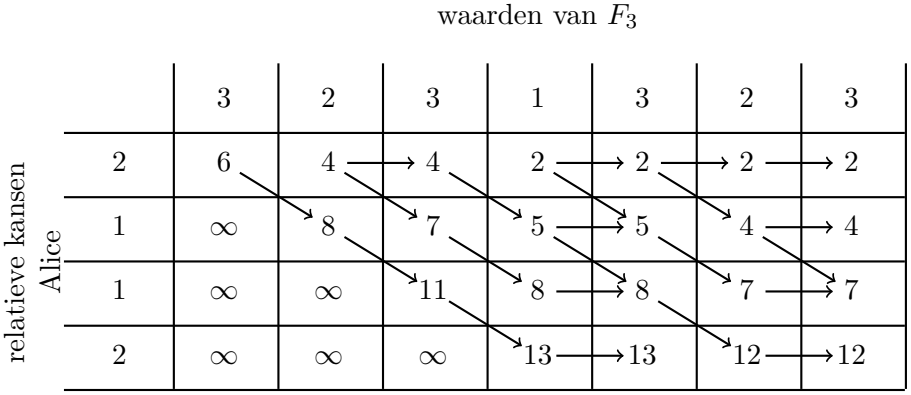
$$W(i,j) \leftarrow \begin{cases} \alpha_1 * F_h(1), & \text{if } i = j = 1 \\ \min\{W(1, j-1), \alpha_1 \cdot F_h(j)\}, & \text{if } i = 1\,, j > 1 \\ \infty, & \text{otherwise} \end{cases}$$

for $i = 1$ **to** N **do**
  for $j = 1$ **to** m **do**
    $W(i,j) = \min\{W(i, j-1), W(i-1, j-1) + \alpha_i \cdot F_h(j)\}$
  end for
end for
return $W(m, N)$

rij werken. Als we op plaats $W(i,j)$ zijn en er geldt $W(i,j-1) = W(i,j)$, dan zoeken we verder vanaf $W(i,j-1)$. Anders voegen we $F_h(j)$ op de i^{de} plaats toe in de deelrij en zoeken we verder vanaf $W(i-1,j-1)$. De stopconditie is $i < 1$ en als dat bereikt is, dan is de deelrij ter lengte N ook ingevuld. Deze deelrij is de zoekboom in infixnotatie. Dit duurt slechts $\mathcal{O}(N)$ stappen.

2.3.4 Voorbeeld

Gegeven de volledig ingevulde tabel van het eerdere voorbeeld:



Tabel 2.3: functie W voor $N = 4$

Elke inkomende pijl geeft aan welke cellen verantwoordelijk zijn voor het minimum voor die cel. Soms zijn er twee inkomende pijlen, omdat de twee te vergelijken waarden gelijk aan elkaar zijn.
We starten rechtsonder en bewandelen de pijlen in omgekeerde richting naar boven. Als we alle mogelijke paden naar de eerste rij bekijken, dan blijven er twee paden over:

waarden van F_3

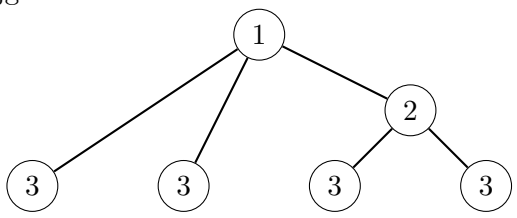
relatieve kansen Alice	3	2	3	1	3	2	3
2	6	4	4	2	2	2	2
1	∞	8	7	5	5	4	4
1	∞	∞	11	8	8	7	7
2	∞	∞	∞	13	13	12	12

Tabel 2.4: functie W voor $N = 4$

Deze twee paden corresponderen met deelrijen $\{2, 3, 1, 2\}$ en $\{2, 1, 3, 2\}$. De bijbehorende bomen zijn optimaal tegen de relatieve kansen $\alpha = \{2, 1, 1, 2\}$.

2.3.5 Niet valide bomen

Sommige deelrijen zijn niet in een valide boom om te zetten. Bijvoorbeeld: $[3, 3, 1, 3, 2, 3]$ We krijgen dan zo'n boom:



Deze is niet binair en na een 1 kun je niet op een 3 uitkomen. Zolang geldt dat $\forall i \in [1, N] : \alpha_i > 0$, is er echter niets aan de hand. We zoeken de deelrij met het minimale inproduct en dan is $[3, 2, 1, 3, 2, 3]$ beter.
Tijdens het uitvoeren van de simplexmethode kunnen de kansen van Alice ook negatief worden. Dan zouden zulke verkeerde bomen wel als huidig beste geselecteerd kunnen worden. Dit geeft echter geen problemen voor de juistheid van het eindresultaat. Mocht Bob een illegale boom gebruiken, dan komt hij niet op een optimum uit en gaat het simplexalgoritme door. Op het eind van het algoritme wordt ook gecheckt of alle strategieën van Bob valide zijn en dat klopt voor alle gevonden oplossingen.

2.4 Sneller convergeren

De vorige paragrafen gingen over het versnellen van het vinden van een zoekboom om aan de basis toe te voegen. We kunnen daarnaast ook het aantal updates van de basis proberen terug te dringen door sneller naar een optimale (gemengde) strategie van Alice te convergeren.
De kansverdeling van Alice gaat flink heen en weer bij updates van de basis. Bob zoekt steeds een optimale zoekboom tegen de huidige kansverdeling van Alice. Deze zoekboom hoeft niet optimaal te zijn tegen een uiteindelijke optimale strategie van Alice. Sterker nog, vaak vindt Bob een extreme boom die de zwakke plekken van Alice helemaal uitbuit ten koste van vaak raden voor getallen, die Alice toch met kleine kans speelt. Dit laatste creëert weer een zwakke plek voor Bob, waar Alice gebruik van maakt na het updaten van de basis. Hierdoor worden veel

relatief kleine kansen groot en omgekeerd. Dit oscilleren resulteert in een vrij trage convergentie naar een optimum.
Eigenlijk zouden we zoekbomen willen vinden die optimaal zijn tegen een optimale strategie van Alice en die het ook goed doen tegen haar huidige strategie. We weten nog niet wat voor strategie voor Alice optimaal is, maar we kunnen dat wel gokken. Uit de resultaten blijkt dat de gevonden optimale strategie van Alice voor N vaak sterk lijkt op die voor kleinere instanties van het spel. We kunnen dus best een goede gok γ doen voor een uiteindelijke optimale strategie van Alice. α is de huidige strategie van Alice, waartegen Bob normaal gesproken de beste zoekboom zoekt. In plaats daarvan zoekt Bob nu de beste zoekboom tegen het gewogen gemiddelde $c\gamma + (1-c)\alpha$, waarbij c een variabele is in de buurt van 1.
Door deze keuze reageert Bob nog steeds op de zwakke plekken van Alice, maar overdrijft daarin niet. Hierdoor convergeren beide spelers sneller naar een optimum en wordt het aantal iteraties fors vermindert.
Het maakt wel uit hoe goed onze gok is. Als N niet in de buurt van een tweemacht zit, dan klopt onze gok precies (voor $N \leq 980$) en is het aantal iteraties zeer beperkt. In de buurt van tweemachten klopt onze gok niet precies. Dan zoekt Bob toch met een gewogen gemiddelde, maar laat hij de variabele c naar nul gaan. Hierdoor wordt toch een optimum bereikt en het aantal iteraties flink naar beneden gebracht, maar veel minder dan bij een gok die precies klopt.

Hoofdstuk 3

Resultaten

3.1 Correctheid resultaten

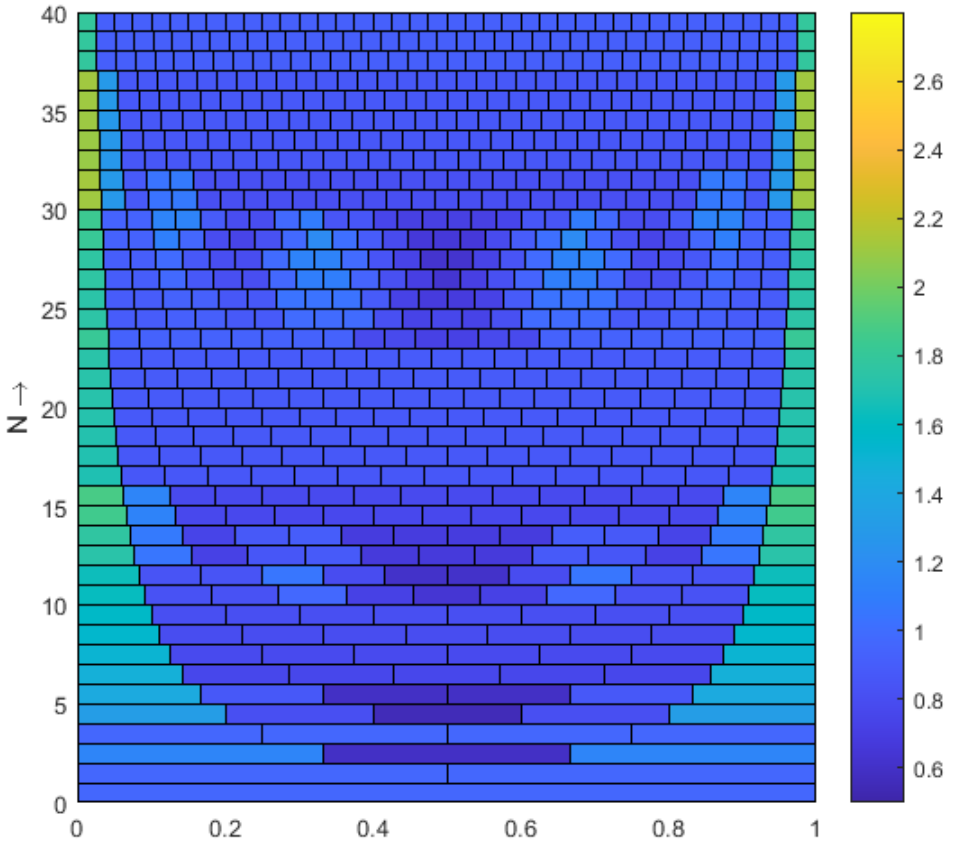
De complexiteit van het algoritme is door de aanpassingen toegenomen en ook in de implementatie kunnen er bugs zitten. Verder reken ik met floating point getallen. Het is dus belangrijk om te checken of de gevonden resultaten kloppen. Daarvoor reken ik eerst opnieuw de inverse van de basis uit, maar dan met breuken in plaats van floating point getallen. Hieruit volgen strategieën voor Alice en Bob. Deze check ik tegen verg. (1.6). Ook test ik of elke voorwaarde correspondeert met een valide boom. Dit zijn allemaal relatief simpele tests parallel aan elkaar, waardoor we vertrouwen mogen koesteren dat alles dat door deze test komt, correct is.
Niet alle gevonden strategieën bleken echter correct. Voor $N = 485$, $N = 528$ en $N = 530$ waren de gevonden strategieën net niet optimaal. Dit heeft met de precisie van de floating point getallen en de wijze van berekening te maken. Ook convergeert het zo traag en met zulke kleine stapjes, dat kleine onnauwkeurigheden al fataal kunnen zijn. Inspanningen om de methode accurater te maken leverden niet genoeg op.

3.2 Optimale strategie Alice

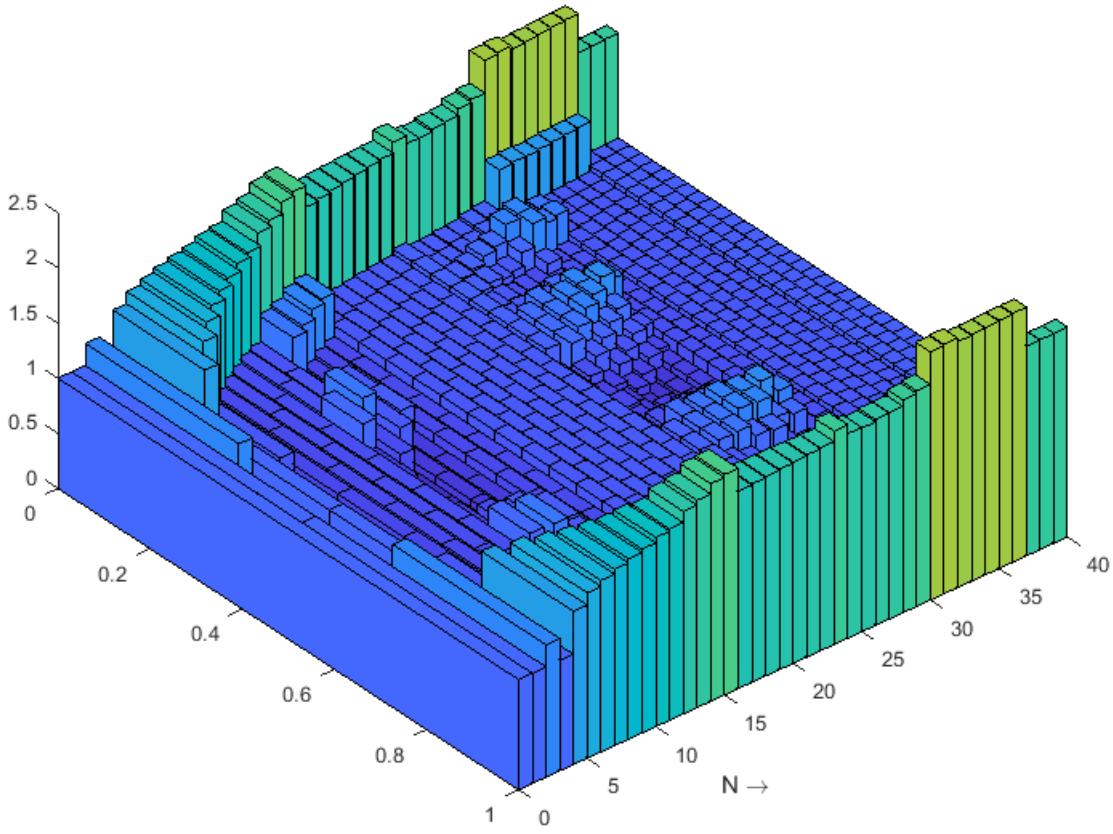
Het is niet gelukt om de optimale strategie van Alice als een functie van N te schrijven, maar door de resultaten uit te breiden tot $N = 980$ zijn er patronen zichtbaar geworden die in voorgaande resultaten nog niet of nauwelijks waarneembaar waren. De optimale strategie van Alice is in fig. 3.1 voor $N <= 40$ visueel weergegeven. Elke rij is verdeeld in N stukken, die de strategieën kies 1 t/m kies N tonen. De kleur van het i^{de} stuk geeft de relatieve kans, waarmee Alice getal i kiest als geheim getal. De grootte van de kansen is ook in 3D te tonen zoals in fig. 3.2, dat precies dezelfde data toont. Zo is het snel duidelijk dat de buitenste strategieën het vaakst worden gekozen en de kansen verder redelijk gespreid worden.

We zien echter ook nog wat hobbels her en der. Het patroon daarvan wordt duidelijker als we de figuur uitbreiden t/m $N = 300$, zoals in fig. 3.3. Daar zien we dat rond de tweemachten de kansverdeling golvend is en dat tussen de tweemachten de kansen netjes gespreid zijn.

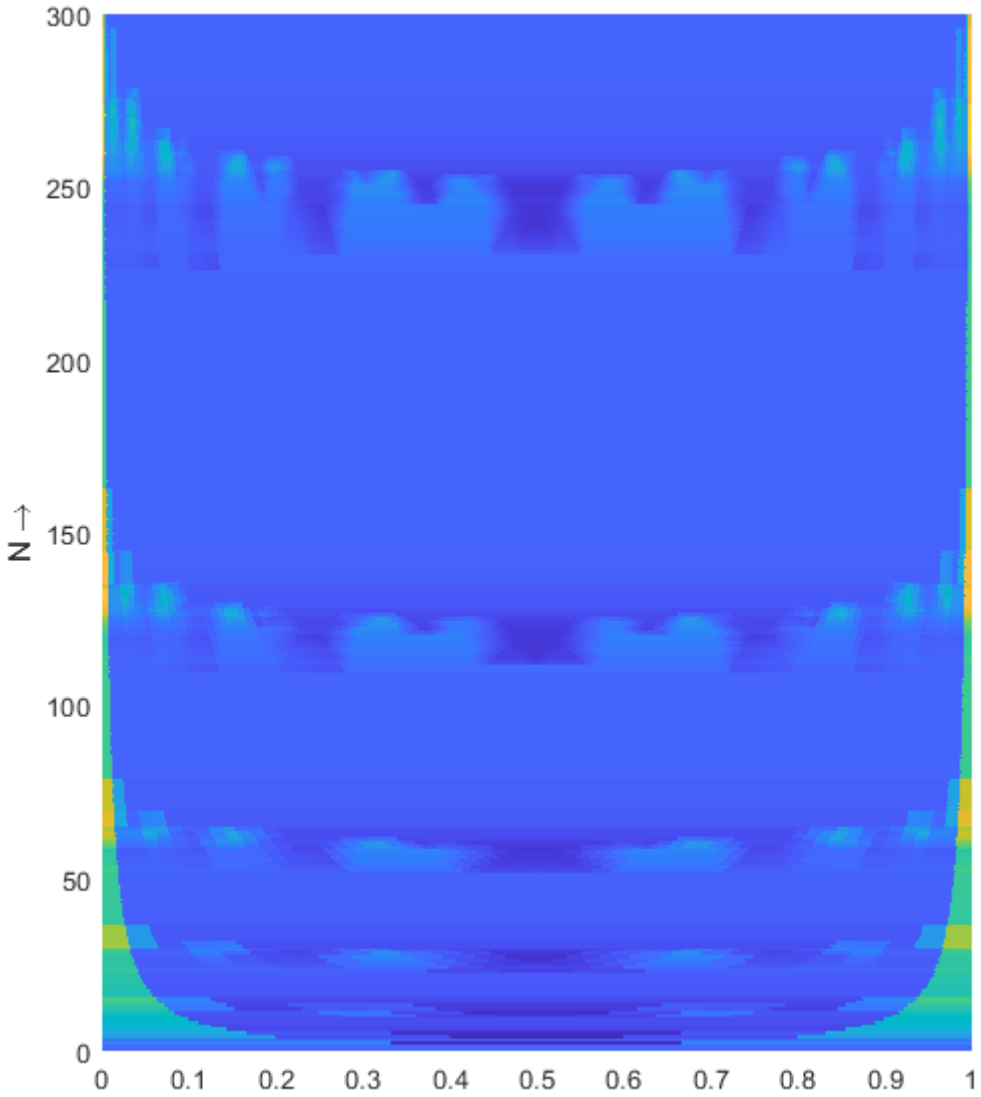
In figuur fig. 3.4 is een optimale strategie van Alice te zien voor $N = 253$.



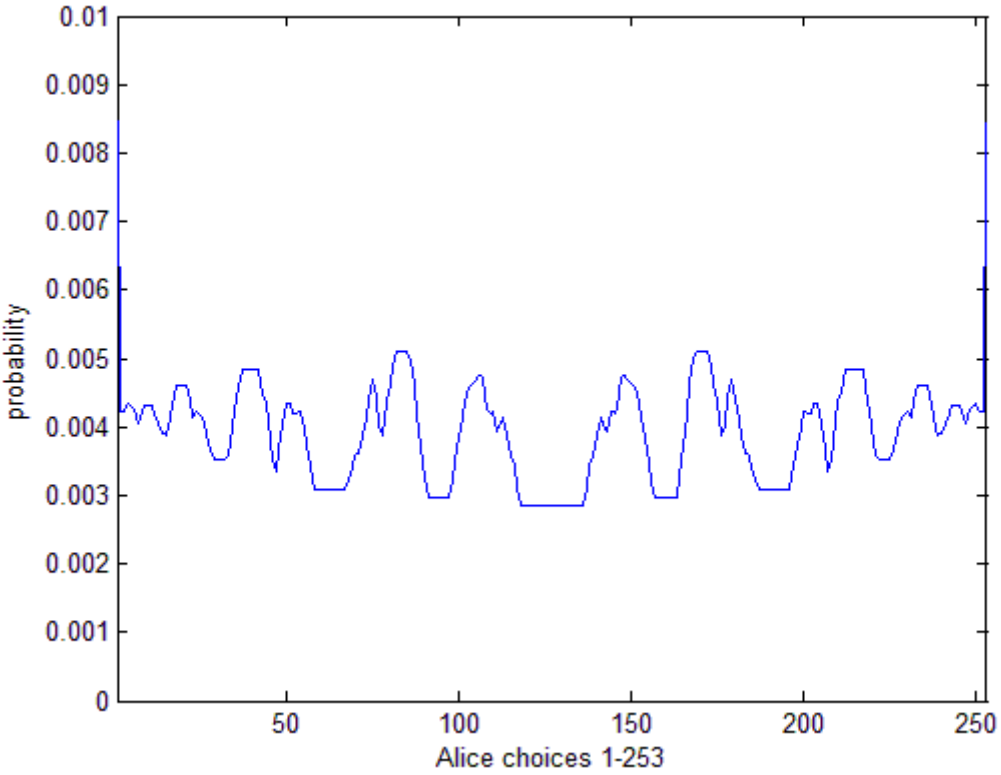
Figuur 3.1: Distributie kansen Alice voor $N \leq 40$



Figuur 3.2: Distributie kansen Alice voor $N \leq 40$



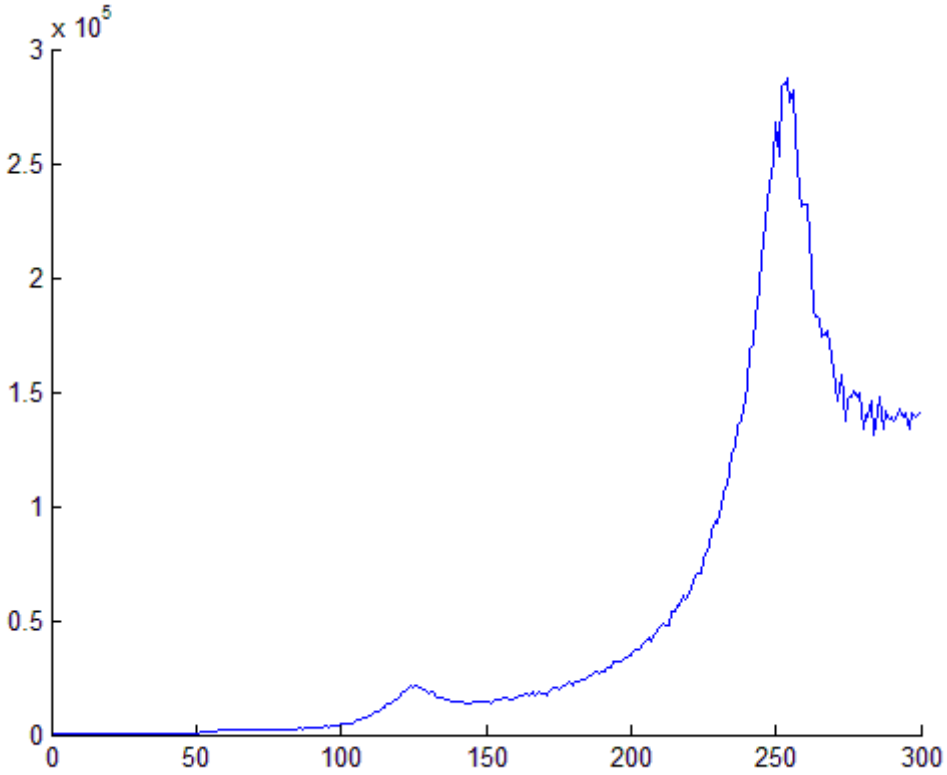
Figuur 3.3: Distributie kansen Alice voor $N \leq 300$



Figuur 3.4: Distributie kansen Alice voor $N = 253$

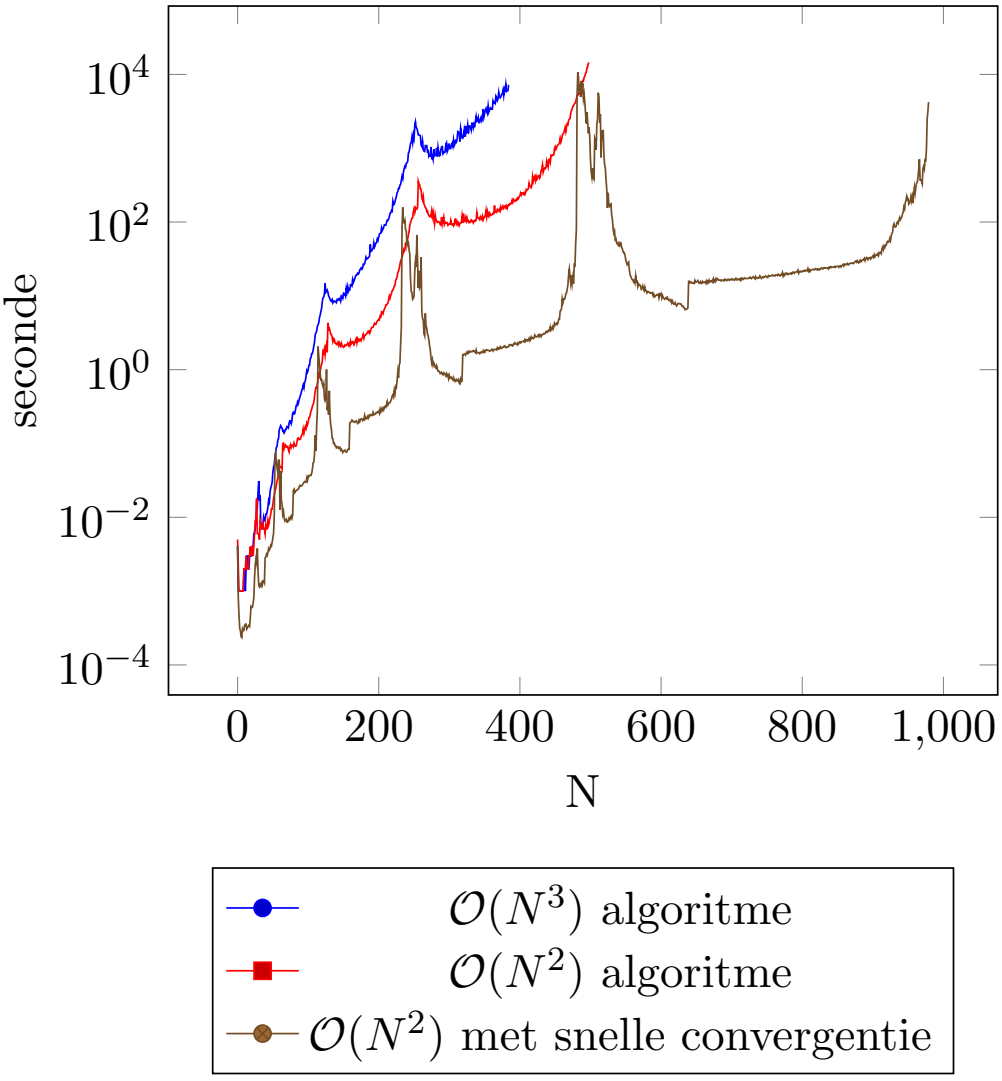
3.3 Berekeningen

Het aantal keer dat een strategie aan de basis wordt toegevoegd, steeg met name, als N dicht in de buurt van een macht van 2 zat. Dit is te zien in fig. 3.5.

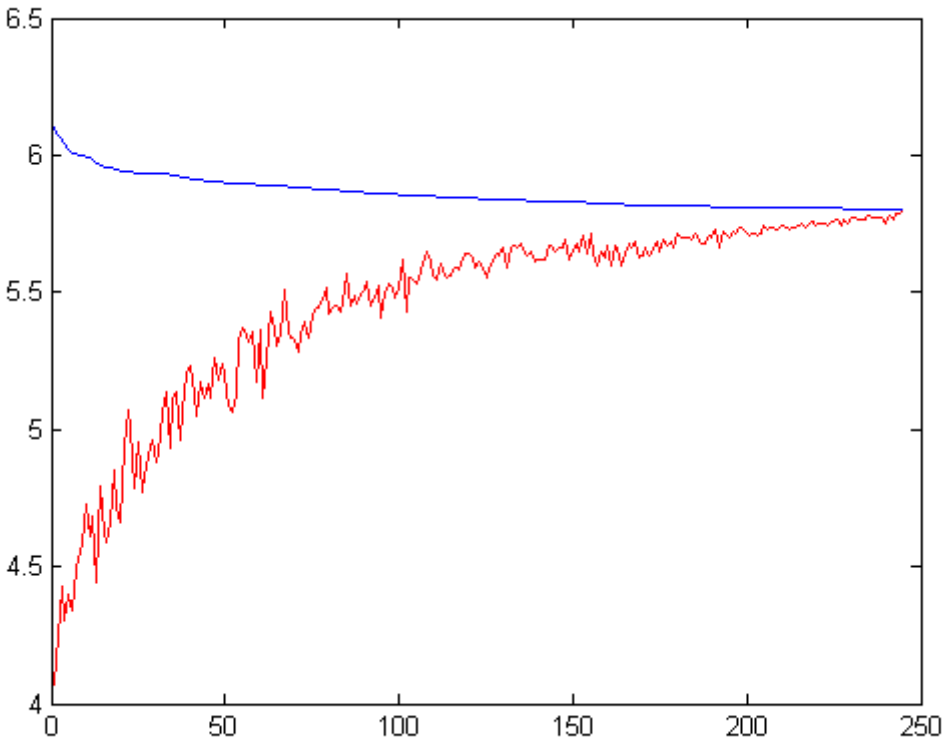


Figuur 3.5: Aantal simplexiteraties

Het uiteindelijke algoritme, waarbij de optimale kansverdeling van Alice wordt ingeschat, heeft veel minder simplexiteraties, met name tussen machten van 2, en dus ook minder runtijd. Een vergelijking tussen de runtijden van de verschillende algoritmen is te zien in paragraaf 3.3. Alle berekeningen zijn op dezelfde standaardcomputer uitgevoerd.



Elke keer dat een strategie de basis binnenkomt, wordt er een nieuwe bovengrens gevonden voor de waarde van v. Het algoritme stopt immers, zodra er geen betere zoekboom meer is voor Bob tegen de huidige gemengde strategie van Alice. In fig. 3.6 geeft de blauwe lijn de bovengrens aan tijdens de berekening voor $N = 100$ en te zien is dat die inderdaad monotoon daalt. De rode lijn geeft de waarde weer van de gevonden zoekboom, die het best werkt tegen de huidige gemengde strategie van Alice. Deze waarde hoeft niet monotoon stijgend te zijn, zoals te zien is.



Figuur 3.6: Boven- en ondergrens tijdens berekening voor $N = 100$

Tabel 3.1: Maximale zoekdiepte

Maximaal aantal keer raden	N
1	1
2	2
3	3-4
4	5-10
5	11-23
6	24-52
7	53-110
8	111-226
9	227-460
10	461-930
11	931-980

Het $\mathcal{O}(N^2)$ algoritme werkt, omdat de maximale zoekdiepte niet zo groot hoeft te zijn. In tabel 3.1 staan de maximale zoekdiepten van de gevonden optimale strategieën van Bob.

De waarde van het spel is een breuk die zeker rond machten van 2 grote getallen kan bevatten. Dit is bijvoorbeeld het verwachte aantal keer raden bij optimaal spel voor $N = 509$:

$$\mathcal{V}(509) = \frac{2801502958915667971701898342801970475106864163534168947198342637184804598745936500730055179424539352778}{34897267295548485623365831503799734637574246131704453513298969288670195708397151020029920953210697 1303}$$

3.4 Conclusie

Een simpele functie vinden voor de optimale strategieën voor beide spelers lijkt onhaalbaar. Wel is meer over de vermoedens van Gilbert en Johnson te zeggen. De volgende patronen gaven Johnson vermoedens over de optimale kansverdeling van Alice:

1. Voor $5 \leq N \leq 11$ geldt: $\alpha_1 = \alpha_2 + \alpha_3$
2. Voor $7 \leq N \leq 11$ geldt: $\alpha_1 = 2\alpha_2$
3. Voor $4 \leq N \leq 11$ geldt: $\alpha_1 > \alpha_j$ met $1 < j < N$

Voor alle $N \leq 980$ gelden de eerste en derde vergelijking. $\alpha_1 = 2\alpha_2$ geldt niet altijd, omdat rond de tweemachten er een andere optimale verdeling is. Bob kan dan minder makkelijk variëren zonder in sommige gevallen vaker te moeten raden. Alice zal dus op een half (en op kwarten, etc.) met een kleinere kans spelen. Hierbij is α_i de kans dat Alice getal i als geheim getal kiest. Gilbert vermoedde dat Alice bepaalde getallen met een kleinere kans speelde. Bijvoorbeeld $N/2$ of in de buurt daarvan en $N/4$ en $3N/4$, etc. Als N rond een tweemacht zit, dan klopt dit vermoeden voor $N \leq 980$. Maar meestal is de verdeling vrij uniform op de randen na.

Bibliografie

[1] Wikimedia Commons. *File:Simplex-method-3-dimensions.png — Wikimedia Commons, the free media repository*. [Online; accessed 8-May-2018]. 2015. URL: `https://commons.wikimedia.org/w/index.php?title=File:Simplex-method-3-dimensions.png&oldid=151784556`.

[2] M. Dresher. *The Mathematics of Games of Strategy*. Reprinted in 1981. Dover Publications, 1961.

[3] E. Gilbert. "Games of identification and convergence". In: *SIAM Rev.* 4.1 (1962), p. 16–24.

[4] S. Johnson. "A search game". In: *Advances in Game Theory*. Red. door M. Dresher en L. S. Shapley. Deel 52. Annals of Mathematics Studies. A.W. Tucker editors, 1964, p. 39–48.