Spellen: van puzzels tot tomografie

dr. Walter Kosters, Informatica
Universiteit Leiden
www.liacs.nl/home/kosters/

Leidsche Flesch, donderdag 15 november 2007

Spellen en puzzels geven aanleiding tot complexe zoekproblemen, bijvoorbeeld bij schaken, go, vier-op-een-rij, sudoku, Japanse puzzels. Extra problemen zijn onzekerheden, kansen, . . .

We hebben vaak een evaluatie-functie nodig om de niet-eindtoestanden te beoordelen, bijvoorbeeld bij schaken: pion 1, paard 3, loper 3, toren 5, koningin 9, veiligheid koning, . . .

We willen prunen: niet alles doorrekenen. We bekijken onder meer het minimax-algoritme van Von Neumann (1928) en het α-β-algoritme uit 1956/58. Maar er zijn ook heel andere technieken . . .

Spellen kunnen als volgt worden ingedeeld:

	deterministisch	kans
perfecte informatie	schaken, dammen, sudoku, checkers, go, othello	monopoly, backgammon
onvolledige informatie	duikbootje, mastermind	bridge, poker, scrabble

Deterministisch: gevolgen van een zet liggen vast, geen kansen.
Perfecte informatie: spelers weten alles van de toestand.

Met name over schaken is veel gepubliceerd, waaronder allerlei anecdotes.
De ultieme uitdaging is vooralsnog het spel go.

Er zijn verschillende strategieën om (een benadering van) de "speltheoretische waarde" van een spel te bepalen. Shannon (1950) onderscheidt drie types:

type A reken alles tot en met zekere diepte door ("spelboom"), en gebruik daar een evaluatie-functie

type B reken soms verder door (als het onrustig is: "quiescence"); gebruik "heuristische" functie om dit te sturen

type C doelgericht menselijk zoeken

A en B zijn meer brute-force, C is meer "knowledge-based".

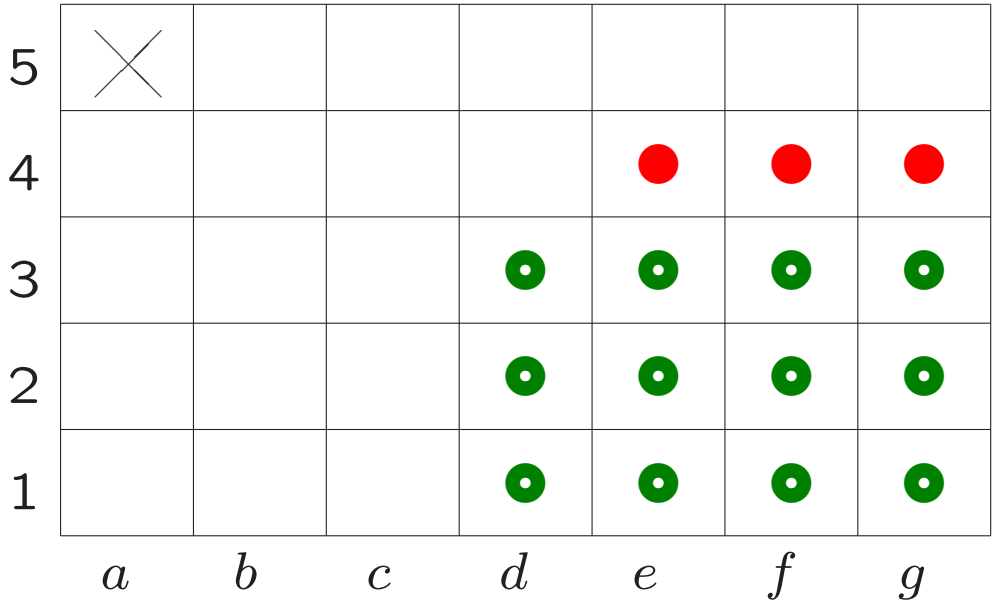
Er worden soms drie soorten oplossingen van spellen onderscheiden:

ultra-zwak de speltheoretische waarde van de beginstand is bekend: “je kunt vier-op-een-rij winnen”

zwak idem, en een optimale strategie is bekend (begin in middelste kolom, . . . , zie later)

sterk in elke legale positie is een optimale strategie bekend

Het spel Chomp wordt gespeeld met een rechthoekige reep chocola, waar de spelers om de beurt een stuk rechtsonder afhappen (een blokje en alles hier onder/rechts van). Wie het (vergiftigde) blokje linksboven eet, heeft verloren.



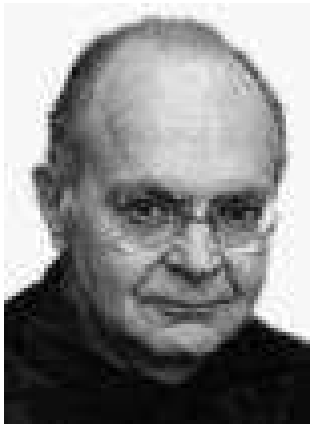
eerste hap: $d3$

tweede hap: $e4$

Bewering: de beginnende speler bij Chomp kan altijd winnen! Immers, als je door het blokje rechtsonder te nemen kunt winnen is het goed. Als dat niet zo is, heeft de tegenstander blijkbaar een voor hem winnende "tegen-hap". Die kun je dan zelf als eerste doen, en dus daarmee winnen! Dit argument heet strategy stealing.

Kortom: het spel Chomp is ultra-zwak opgelost, de echte winnende zet weten we niet . . .

Voor bijvoorbeeld 2×2 en 2×3 Chomp "wint" het blokje rechtsonder (algemener voor vierkanten: neem het vakje rechts onder het vergiftigde); bij 3×4 Chomp het blokje uit de middelste rij, derde kolom. Zie Algoritmiek.

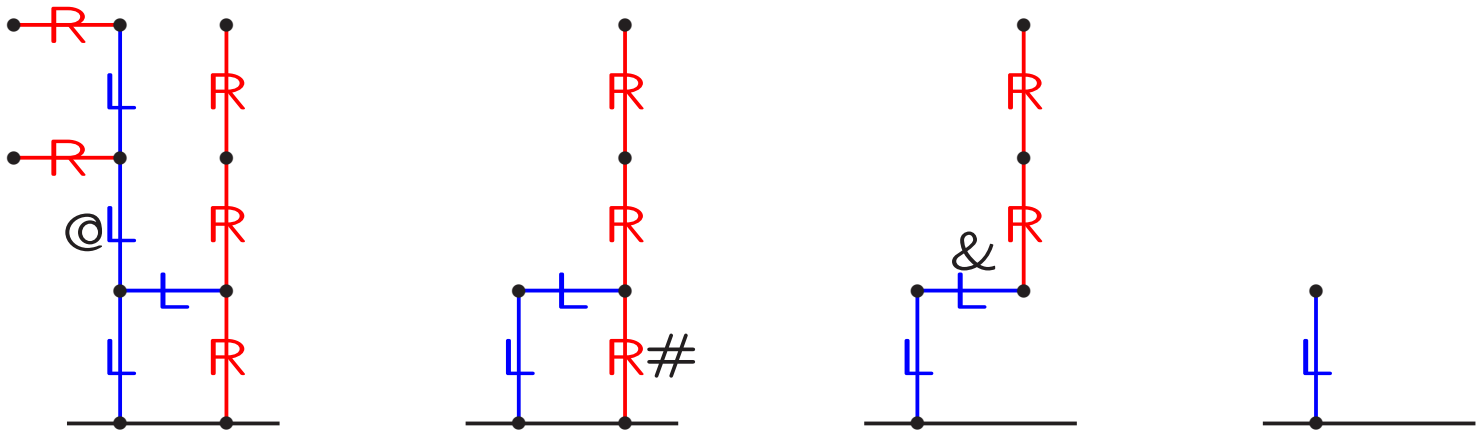


Donald E.(Ervin) Knuth
1938, US
NP; KMP
TEX
change-ringing; 3:16
The Art of Computer Programming

John H.(Horton) Conway
1937, UK → US
Co_1, Co_2, Co_3
Doomsday algoritme
game of Life; Angel problem
Winning Ways for your Mathematical Plays

Surreële getallen (**Surreal numbers**)

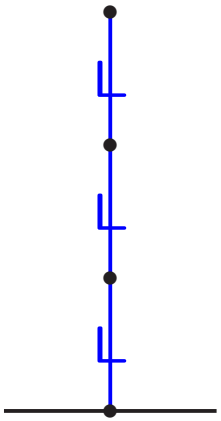
Bij het spel Hackenbush verwijderen Links en Rechts om de beurt respectievelijk een bLauw of een Rood streepje, waarna alle streepjes die niet meer met de grond verbonden zijn ook worden verwijderd. *Wie niet kan, heeft verloren!*



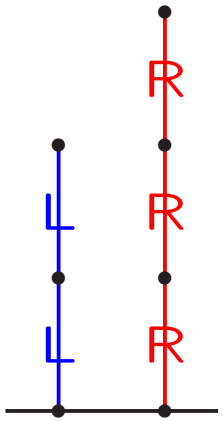
Links kiest @, Rechts kiest # (dom), Links kiest & en wint

Overigens: hier kan Rechts altijd winnen, wie er ook begint!

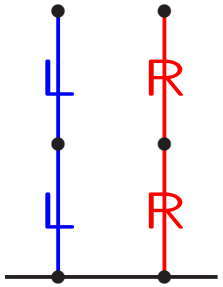
Wat is bij Hackenbush de waarde van een positie?



waarde 3



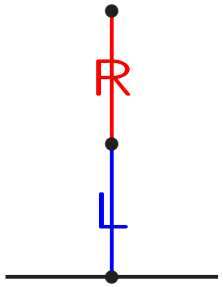
waarde $2-3=-1$



waarde $2-2=0$

Als de waarde positief (>0) is, *kan* Links altijd winnen (wie er ook begint; in het linker voorbeeld met voorsprong 3), als de waarde negatief (<0) is *kan* Rechts altijd winnen, en als de waarde 0 is verliest de beginspeler.

Maar wat is de waarde van deze positie?



Als Links begint, wint hij meteen; als Rechts begint, kan Links nog een keer, en wint hij ook. Dus Links wint altijd. De waarde is daarom > 0.

Vraag: is de waarde gelijk aan 1?

Als de waarde links 1 zou zijn, zou de waarde van de rechter positie $1 + (-1) = 0$ moeten zijn, en zou de beginspeler hier moeten verliezen. Is dat zo?

Nee: het is zo dat als Links begint, Links verliest, en als Rechts begint Rechts ook kan winnen. Dus Rechts wint altijd (= kan altijd winnen), en daarom is de rechter positie < 0, en de linker tussen 0 en 1.

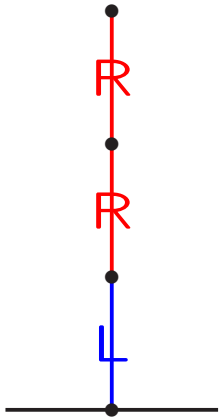
We noteren de waarde van de linker positie met $\{\, 0 \mid 1 \,\}$.

We merken op dat de rechter positie wél waarde 0 heeft: de beginspeler verliest. En dus geldt:

$$\{\, 0 \mid 1 \,\} + \{\, 0 \mid 1 \,\} + (-1) = 0,$$

en blijkbaar $\{\, 0 \mid 1 \,\} = 1/2$.

We noteren de waarde van een positie waarin Links kan spelen naar (waardes van) posities uit de verzameling L en Rechts kan spelen naar (waardes van) posities uit de verzameling R met $\{ L \mid R \}$. Een voorbeeld:



De waarde is hier $\{ 0 \mid \frac{1}{2}, 1 \} = \frac{1}{4}$.

De waarde blijkt altijd het "eenvoudigste" getal dat tussen linker en rechter verzameling in zit.

Op deze manier definiëren we surreële getallen: het zijn "nette" paren van verzamelingen eerder gedefinieerde surreële getallen.

We beginnen met $0 = \{ \emptyset \mid \emptyset \} = \{ \text{NIKS} \mid \text{NIKS} \} = \{ \mid \}$: het spel waarbij de beginspeler geen enkele mogelijkheid heeft, en dus verliest.

En dan $1 = \{ 0 \mid \}$ en $-1 = \{ \mid 0 \}$.
En $42 = \{ 41 \mid \}$.
En $\frac{3}{8} = \{ \frac{1}{4} \mid \frac{1}{2} \}$.
En $\pi = \{ 3, 3\frac{1}{8}, 3\frac{9}{64}, \ldots \mid 4, 3\frac{1}{2}, 3\frac{1}{4}, 3\frac{3}{16}, 3\frac{5}{32}, \ldots \}$.

De reële getallen (de verzameling $\mathbf{R}$) zitten in de surreële getallen (de verzameling $\mathbf{S}$).

De zogeheten Dali-functie $\delta : \mathbf{R} \to \mathbf{S}$ verzorgt die "inbedding": $\delta(1) = \{\, 0 \mid \} = 1$. Maar er is meer ...

`www.tondering.dk/claus/surreal.html`

We definieren bijvoorbeeld:

$$\varepsilon = \{\, 0 \mid \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \ldots \,\},$$

een "ongelooflijk klein positief getal", en

$$\omega = \{\, 0, 1, 2, 3, \ldots \mid \,\} = \{\, \mathbf{Z} \mid \emptyset \,\},$$

een "verschrikkelijk groot getal, een soort ∞".

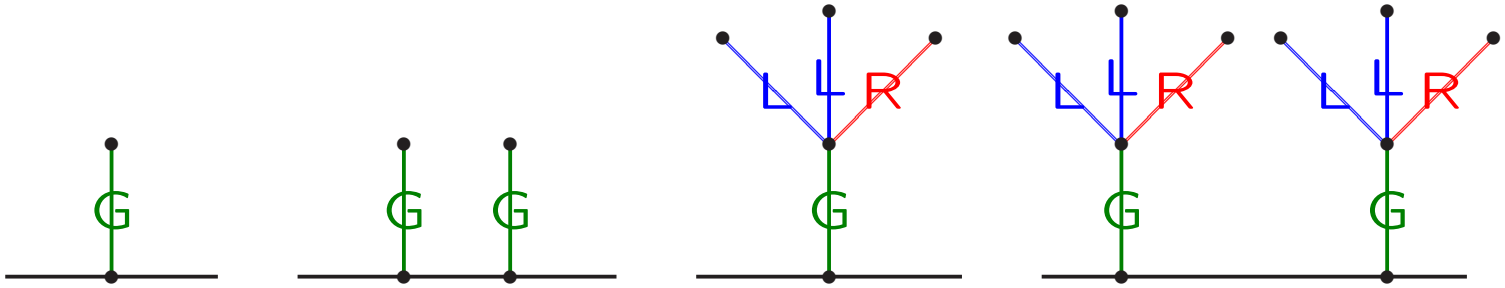
Dan blijkt te gelden:

$$\varepsilon \cdot \omega = 1$$

— mits je vermenigvuldiging goed gedefinieerd is . . .

En dan heb je ook $\omega + 1$, $\sqrt{\omega}$, ω^ω, $\varepsilon/2$, enzovoorts!

Bij Hutspot-Hackenbush zijn er ook nog Groene streepjes, die door *beide* spelers mogen worden weggepakt.

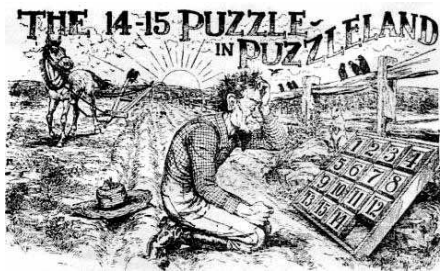


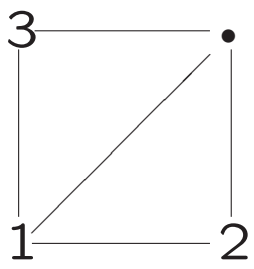
De meest linker positie heeft waarde $* = \{ 0 \mid 0 \}$ (dat blijkt *geen* surreëel getal te zijn), want de beginspeler kan winnen door het Groene streepje te pakken. De tweede van links is $* + * = 0$ (beginner verliest).

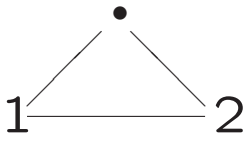
De tweede van rechts is weer gewonnen voor de beginspeler. De meest rechter positie is gewonnen voor Links (wie er ook begint), en is dus > 0.

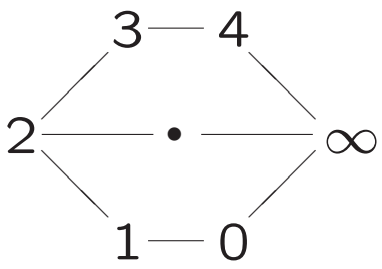
Bij een schuifpuzzel heb je een graaf waarbij alle knopen behalve één een uniek nummer hebben. Je mag een getal naar zijn buur schuiven als die buur geen nummer heeft. De vraag is: welke posities kun je vanuit een vaste bereiken?

Voorbeelden: de 15-puzzel, en onderstaande grafen. De meest rechtse is Wilson's Tricky Six Puzzle.

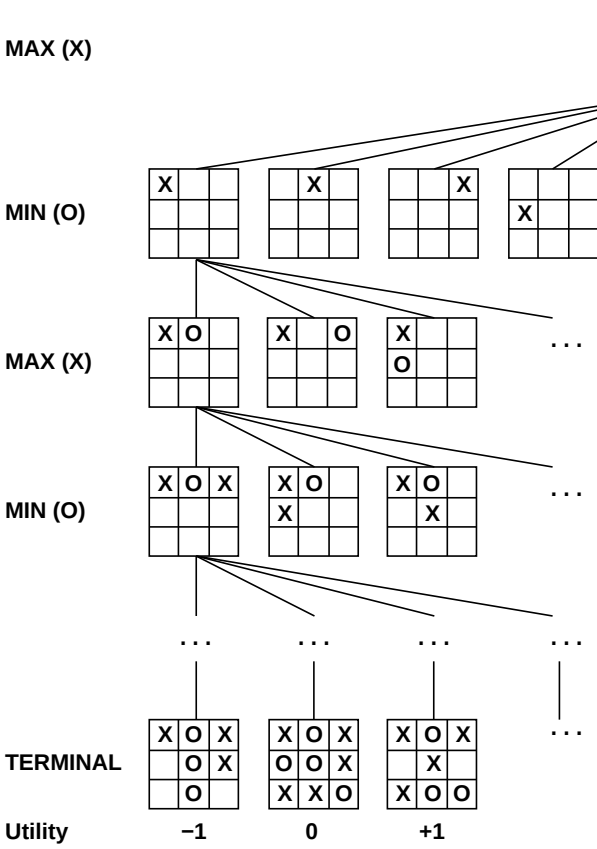








Voorbeeld spelboom



Boter, kaas en eieren

twee spelers:
MAX (X) en MIN (O)

5478 toestanden

(In de zomer van 2007 is Checkers "opgelost": remise.)

Boter, kaas en eieren is een voorbeeld van een tweepersoons deterministisch nulsom-spel met volledige informatie, waarbij de spelers om de beurt een "legale zet" doen. MAX moet een strategie vinden die tot een winnende eindtoestand leidt, ongeacht wat MIN doet. De strategie moet elke mogelijke zet van MIN correct beantwoorden.

Een utility-functie (= payoff-functie) geeft de waarde van eindtoestanden. Hier is dat: $-1/0/1$; bij backgammon is dat: $-192 \ldots +192$.

Uit symmetrie-overwegingen kunnen veel toestanden worden wegbezuinigd.

Maxi en Mini spelen het volgende eenvoudige spel: Maxi wijst eerst een horizontale rij aan, en daarna kiest Mini een verticale kolom.

3	12	8
2	4	6
14	5	2

Bijvoorbeeld: Maxi kiest rij 3, daarna kiest Mini kolom 2; dat levert einduitslag 5.

Maxi wil graag een zo groot mogelijk getal, Mini juist een zo klein mogelijk getal.

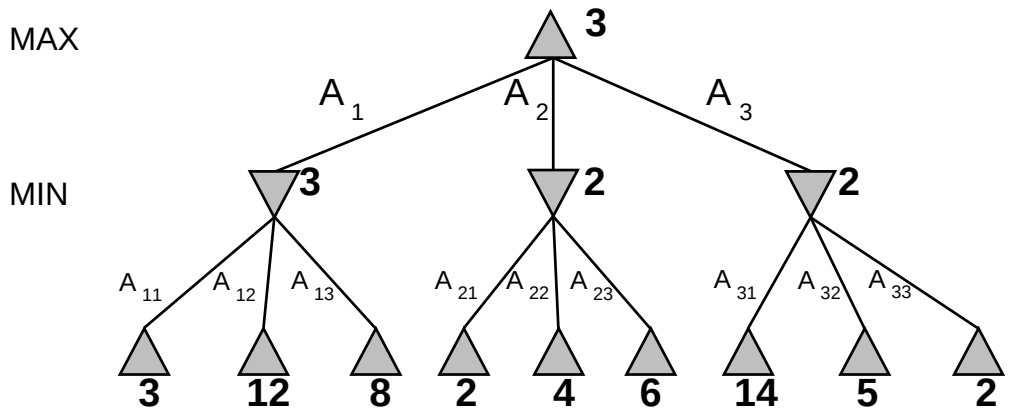
Hoe analyseren we dit spel?

Als Maxi rij 1 kiest, kiest Mini kolom 1 (levert 3); als Maxi rij 2 kiest, kiest Mini kolom 1 (levert 2); als Maxi rij 3 kiest, kiest Mini kolom 3 (levert 2). Dus kiest Maxi rij 1! Dit heet een brute force redenering: we hebben echt alles bekeken.

3	12	8
2	**?**	**?**
14	5	2

Nu merken we op dat de analyse (het minimax-algoritme) hetzelfde verloopt als we niet eens weten wat onder de twee vraagtekens zit. Het α-β-algoritme onthoudt als het ware de beste en slechtste mogelijkheden, en kijkt niet verder als dat toch nergens meer toe kan leiden.

In boomvorm:



Het minimax-algoritme is "recursief": neem in bladeren de evaluatie-functie, in MAX-knopen het maximum van de kinderen, in MIN-knopen het minimum van de kinderen. MAX- en MIN-knopen wisselen elkaar af.

Bovenstaande boom is één zet (= move) diep, oftewel twee ply.

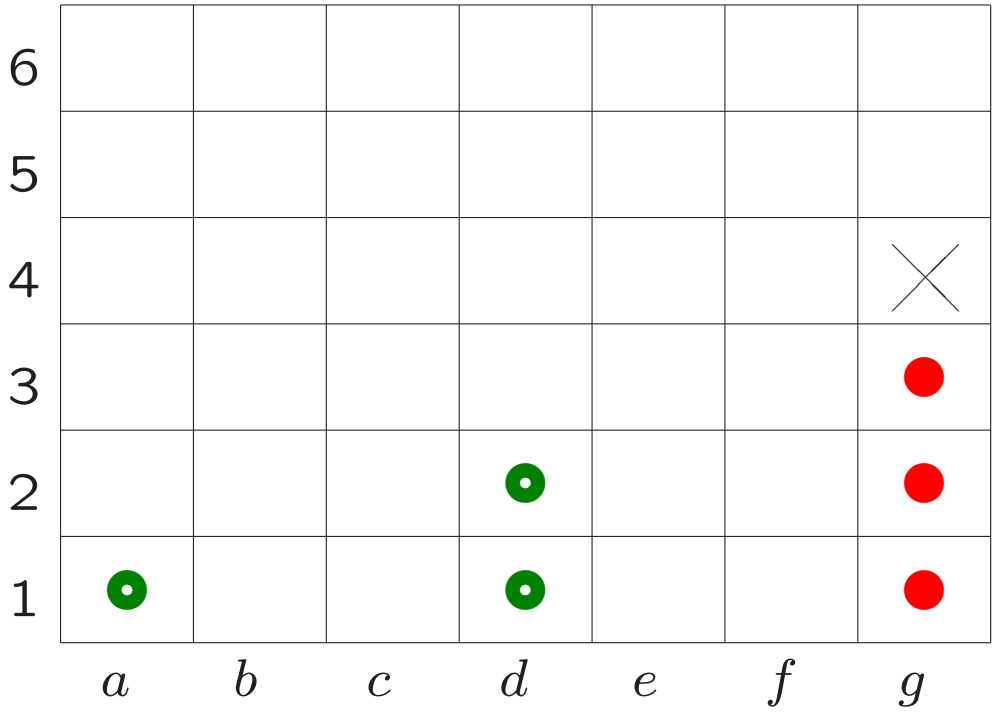
Het is van belang heuristieken (vuistregels) te hebben om je zetten te ordenen. Enkele voorbeelden:

null-move bekijk eerst voor de tegenstander goede zetten (sla je eigen zet in gedachten even over)

killer als een zet ergens een "snoeiing" teweeg brengt, doet hij dat elders wellicht ook

conspiracy-number $\approx$ aantal kinderen dat van waarde moet veranderen (samenzweren) om ouder van waarde te laten veranderen (voor stijgen MAX-knoop is maar één kind nodig, voor stijgen MIN-knoop zijn alle kinderen nodig)

tabu-search onthoud aantal (zeer) slechte zetten

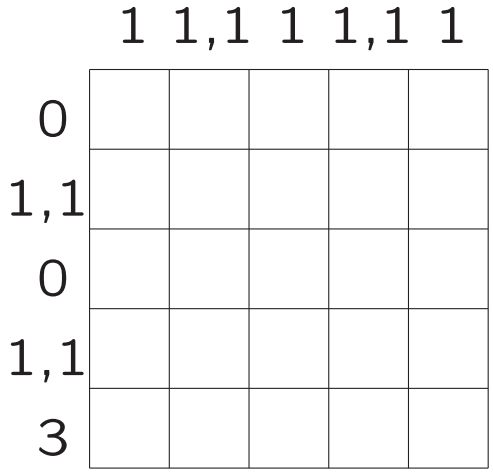


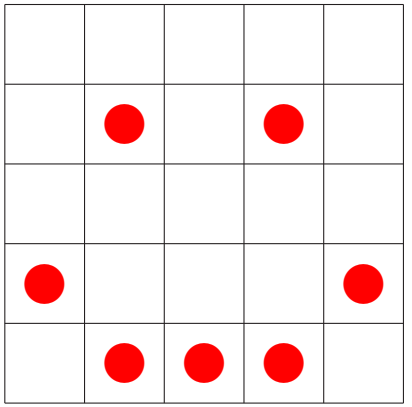
Vier-op-een-rij

Met groen aan de beurt, is $g4$ zowel voor de null-move- als de killer-heuristiek de aangewezen zet.

De voor groen winnende (begin)serie: $d1! - d2 - d3! - d4 - d5! - b1 - b2$ (een ! betekent: unieke winnende zet).

Japanse puzzels (Nonogrammen) zien er zo uit:





Naast iedere rij en boven iedere kolom staan in volgorde de lengtes van aaneengesloten series rode blokjes.

00000010100000010001011110

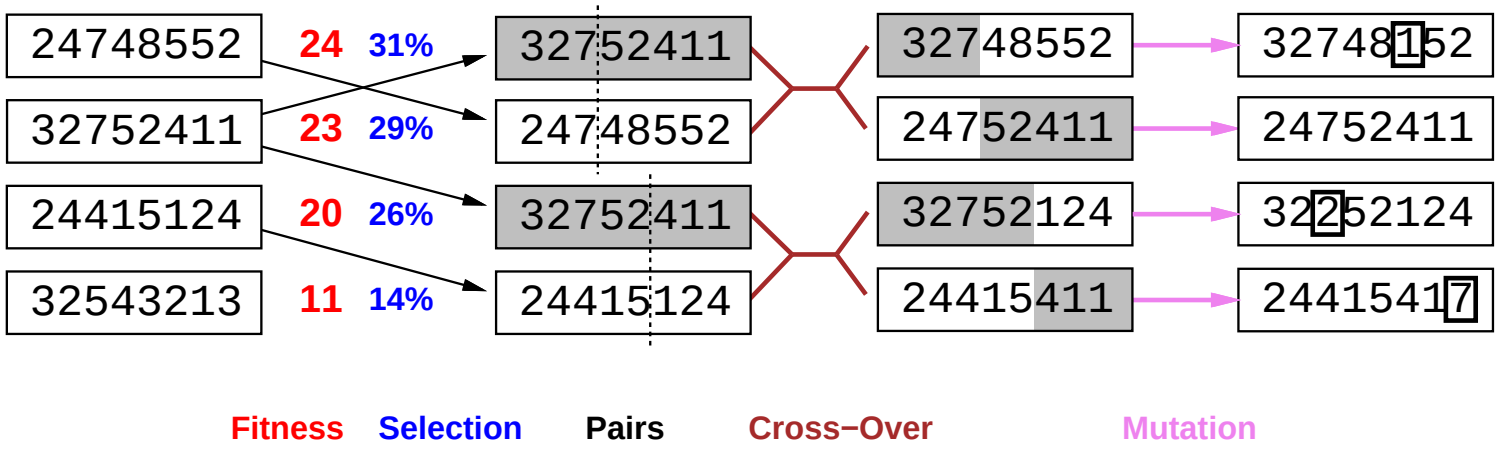
Hoe werkt in het algemeen een evolutionair (genetisch) algoritme?

We hebben steeds een stel kandidaat-oplossingen, individuen, die samen de populatie vormen. Je begint met een aantal willekeurige exemplaren.

Nu ga je die veranderen: muteren op willekeurige plaatsen, en combineren via crossover. Je bewaart steeds de beste individuen; om dat te beoordelen heb je een fitness-functie nodig: hoe hoger die is, des te beter is de oplossing.

Dit alles doe je net zo lang tot je er genoeg van hebt. En het op dat moment beste individu is je "oplossing".

Een voorbeeld-stap is:



Hier hebben we geen bits (0/1), maar getallen in de individuen. Je zou dit kunnen zien als genen. De string heet soms wel het genotype, dat wat hij voorstelt het phenotype.

Evolutionaire (genetische) algoritmen kunnen gebruikt worden om Japanse puzzels op te lossen.

Een individu, een kandidaat-oplossing, is een string met 25 bits (algemener, voor een m bij n puzzel, met mn bits), waarbij een 1 rood en een 0 leeg voorstelt. Je kunt er voor kiezen om het aantal enen per rij altijd "goed" te houden.

Mutatie zou bijvoorbeeld in een rij een 1 en een 0 kunnen verwisselen. Als je handig muteert kun je "alles" bereiken!

De fitness-functie is een som over rijen en kolommen. Per rij/kolom geef je aan hoeveel je van de specificatie afwijkt — en dat is nog lastig precies te maken.

Maar nu: hoe los je "gewoon" Japanse puzzels op?

De meeste mensen gebruiken logische regels, en heuristieken zoals "redeneer eerst een keer via de rijen, en dan via de kolommen".

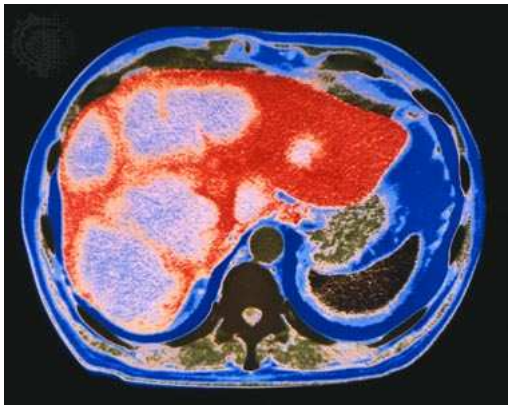
Een voorbeeld van een logische regel voor een enkele rij: "als een getal 3 naast een rij van breedte 5 staat, moet het middelste vakje wel rood zijn".

En verder: NP-volledig, grafen, netwerk-flow, 2-SAT, reguliere expressies, . . .

Tomografie houdt zich bezig met het volgende probleem:

Hoe reconstrueer je een object uit projecties?

Voorbeelden:



- Japanse puzzels oplossen
- Hoe zien de organen eruit, gegeven CT-scans?
- Waar zitten de "gaten" in een diamant?