tekemällä pieniä muutoksia olemassa olevaan puutteelliseen ratkaisuun.

Eräs heuristiikka on *konfliktien minimointi (min-conflicts)*: (jollakin muulla heuristiikalla) valitulle muuttujalle X_j kokeillaan jotakin sellaista uutta arvoa $V \in D_j$, joka toteuttaa mahdollisimman monta tähän muuttujaan liittyvää binäärirajoitetta.

Kuvasarja 38 antaa esimerkin 8 kuningattaren ongelman ratkaisusta tällä periaatteella. Sen avulla jopa satunnainen tapaus $N = 1\ 000\ 000$ ratkeaa keskimäärin alle 50 muutoksella!

Esitetään vielä vuorottain rekursiivinen kaarikonsistenssialgoritmi, joka

- kieltää mahdottomana arvon $X_i = v_i$ poistamalla alkion v_i mahdollisten arvojen joukosta D_i
  (aliohjelma `poista-yksi`)
- kieltää mahdottomana arvoyhdistelmän $X_i = v_i, X_j = v_j$ poistamalla parin $\langle v_i, v_j \rangle$ mahdollisten arvoparien joukosta $R_{i,j}$
  (aliohjelma `poista-pari`)
- jatkaa näitä poistoja rekursiivisesti kunnes
  joko konsistenssi on palautettu jälleen voimaan poistamalla kaikki mitä voidaan
  tai mahdollisten joukot tyhjenevät, mikä osoittaa ettei koko rajoiteongelmalla ole yhtään ratkaisua.

Esityksen yksinkertaistamiseksi oletetaan, että

$$R_{j,i} = \{\langle v_j, v_i \rangle : \langle v_j, v_i \rangle \in R_{i,j}\}$$

kun $i > j$, eli että indeksien järjestyksellä ei ole väliä.

Algoritmia voi optimoida eri tavoin.

Polkukonsistenssi saataisiin muuttamalla algoritmia sellaiseksi, joka saattaa voimaan seuraavan "kolmioehdon":

Kun jokin joukko D_j kutistuu, niin

- Tutki jokainen erillinen (muuttuja)indeksipari i, k (missä siis $i \neq j \neq k \neq i$).
- Poista indeksiparin joukosta $R_{i,k}$ jokainen sellainen arvopari $\langle v_i, v_k \rangle$ joka. . .
- . . . menetti viimeisenkin välittävän alkion $v_j \in D_j$, eli sellaisen jolla $\langle v_i, v_j \rangle \in R_{i,j}$ ja $\langle v_j, v_k \rangle \in R_{j,k}$.

procedure `poista-yksi`(i, v_i)

if $v_i \in D_i$ **then**
  $D_i \leftarrow D_i \setminus \{v_i\}$;
  for all $\langle v_i, v_j \rangle \in R_{i,j}$ missä $j \neq i$ **do**
    `poista-pari`$(i, j, \langle v_i, v_j \rangle)$
  end for
end if

procedure `poista-pari`$(i, j, \langle v_i, v_j \rangle)$

if $\langle v_i, v_j \rangle \in R_{i,j}$ **then**
  $R_{i,j} \leftarrow R_{i,j} \setminus \{\langle v_i, v_j \rangle\}$;
  if $R_{i,j}$ ei sisällä yhtään paria $\left\langle v_i, v'_j \right\rangle$ **then**
    `poista-yksi`(i, v_i)
  end if
  if $R_{i,j}$ ei sisällä yhtään paria $\left\langle v'_i, v_j \right\rangle$ **then**
    `poista-yksi`(j, v_j)
  end if
end if

Algoritmi 9: Kaarikonsistenssialgoritmi.

- 3-konsistenssissa kolmiosta i, j, k tulee "tetra" i, j, k, l
- 4-konsistenssissa "pentagrammi" i, j, k, l, m
- ja k-konsistenssissa täydellinen $k + 1$ solmun aliverkko.

4.3. Pelit [RN95, luku 5][Nil98, luku 12]

Lautapelit ovat valmiiksi abstrahoitu ongelmaluokka, jossa älykkäät toimijat ottavat mittaa

- toisistaan ja/tai
- nopasta (joka edustaa ympäristön sattumanvaraisuutta),

ja siksi kauan ja paljon tutkittuja tekoälyssä.

Pelit ovat esimerkki luvun 3 varautumisongelmasta, jossa hyvää pelistrategiaa etsivän toimijan on *varauduttava siihen, että vastustaja ja/tai noppa eivät tottelekaan*.

Pelit ovat (yleensä) *laskennallisesti erittäin työläitä:*

Kussakin shakkitilanteessa on keskimäärin noin 35 eri siirtomahdollisuutta, ja peli saattaa jatkua 50 siirtoparia.

Siis etsintäpuussa on 35^{100} solmua (joista tosin "vain" noin 10^{40} keskenään erilaista).

Noppa ei siis ole ainoa epävarmuuden lähde, vaan toimija ei myöskään *käytännössä* voi kerätä kaikkea periaatteessa saatavilla olevaa informaatiota päätöksenteon tueksi — varsinkaan jos pelissä on aikaraja, eli jos se on semistaattinen ympäristö.

Rajoitutaan tarkastelemaan

kahden pelaajan. . . nimiltään $\max$ (joka tekee aloitus*siirron*) ja $\min$ (joka *vastaa* aloitukseen).

Useamman pelaajan peleissä joudutaan esimerkiksi määrittelemään, saavatko pelaajat sopia liittoutuvansa keskenään ja pitääkö sopimuksesta kertoa muille pelaajille,. . .

täydellisen informaation. . . eli kumpikin saa pelistä kaiken tarpeellisen tiedon (erityisesti sen säännöt).

Luvun 2.6 käsittein ympäristö on saavutettavissa.

nollasummapelejä. Jaossa oleva "palkintosumma" koostuu tällöin pelkistä pelaajien osallistumismaksuista. Kummankaan ei siis kannata tavoitella häviötä.

Vastakohtana on esimerkiksi nyrkkeilyottelu, jossa häviäjäkin tietää saavansa lipputuloista kipurahoja, joskin vähemmän kuin voittaja.

Silloin rationaalinen nyrkkeilijä saattaa tulla siihen tulokseen, että luovuttaminen heti kannattaa kun sairaalakulutkin lasketaan.

Määritellään tällainen peli seuraavana etsintäongelmana:

Pelitilanne käsittää sekä *laudan L tilanteen* että tiedon *kumman pelaajan vuoro* on siirtää seuraavaksi.

Alkutilanteessa siis L on alkuasetelmassa ja $\max$ vuorossa.

Operaattorit kertovat vuorossa olevan pelaajan P mahdolliset siirrot laudalla L:

Kaikki ne pelitilanteet, joissa

- pelivuoro vaihtuu
- lauta L' saadaan kun P siirtää jotakin "nappulaansa" laudalla L sääntöjen mukaan.

Lopputesti (terminal test) kertoo onko peli ohi annetussa pelitilanteessa.

Näitä pelitilanteita kutsutaan **lopputilanteiksi (terminal states)**.

Utiliteetti- eli tulosfunktio (utility/payoff function) antaa numeroarvon jokaiselle lopputilanteelle H.

Esimerkiksi shakissa

$$U(H) = \begin{cases} +1 & \text{kun } \max \text{ voitti lopputilanteessa } H \\ \pm 0 & \text{kun tuli tasapeli} \\ -1 & \text{kun } \min \text{ voitti.} \end{cases}$$

Tavoitteena on löytää **strategia** aloittajalle $\max$:

Sellainen pelitapa, jonka noudattaminen johtaa aina hänen kannaltaan parhaimpaan tarjolla olevaan lopputulokseen — sellaiseen H, jossa $U(H)$ on niin korkea kuin suinkin on mahdollista saavuttaa.

Vastaavassa etsintä- eli pelipuussa $\max$- ja $\min$-tasot vuorottelevat.

Strategia ei ole etsintäpuun polku vaan *kokonainen alipuu*:

Vastustajan $\min$ siirtoja ei tiedetä etukäteen, vaan niihin kaikkiin on varauduttava.

Pelaaja $\max$ pyrkii valitsemaan sellaisen alipuun, jonka lopputilannelehdet ovat hänen kannaltaan parhaimmat.

Käytännössä tämä valinta joudutaa siis tekemään *epätäydellisen informaation* pohjalta: eri alipuista voidaan tutkia vain jokin juurta lähellä oleva osa.

4.3.1. Minimax-periaate

Parhaimman alipuun valinnassa $\max$ voi järkeillä seuraavasti:

- "Nykyisessä lautatilanteessa L_1 minun kannattaa tehdä *jokin* sellainen siirto M_1 joka vie sellaiseen lautatilanteeseen L'_1, jossa. . .
- *jokainen* siirto M'_1 jonka vastustajani $\min$ voi tehdä lautatilanteessa L'_1 vie sellaiseen lautatilanteeseen L_2, jossa. . .
- minulla on *jokin* seuraava siirto M_2 lautatilanteessa L_2 joka vie sellaiseen lautatilanteeseen L'_2 jossa. . .
- *jokainen* siirto M'_2 jonka vastustajani $\min$ voi tehdä lautatilanteessa L'_2 vie sellaiseen lautatilanteeseen L_3, jossa. . .

$\vdots$

- (viimeisessä) lautatilanteessa L_n peli on päättynyt minun voittooni.

Vastustaja $\min$ ajattelee täsmälleen samoin.

Tarkastellaan sitten sitä *koko pelipuuta*, jonka juurena on nykyinen pelitilanne.

Järkeily johtaa seuraavaan tapaan arvottaa sen kaikki solmut:

Lehti saa arvokseen sitä vastaavan lopputilanteen H utiliteetin $U(H)$.

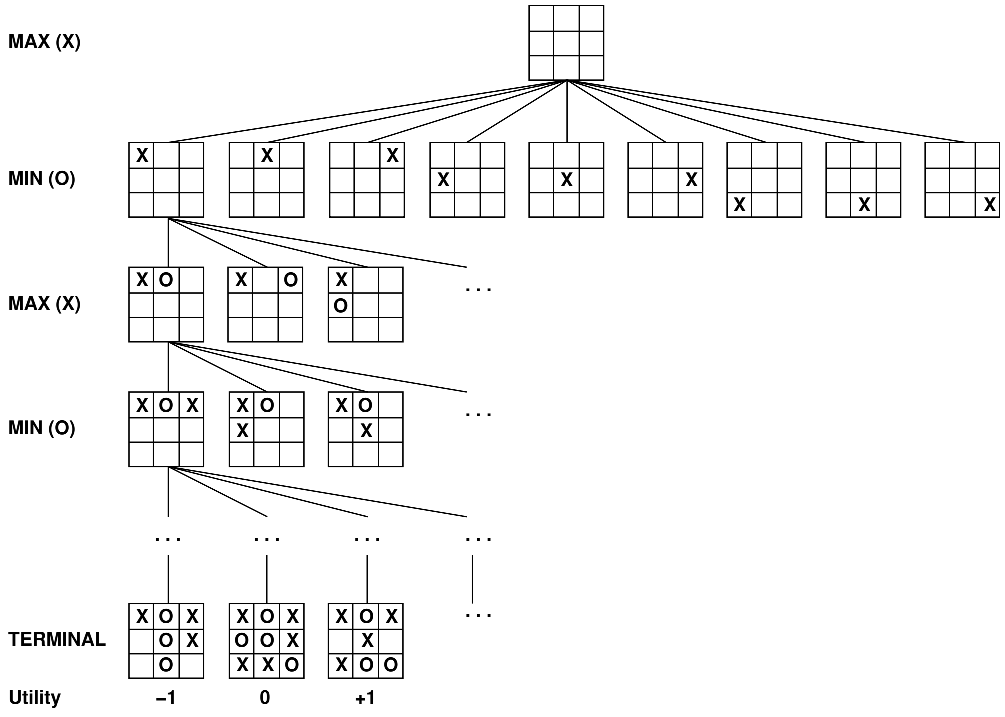
$\max$-**tason sisäsolmu** saa arvokseen luvun *maksimin* sen lapsisolmujen arvoista.

$\min$-**tason sisäsolmu** saa arvokseen luvun *minimin* sen lapsisolmujen arvoista.

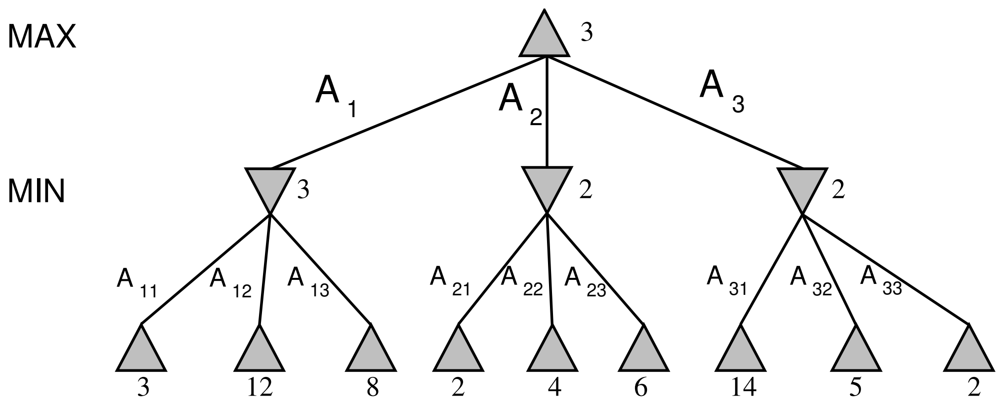
Näin edetään lehdistä aina juureen asti.

Sitten $\max$ tekee **minimax-päätöksen** valitsemalla juuressa sen siirron, joka vie johonkin maksimiarvon antaneeseen alipuuhun.

Siis $\max$ valitsee sen siirron, joka on paras oletuksella, että $\min$ valitsee aina mahdollisimman ikävän vastaussiirron.



Kuva 45: Risti-nolla -pelin etsintäpuu [RN95, kuva 5.1].



Kuva 46: Kahden siirron minimax-puu [RN95, kuva 5.2].