Tuntemattomassa ympäristössä etsiminen

- Online haussa agentin on reagoitava havaintoihinsa välittömästi tekemättä pitkälle tulevaisuuteen ulottuvia suunnitelmia
- Tuntemattomassa ympäristössä tutkiskelu on välttämätöntä: agentin on kokeiltava toimintojensa vaikutuksia saadakseen tietoja maailman tiloista
- Nyt agentti ei voi laskea nykytilan seuraajia, vaan sen on kokeiltava mikä tila seuraa toiminnon suorittamisesta
- Online-algoritmia verrataan usein parhaan offline-algoritmin toimintaan
- Näiden antamien ratkaisujen suhdetta kutsutaan online-algoritmin *kilpasuhteeksi* (competitive ratio)

- Online-hakualgoritmin kilpasuhteen määräämiseksi verrataan agentin kulkeman polun kustannusta sen polun kustannukseen, joka olisi kuljettu, jos ympäristö olisi ollut ennalta tuttu
- Mitä pienempi kilpasuhde on, sen parempi
- Online-algoritmeja voidaan analysoida ajattelemalla niiden toiminta pelinä pahantahtoisen vastustajan kanssa
- Vastustaja saa valita maailman lennosta pakottaakseen online-algoritmin huonoon käyttäytymiseen

- Kaikki käsittelemämme hakualgoritmit eivät sovellu online-hakuun
- Esimerkiksi A* perustuu oleellisesti siihen, että hakupuussa voidaan laajentaa mikä tahansa generoitu solmu
- Online-maailmassa laajennuksen tulee kohdistua siihen solmuun, jossa agentti on
- Syvyyssuuntaisen haun lokaalisuus (paitsi peruutusvaiheessa) tekee algoritmista käyttökelpoisen (jos toiminnot voidaan konkreettisesti peruuttaa)
- Syvyyssuuntainen haku ei ole kilpailukykyinen (kilpasuhteelle ei voida asettaa rajaa)

- Mäenkapuaminen on itse asiassa online-algoritmi, mutta juuttuu lokaaleihin maksimeihin
- Satunnaisia uudelleenkäynnistyksiä ei voi käyttää
- Satunnaiskulut ovat liian tehottomia
- Lisämuistin avulla mäenkapuaminen voi olla käyttökelpoinen menetelmä
- Kullekin vieraillulle tilalle ylläpidetään tämänhetkistä parasta arviota matkasta maaliin
- Lokaalissa minimissä pysymisen sijaan agentti pyrkii arvionsa mukaan lyhintä reittiä maaliin
- Samalla lokaalin minimin arvot päivittyvät pois

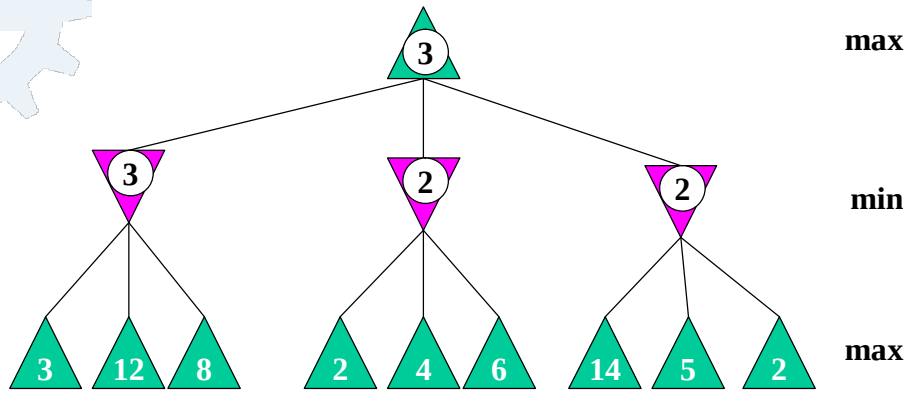
3.2 KAHDEN PELAAJAN PELIT

- Nyt ei enää tarkastella pelkkää polunetsintää alkutilasta maalitilaan
- Myös vastustaja saa tehdä tilasiirtymiä ja pyrkii suistamaan meidät hyvältä polulta
- Tavoitteena on etsiä *siirtostrategia*, joka johtaa maalitilaan riippumatta vastustajan vastasiirroista
- Kahden pelaajan deterministiset, vuoroittaiset, täydellisen informaation, *nollasumma* (zero sum) (lauta)pelit
- Pelin lopussa toinen pelaajista on hävinnyt ja toinen voittanut

- Olkoon pelaajien nimet min ja max
- Lähtötilanteessa peliasetelma on pelin sääntöjen mukainen ja ensimmäinen siirtovuoro on pelaajalla max
- Seuraajafunktio määrää pelin sallitut siirrot
- Pelin loppuminen tunnistetaan
- Hyötyfunktio (utility f.) antaa lopputiloille numeerisen arvon, numeerinen arvo voi olla esim. shakissa vain -1, 0, +1 tai laudalla olevien nappuloiden laskennallinen arvo
- max pyrkii maksimoimaan ja min minimoimaan hyötyfunktion arvon
- Alkutila ja seuraajafunktio määräävät *pelipuun*, jossa pelaajat vuorotellen valitsevat kuljettavan kaaren

- Parhaan mahdollisen pelistrategian etsimisessä oletamme vastustajankin olevan erehtymätön
- Pelaaja min valitsee kannaltaan parhaat siirrot
- Optimaalisen strategian määräämiseksi kullekin solmulle n lasketaan *minimax-arvo:*

$$MM(n) = \begin{cases} hyöty(n), & n \text{ on loppusolmu} \\ \max_{s \in S(n)} MM(s), & n \text{ on max-solmu} \\ \min_{s \in S(n)} MM(s), & n \text{ on min-solmu} \end{cases}$$



- Kahden optimaalisen pelaajan välinen pelinkulku määräytyy täysin minimax-arvojen mukaan
- Pelaajan max kannalta minimax-arvot määräävät pahimman tapauksen tuloksen — vastustaja min on optimaalinen
- Jos vastustaja ei teekään parhaita mahdollisia siirtoja, niin max pärjää vähintään yhtä hyvin
- Ei-optimaalista vastustajaa vastaan jokin muu strategia kuin minimax voi johtaa parempaan lopputulokseen
- Minimax-algoritmi käy läpi koko pelipuun, joten sen aikavaativuus on $O(b^d)$, missä b siirtojen lukumäärä kussakin vaiheessa ja d on puun maksimisyvyys
- Eksponentiaalinen aika on liikaa oikeissa pelipuissa

Alpha-beta-karsinta

- Minimax-strategian eksponentiaalista vaativuutta voidaan helpottaa karsimalla pelipuun evaluoitavia solmuja
- Oikea minimax-päätös voidaan laskea tutkimatta kaikkia puun solmuja
- Esim. edellä olleen puun minimax-arvon määräämiseksi kaksi lehteä voidaan jättää tutkimatta, koska

MM(juuri)
= max(min(3, 12, 8), min(2, x, y), min(14, 5, 2))
= max(3, min(2, x, y), 2)
= max(3, z, 2), missä $z \leq 2$
= 3

- Juuren minimax-arvo ei riipu lehtien x ja y arvosta
- Yleisemmin karsinnan periaate on:
  - pohdittaessa solmuun n siirtymistä,
  - jos pelaajalla on parempi valinta m tehtävänään joko n:n isäsolmussa tai missä tahansa pelipuun ylemmällä tasolla,
  - niin solmua n ei koskaan valita pelissä
- Alpha-beta-karsinnan nimi tulee ylläpidetyistä muuttujista
  - α = parhaan (korkeimman) tavatun valintamahdollisuuden arvo pelaajan max polulla
  - β = parhaan (pienimmän) tavatun valintamahdollisuuden arvo pelaajan min polulla

- Muuttujien α ja β arvoja päivitetään puuta läpikäytäessä
- Heti kun solmun minimax-arvon tiedetään jäävän heikommaksi kuin α (max) tai β (min) jäljellä olevat haarat voidaan karsia
- Alpha-beta-karsinnan tehokkuus riippuu oleellisesti seuraajasolmujen tutkimisjärjestyksestä
- Esimerkkipuumme viimeisestä haarasta ei voitu karsia yhtään seuraajaa, koska minimoinnin kannalta huonoimmat seuraajat generoitiin ensin
- Jos viimeinen lehti olisi generoitu ennen sisariaan, ne olisi molemmat voitu karsia

- Jos parhaat seuraajat pystyttäisiin tutkimaan ensin, putoaisi minimax-haun $O(b^d)$ aikavaativuus luokkaan $O(b^{d/2})$
- Efektiivisesti siis haarautumisaste b putoaisi $\sqrt{b}$:hen
- Esimerkiksi shakissa tämä tarkoittaa käytännössä astetta 6 alkuperäisen 35:n sijaan
- Peleissä ei voida evaluoida kokonaisia hakupuita, vaan niissä pyritään evaluoimaan osittaisia hakupuita mahdollisimman monta siirtoa eteenpäin
- Alpha-beta-haku siis pystyisi tutkimaan hakupuuta suunnilleen kaksi kertaa niin pitkälle kuin minimax-haku samassa ajassa

Tosiaikaiset päätökset

- Koko pelipuun läpikäyminen millä tahansa realistisella pelillä on liian hidasta
- Pelipuun generoiminen on katkaistava johonkin syvyyteen ja absoluuttiset lopputilojen arvot on korvattava heuristisilla arvioilla
- Pelitilanteita on siis arvotettava sen mukaan kuinka hyviltä ne vaikuttavat (maalitilaan pääsemisen kannalta)
- Perusvaatimus heuristiselle evaluointifunktiolle on, että se arvottaa lopputilanteet oikeaan järjestykseen
- Tietysti pelitilanteiden evaluointi ei saa olla liian hidasta ja arvioiden tulisi vastata tilanteiden oikeaa hyödyllisyyttä

- Heuristinen evaluointifunktio perustuu usein peliaseman *piirteiden* (feature) huomioimiseen
- Esimerkiksi shakissa yksi piirre voisi olla kummankin pelaajan sotamiesten lukumäärä laudalla
- Kun peliasetelma typistetään valittujen piirteiden arvoiksi, erilaiset pelitilanteet voivat näyttää ekvivalenteilta, vaikka osa niistä johtaakin voittoon, osa tasapeliin ja osa tappioon
- Tällaisessa ekvivalenssiluokassa voimme laskea odotusarvoisen tuloksen
- Jos esim. 72% peleistä päättyy voittoon (+1), 20% tappioon (-1) ja 8% tasapeliin, niin tästä piirrearvojen asetuksesta jatkuvan pelin hyödyn odotusarvo on:

$$(0{,}72 \times 1) + (0{,}20 \times -1) + (0{,}08 \times 0) = 0{,}52$$

- Koska piirteitä ja niiden mahdollisia arvoja yleensä on paljon, ei tällaista kategorioihin perustuvaa menetelmää usein voida käyttää
- Sen sijaan kukin piirre f_i usein arvotetaan yksinään ja pelitilanteen s evaluoinniksi otetaan valittujen piirteiden painotettu lineaarikombinaatio

$$eval(s) = \sum_{i=1,\ldots,n} w_i \, f_i(s)$$

- Esimerkiksi shakissa piirteitä f_i voisivat olla sotilaiden, lähettien, tornien ja kuningattarien lukumäärät
- Näiden piirteiden painoja w_i puolestaan olisivat nappuloiden materiaaliset arvot (1, 3, 5 ja 9)

- Piirteiden lineaarikombinaatioon sisältyy ajatus piirteiden *riippumattomuudesta*
- Kuitenkin esimerkiksi shakissa lähetit ovat loppupelissä, kun laudalla on paljon tilaa, vahvoja nappuloita
- Siksi nykyiset shakkiohjelmat sisältävätkin myös piirteiden *epälineaarisia* kombinaatioita
- Esimerkiksi lähettipari on painoarvoltaan enemmän kuin kaksi kertaa yhden lähetin paino ja loppupelissä lähetin arvoa nostetaan
- Jos eri nappuloiden painoarvoilla ei ole vuosisatojen kokemuksen tuomaa taustaa kuten shakissa, voidaan tällaisia sääntöjä pyrkiä oppimaan koneoppimisen menetelmin

Sattuman huomioiminen

- Peleihin sattuma voidaan lisätä eksplisiittisesti esimerkiksi nopanheitolla
- Lautapeleistä ainakin backgammon on tällainen
- Vaikka pelaaja nyt tunteekin omat siirtonsa, hän ei tiedä vastustajan pyöräyttämiä silmälukuja eikä täten vastustajan sallittuja siirtoja
- Tavallista pelipuuta ei siis voida muodostaa
- Pelipuuhun voidaan lisätä max- ja min-solmujen lisäksi sattumasolmuja, joista johtavat kaaret nimetään nopanheiton mahdollisilla tuloksilla ja niiden todennäköisyyksillä

- Esim. backgammonissa heitetään kahta noppaa kerrallaan, joten erilaisia parimahdollisuuksia on 6+15 ja niiden todennäköisyydet ovat 1/36 ja 1/18
- Tarkkojen minimax-arvojen sijaan meidän on nyt tyydyttävä odotusarvon laskemiseen
- Esim. pelipuun max-solmun n arvo määrätään nyt

$$= \max_{s \in S(n)} \mathbf{E}[\, MM(s) \,]$$

- Sattumasolmussa n lasketaan kaikkien nopanheittojen todennäköisyydellä painotettu keskiarvo

$$\sum_{s \in S(n)} P(s) \cdot \mathbf{E}[\, MM(s) \,]$$

- Tilanteiden arvottaminen sattuman vallitessa on herkempi tehtävä kuin deterministisessä pelissä

- Sattuman mukaan tuominen nostaa pelipuun läpikäynnin vaativuuden luokkaan $O(b^m n^m)$, missä n on nopanheiton mahdollisten tulosten lukumäärä
- Backgammonissa $n = 21$ ja b on yleensä noin 20, mutta voi olla jopa 4000 kun nopanheiton tulos on pari
- Vaikka laskentasyvyys olisi rajoitettu, niin sattuman vallitessa ei pelejä voi laskea pitkälle eteenpäin
- Alpha-beta-karsinnan voi nähdä keskittyvän todennäköisiin puun haaroihin
- Sattumanvaraisessa pelissä ei ole todennäköisiä siirtojonoja
- Kuitenkin jos hyötyarvot ovat rajoitettuja, niin myös sattumasolmuja sisältävässä pelipuussa voidaan tehdä karsintaa

Deep Blue

- IBM:llä kehitetty rinnakkaislaitteistoon perustuva shakki-ohjelmisto, joka vuonna 1997 voitti maailmanmestari Garry Kasparovin kuuden pelin ottelussa
- Hakunopeus keskimäärin 126 miljoonaa solmua sekunnissa (max 330 miljoonaa)
- Normaali hakusyvyys pelipuussa: 14
- Perusmenetelmä on iteratiivisesti syvenevä alpha-beta haku
- Menestyksen salaisuus mahdollisuus nähdä hakusyvyyttä pidemmälle joissain riittävän mielenkiintoisissa tapauksissa (jopa 40 puolisiirtoa)
- Evaluointifunktiossa 8000 piirrettä
- Laajat alku- ja loppupelikirjastot

4. EPÄVARMA TIETÄMYS JA PÄÄTTELY

- Käytännössä agentilla ei ole koskaan täyttä tietämystä toimintaympäristöstään, vaan se joutuu toimimaan *epävarmuuden* vallitessa
- Logiikkaan perustuva agentti voi joutua tilanteeseen, jossa se ei saa varmuudella tarvitsemiaan tietoja
- Jos agentti ei voi todeta jonkin toimintasuunnitelman toteuttavan sen tavoitetta, niin se ei voi toimia
- Ehdollinen suunnittelu voi helpottaa tilannetta, mutta ei ratkaise sitä
- Yksinomaan logiikkaan perustuva agentti ei kykene valitsemaan rationaalisia toimintoja epävarmassa toimintaympäristössä

- Looginen tietämyksen esitys vaatii poikkeuksettomia sääntöjä
- Usein käytännössä voimme taata vain jonkin *uskomuksen asteen* esitetylle väittämälle
- Uskomusten käsittelyssä turvaudumme *todennäköisyysteoriaan*
- Todennäköisyys 0 on vastaansanomaton usko lauseen epätotuudesta ja 1 usko sen totuuteen
- Väliin jäävät arvot kertovat uskomme lujuudesta lauseen totuudesta, eivät lauseen totuuden suhteellisuudesta
- Todennäköisyyksillä painotetut hyötyarviot (utility) antavat agentilla mahdollisuuden rationaaliseen toimintaan hyödyn odotusarvon maksimoinnin kautta

Todennäköisyysteoria

- Satunnaismuuttuja (random variable) viittaa maailman osaan, jonka tilannetta ei tunneta ennalta
- Satunnaismuuttujat vastaavat propositiosymboleita
- Esim. satunnaismuuttuja Reikä voisi viitata siihen onko vasemmassa alaleuan viisaudenhampaassani reikä
- Satunnaismuuttujan *arvoalue* voi olla
  - *totuusarvoinen*:
    merk. Reikä = true ⇨ reikä ja Reikä = false ⇨ ¬reikä;
  - *diskreetti*: esim. satunnaismuuttujan Säätila arvoalue voisi olla { aurinkoa, sadetta, pilveä, lunta };
  - *jatkuva*: tällöin tarkastellaan useimmiten kertymäfunktiota, esim. X ≤ 4.02

- Alkeispropositioita kootaan loogisin konnektiivein monimutkaisemmiksi lauseiksi: reikä ∧ ¬hammassärky
- Atominen *tapahtuma* (event) on kaikkien muuttujien täydellinen arvoasetus
- Atomisille tapahtumille pätee, että
  - ne ovat toisensa poissulkevia,
  - niiden joukko on kattava — ainakin yksi niistä on tosi,
  - Mikä tahansa atominen tapahtuma määrää kaikkien propositioiden totuuden ja
  - Mikä tahansa propositio on loogisesti ekvivalentti kaikkien niiden atomisten tapahtumien disjunktion kanssa, joista sen totuus seuraa

reikä ≡ (reikä ∧ hammassärky) ∨ (reikä ∧ ¬hammassärky)

- Uskomuksen aste kohdistetaan propositionaalisiin lauseisiin
- Prioritodennäköisyys P(a) antaa proposition a uskomuksen asteen muun tiedon puuttuessa
- P(reikä) = 0.1
- Erityisesti siis prioritodennäköisyys on agentin alkuperäinen uskomus, ennen omia havaintoja
- Muuttujan X todennäköisyysjakauma **P**(X) on sen (järjestetyn) arvoalueen alkioiden todennäköisyyksien vektori
- Esim. kun P(aurinkoa) = 0.02, P(sadetta) = 0.2, P(pilveä) = 0.7 ja P(lunta) = 0.08, niin

P(Säätila) = [0.02, 0.2, 0.7, 0.08]

- Kahden muuttujan yhteis(todennäköisyys)jakauma (joint probability distribution) on niiden arvoalueiden tulo
- Esim. **P**(Säätila, Reikä) on 4 × 2 taulukko todennäköisyyksiä
- Täysi yhteistodennäköisyysjakauma on kaikkien maailman kuvaamiseen käytettyjen satunnaismuuttujien yhteistodennäköisyysjakauma
- Jatkuva-arvoisille muuttujille ei voida kirjoittaa jakaumataulukkoa, vaan sen sijaan on tarkasteltava todennäköisyyden tiheysfunktiota
- Pistetodennäköisyyksien (joiden arvo on 0) sijaan tarkastellaan osavälien todennäköisyyksiä
- Tarkastelemme enimmäkseen diskreettejä satunnaismuuttujia

- Kun agentti saavuttaa tietoa ympäristön satunnaismuuttujien arvoista, niin prioritodennäköisyyksien sijaan on siirryttävä *ehdollisiin* (posteriori) todennäköisyyksiin
- $P(a \mid b)$ on proposition a tn., kun kaikki mitä tiedämme on b
- $P(\text{reikä} \mid \text{hammassärky}) = 0.8$
- $P(\text{reikä}) = P(\text{reikä} \mid)$
- Kun $P(b) > 0$, niin $P(a \mid b) = P(a \wedge b) / P(b)$,
  joka voidaan kirjoittaa myös *tulosääntönä*:
  $P(a \wedge b) = P(a \mid b)\, P(b)$
- Sääntö voidaan tietysti myös kääntää
  $P(a \wedge b) = P(b \mid a)\, P(a)$
- $\mathbf{P}(X \mid Y) \equiv P(X = x_i \mid Y = y_j)\ \forall\ i, j$

Todennäköisyyden Kolmogorov-aksioomat

- Todennäköisyysteorian aksiomatisointi (1933) kolmen yksinkertaisen aksiooman pohjalta

1. Minkä tahansa proposition a todennäköisyys on välillä 0 ja 1: $0 \leq P(a) \leq 1$
2. Välttämättä toden (validin) proposition todennäköisyys on 1 ja välttämättä epätoden (toteutumattoman) proposition todennäköisyys on 0:
   $P(true) = 1 \qquad P(false) = 0$
3. Disjunktion todennäköisyys on
   $P(a \vee b) = P(a) + P(b) - P(a \wedge b)$

- Aksioomien perusteella voimme päätellä mm. seuraavaa

$$P(a \vee \neg a) = P(a) + P(\neg a) - P(a \wedge \neg a)$$
$$P(true) = P(a) + P(\neg a) - P(false)$$
$$1 = P(a) + P(\neg a)$$
$$P(\neg a) = 1 - P(a)$$

- Kolmannen rivin sääntö yleistyy diskreetille muuttujalle D, jonka arvoalue on $d_1,\ldots,d_n$:

$$\sum_{i=1,\ldots,n} P(D = d_i) = 1$$

- Jatkuva-arvoisella muuttujalla X summa on korvattava integraalilla:

$$\int_{-\infty,\ldots,\infty} P(X = x)\, dx = 1$$

- Yksittäisen muuttujan todennäköisyysjakauman on siis summauduttava 1:een
- Samoin kaikkien yhteisjakaumien
- Propositio a on ekvivalentti kaikkien niiden atomitapahtumien disjunktion kanssa, joiden looginen seuraus se on
- Merkitään näiden atomitapahtumien joukkoa $\mathbf{e}(a)$
- Aksiooman 3 perusteella

$$P(a) = \sum_{e_i \in \mathbf{e}(a)} P(e_i)$$

- Jos täysi yhteisjakauma on tunnettu, niin voimme tämän perusteella laskea minkä tahansa proposition todennäköisyyden

Probabilistinen päättely täydestä yhteisjakaumasta

	hammassärky		¬hammassärky	
	osuma	¬osuma	osuma	¬osuma
reikä	0.108	0.012	0.072	0.008
¬reikä	0.016	0.064	0.144	0.576

- Voimme laskea esim. lauseen reikä ∨ hammassärky tn.:n
  0.108 + 0.012 + 0.072 + 0.008 + 0.016 + 0.064 = 0.28
- Muuttujan (tai muuttujajoukon) arvon *marginaalitodenn*äköisyys saadaan laskemalla vastaavien rivien tai sarakkeiden summat
- Esim. $P(\text{reikä}) = 0.108 + 0.012 + 0.072 + 0.008 = 0.2$
- Yleisesti mille tahansa muuttujajoukoille $\mathbf{Y}$ ja $\mathbf{Z}$ pätee marginalisointisääntö $\mathbf{P}(\mathbf{Y}) = \sum_{\mathbf{z} \in \mathbf{Z}} \mathbf{P}(\mathbf{Y}, \mathbf{z})$

- Ehdollisen todennäköisyyden laskeminen

  $P(\text{reikä} \mid \text{särky}) = P(\text{reikä} \wedge \text{särky})/P(\text{särky}) =$
  $(0.108 + 0.012)/(0.108 + 0.012 + 0.016 + 0.064) = 0.6$
- Vastaavasti $P(\neg\text{reikä} \mid \text{särky}) = 0.08/0.2 = 0.4$
- $1/P(\text{särky}) = 1/0.2 = 5$ on normalisointitekijä, joka varmistaa, että jakauma $\mathbf{P}(\text{Reikä} \mid \text{särky})$ summautuu 1:een
- Merkitään normalisointivakiota α:lla

  $\mathbf{P}(\text{Reikä} \mid \text{särky}) = \alpha\, \mathbf{P}(\text{Reikä}, \text{särky})$
  $= \alpha(\mathbf{P}(\text{Reikä}, \text{särky}, \text{osuma}) + \mathbf{P}(\text{Reikä}, \text{särky}, \neg\text{osuma}))$
  $= \alpha([0.108, 0.016] + [0.012, 0.064])$
  $= \alpha[0.12, 0.08]$
  $= [0.6, 0.4]$

- Yleisemmin: meidän tulee selvittää kyselymuuttujan X jakauma, todistemuuttujien $\mathbf{E}$ havaitut arvot ovat $\mathbf{e}$ ja loput havaitsemattomat muuttujat ovat $\mathbf{Y}$
- Kyselyn evaluointi:

  $$\mathbf{P}(X \mid \mathbf{e}) = \alpha\, \mathbf{P}(X, \mathbf{e}) = \alpha \sum_{\mathbf{y}} \mathbf{P}(X, \mathbf{e}, \mathbf{y}),$$

  missä $\mathbf{y}$:t käyvät yli havaitsemattomien muuttujien kaikkien mahdollisten arvokombinaatioiden
- $\mathbf{P}(X, \mathbf{e}, \mathbf{y})$ on osajoukko kaikkien muuttujien X, $\mathbf{E}$ ja $\mathbf{Y}$ yhteistodennäköisyysjakaumasta
- Taulukon koko on $O(2^n)$ ja sen läpikäymisen aikavaativuus on $O(2^n)$, joten yleisesti ottaen menetelmä ei ole käyttökelpoinen