# Rajoitteiden osittainen täyttäminen

Otto Seppälä

# Mikä?

- Osittaisessa rajoitteiden täyttämisessä täytetään vain jokin osajoukko kaikista rajoitteista
- Voidaan nähdä rajoitteiden lieventämisenä

# Miksi / Milloin?

- Ylirajoitetulle ongelmalle ei löydy ratkaisua
- Liian monimutkainen / laaja ongelma
- Haetaan parasta ratkaisua rajoitettujen resurssien puitteissa
- Osittaisratkaisu täytyy saada tarvittaessa välittömästi

# Rajoitteiden maksimaalinen täyttäminen

- Etsitään ratkaisu joka täyttää mahdollisimman monta rajoitetta

# Rajoitteiden riittävä täyttäminen

- Etsitään ratkaisua jossa rikottujen rajoitteiden lukumäärä laskee ennalta määrätyn rajan alle

# Rajoitteiden täyttäminen resurssien puitteissa

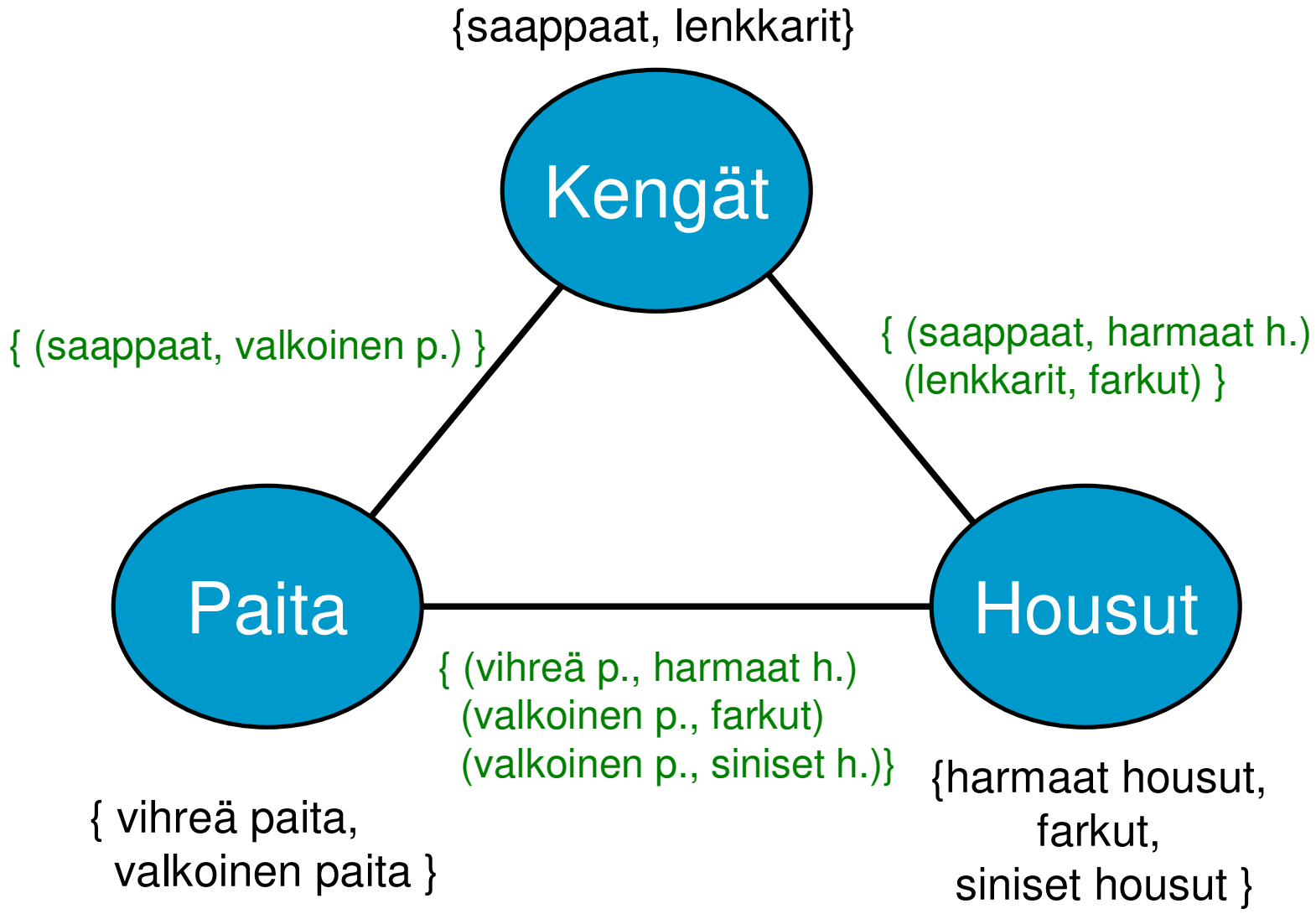
- Etsitään ratkaisua jossa rikottujen rajoitteiden kuluttamat resurssit laskevat ennalta määrätyn rajan alle

# Muuttujat

- N - saavutettu raja-arvo (necessary bound)
- d - etäisyys täydelliseen ratkaisuun
- S - riittävä raja-arvo (sufficient bound)
- Best-solution - paras tähän mennessä löydetty ratkaisu

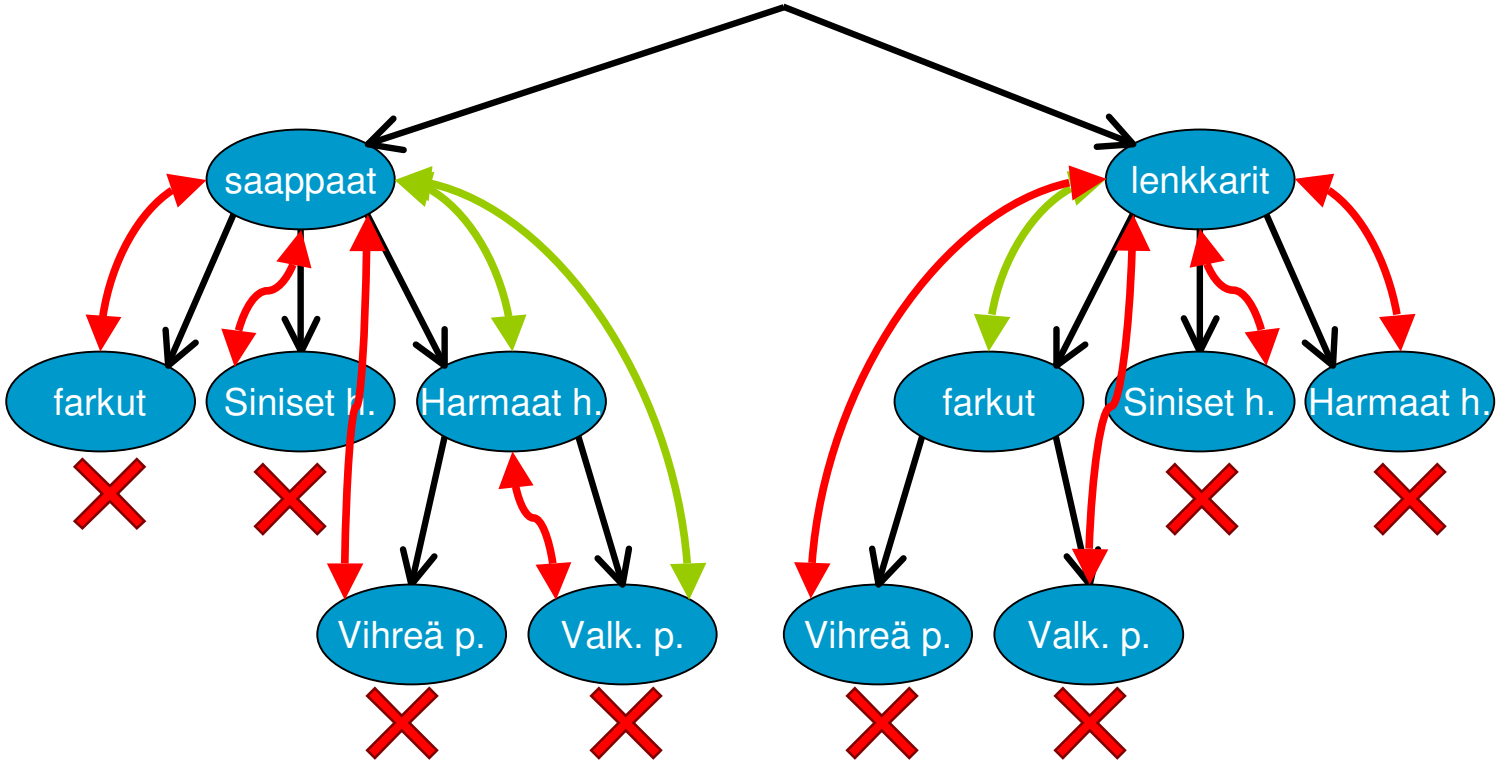
- Jollei ole a priori tietoa ongelmasta, asetetaan
  $S = 0$
  $N = \infty$

# Päivän esimerkki



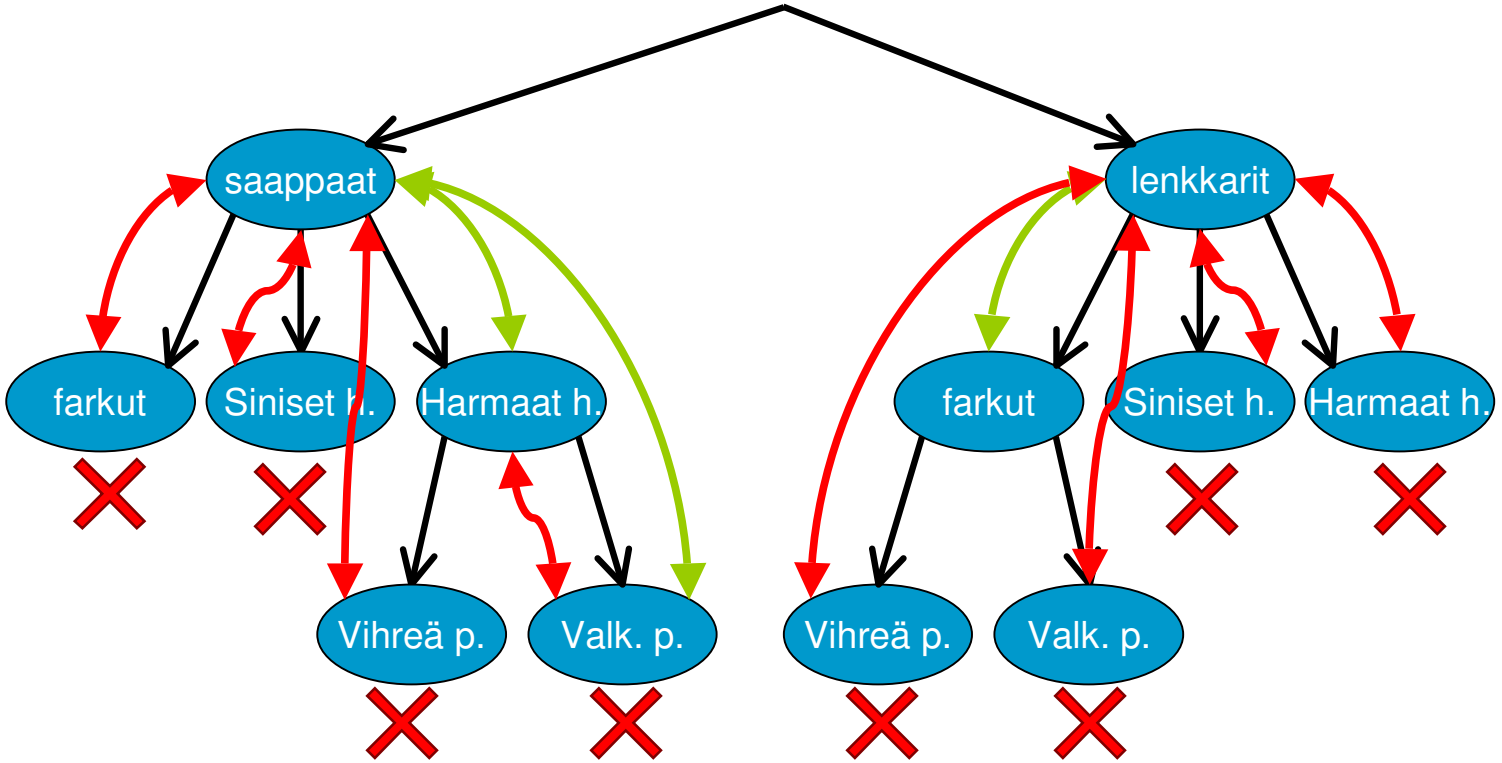
# Retrospektiiviset tekniikat

# Backtracking
# Tavallinen CSP



# Backtracking
# Tavallinen CSP

11 rajoite-
tarkistusta



# Branch&Bound

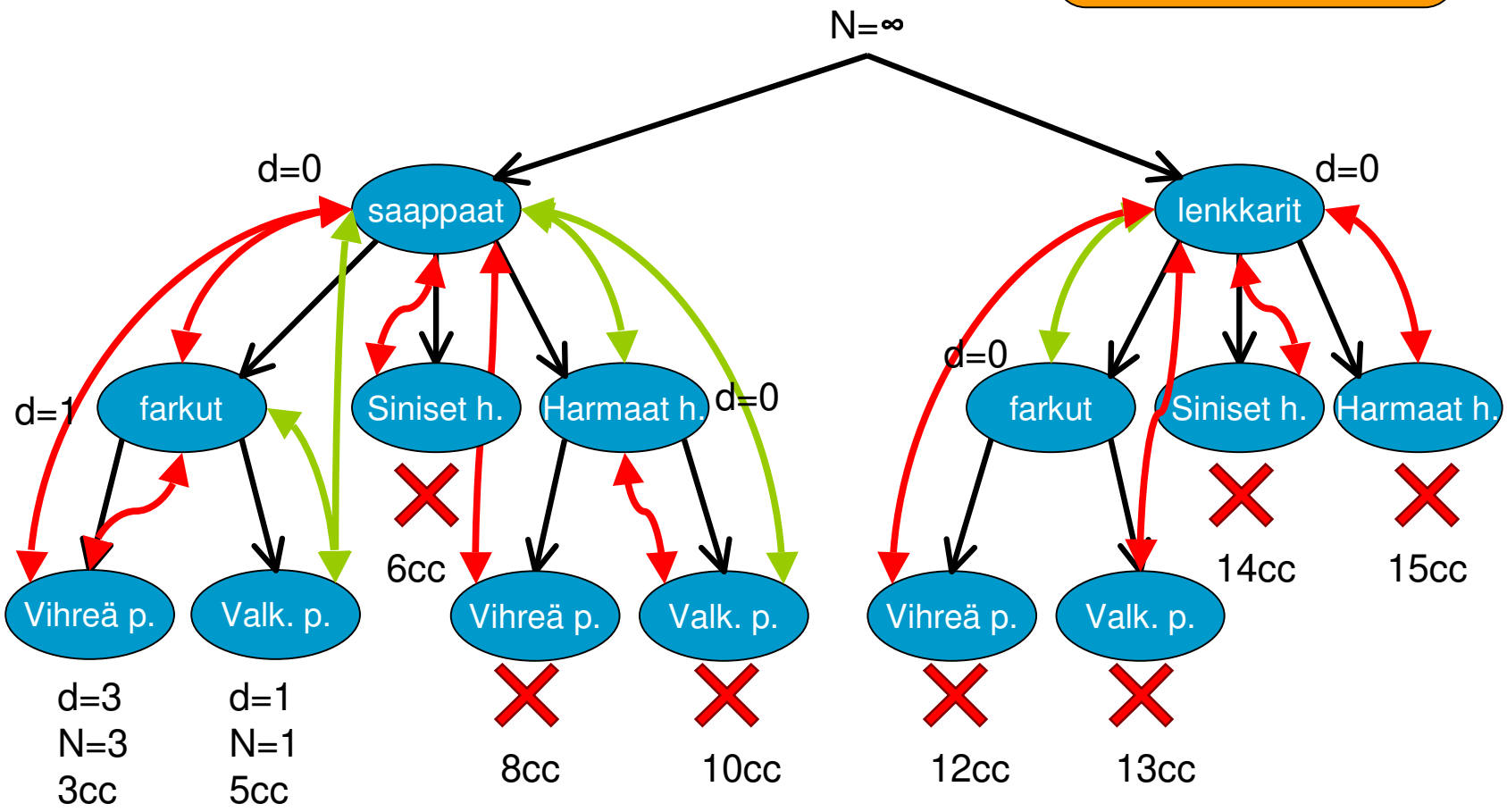
## PCSP, maximal solution

- Branch&Bound karsii hakupuuta kun haettavasta haarasta ei voi löytyä ratkaisua joka voittaisi nykyisen parhaan ratkaisun.

# Branch&Bound

## PCSP, maximal solution

15 rajoite-tarkistusta



# Backjumping

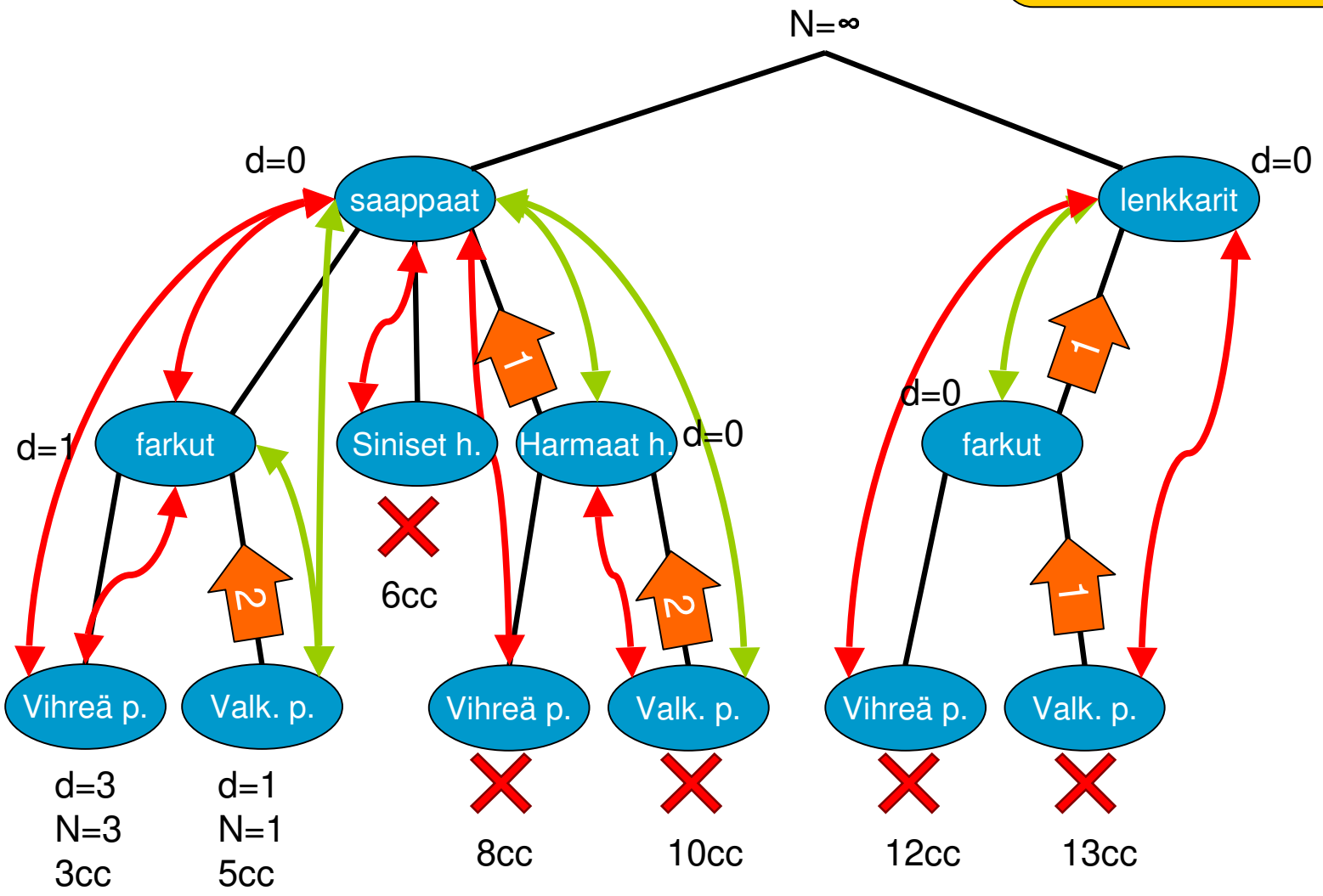
## PCSP, maximal solution

- Backjumping muistaa syvimmän tason, jota vastaan käsitelty muuttuja oli ristiriidassa.
- Kun kaikki arvot on käsitelty “hypätään” tälle tasolle tutkimatta muuttujia väliin jäävillä tasoilla.
- Näin voidaan tehdä, koska käsitelty muuttuja johtaisi ristiriitaan riippumatta “välitasojen” instantiaatioista.

# Backjumping

## PCSP, maximal solution

13 rajoite-tarkistusta



# Backjumping

## PCSP, maximal solution

- Rajoitteiden osittainen täyttäminen johtaa kuitenkin siihen, että välitasoilla on voitu instantioida tarpeettomasti ristiriitaisesti, jolloin raja-arvo tulee vastaan myöhemmin.
- Tämän vuoksi PCSP:ssä haku palaa tarvittaessa vain syvimmällä olevaan epäkonsistenttiin instantiaatioon asti.

# Backmarking
## CSP

- Backmarking on tekniikka, jolla pyritään välttämään redundantit rajoitteiden tarkistukset.
- CSP-versiossa käytetään kahta taulukkoa
  - ***Mark***-taulukkoon, kohtaan *(x,v)* merkitään ylin taso, jota vastaan muuttujan **x** arvon **v** havaittiin olevan ristiriidassa
  - ***BackTo***-taulukkoon kohtaan *(x)* merkitään ylin taso, jolle on peruutettu muuttujan **x** viimeisimmän tutkimisen jälkeen
- Vertaamalla taulujen arvoja voidaan päätellä seuraavaa
  - Mark[ x ][ v ] < BackTo[ x ]
    v tulee olemaan ristiriidassa, koska peruutus ei ole saavuttanut tasoa jota vastaan ristiriita löytyi
  - Mark[ x ][ v ] >= BackTo[ x ]
    v:n rajoitteet tulee testata tasolla BackTo ja siitä alaspäin

# Backmarking
## PCSP

- PCSP-versiossa käytetään neljää taulukkoa
  - ***LastMark***-taulukkoon, kohtaan *(x,v)* merkitään taso, jota vastaan muuttujan **x** arvo **v** aiheutti raja-arvon N saavuttamisen
  - ***FirstMark***-taulukkoon, kohtaan *(x,v)* merkitään taso, jolta löytyy ylin muuttujan **x** arvosta **v** aiheutuva inkonsistenssi
  - ***BackTo***-taulukkoon kohtaan *(x)* merkitään ylin taso, jolle on peruutettu muuttujan **x** viimeisimmän tutkimisen jälkeen
  - ***Inconsistencies***-taulukkoon, kohtaan *(x,v)* merkitään muuttujan **x** arvon **v** aiheuttamien inkonsistenssien lukumäärä
- Vertaamalla taulujen arvoja voidaan päätellä seuraavaa
  - LastMark[ x ][ v ] < BackTo[ x ]
    v tulee olemaan ristiriidassa, koska peruutus ei ole saavuttanut tasoa jota vastaan ristiriita löytyi
  - **Muut vaihtoehdot muuttuvat monimutkaisemmiksi**

# Backmarking
## PCSP

- Vertaamalla taulujen arvoja voidaan päätellä seuraavaa
  - LastMark[ x ][ v ] >= BackTo[ x ]
    v:n rajoitteet tulee testata, aloitustaso määräytyy vertaamalla BackTo:n arvoa FirstMark:iin

    (a) BackTo[ x ] <= FirstMark[ x ][ v ]
    FirstMark:in yläpuolella olevat arvot ovat konsistentteja, joten voidaan toimia kuten CSP-tapauksessa ja tarkistaa alkaen tasolta BackTo

    (b) BackTo[ x ] > FirstMark[ x ][ v ]
    Aloitetaan tarkistus ensimmäisestä inkonsistentista arvosta, Firstmark Tämän yläpuolella kaikki arvot ovat yhä konsistentteja

# Backmarking
## PCSP

- Käytännössä ensimmäinen tapaus jakaantuu kahtia koska kyseessä on PCSP
  - LastMark[ x ][ v ] < BackTo[ x ]

    (a) Distance + Inconsistencies [ x ][ v ] >= N

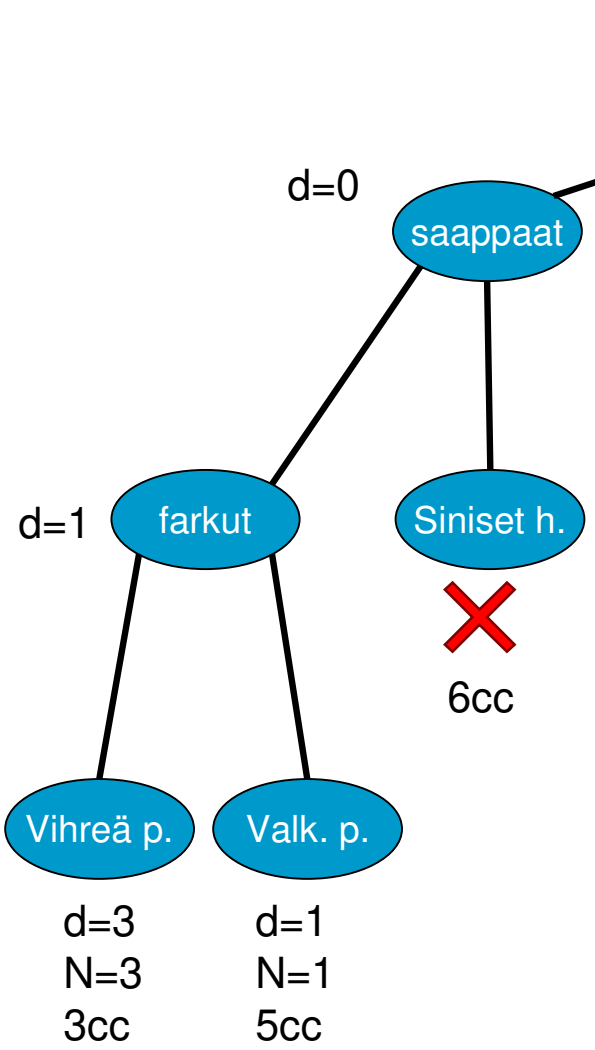
    Arvo v tulee olemaan ristiriidassa, joten v:n tutkimista ei jatketa

    (b) Muulloin…

    Lisätään nykyiseen etäisyyteen Inconsistencies[ x ][ v ] ja jatketaan testausta tason LastMark alapuolelta. Tällä tasolla summauksen tulos pätee, sillä kaikki LastMark:in yläpuolella olevat inkonsistenssit ovat yhä hakupolussa.

# Backmarking

## PCSP, maximal solution




FirstMark

| | 1 | 2 | 3 |
|---|---|---|---|
| 2 | 1 | 1 | |
| 3 | 1 | 2 | |

BackTo

| | |
|---|---|
| 2 | 1 |
| 3 | 2 |

LastMark

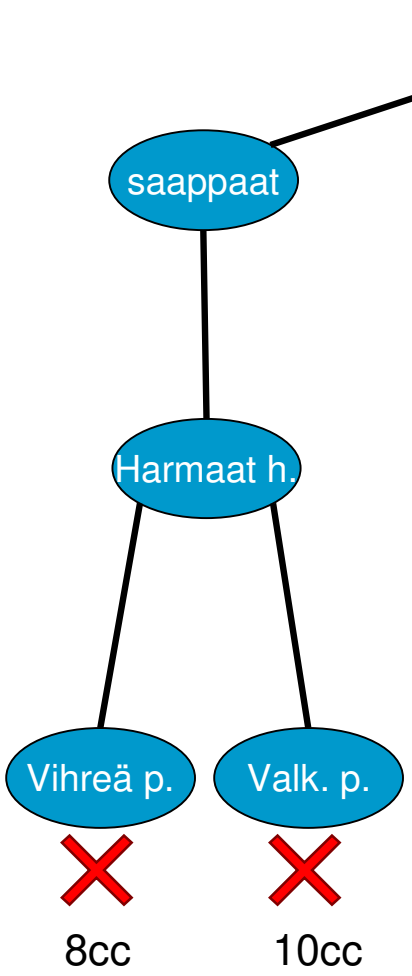
| | 1 | 2 | 3 |
|---|---|---|---|
| 2 | 1 | 1 | |
| 3 | 2 | 2 | |

Inconsistencies

| | 1 | 2 | 3 |
|---|---|---|---|
| 2 | 1 | 1 | |
| 3 | 2 | 1 | |

# Backmarking

## PCSP, maximal solution




FirstMark

| | 1 | 2 | 3 |
|---|---|---|---|
| 2 | | | 1 |
| 3 | 1 | 2 | |

BackTo

| | |
|---|---|
| 2 | 1 |
| 3 | 1 |

LastMark

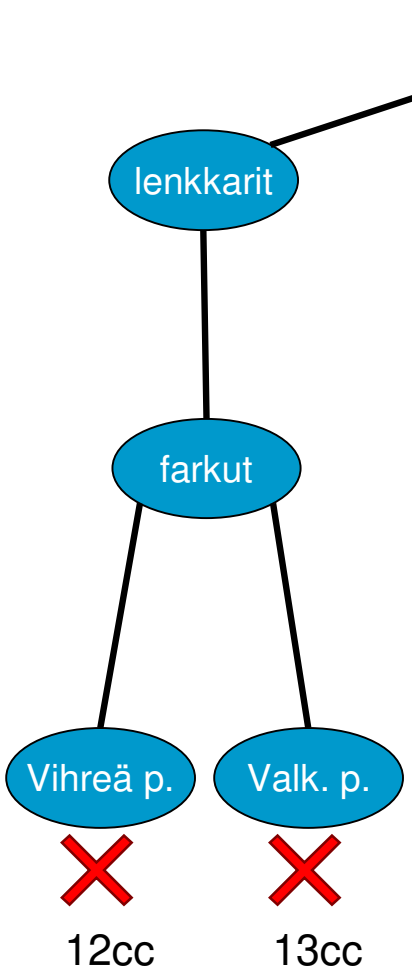
| | 1 | 2 | 3 |
|---|---|---|---|
| 2 | | | 1 |
| 3 | 1 | 2 | |

Inconsistencies

| | 1 | 2 | 3 |
|---|---|---|---|
| 2 | | | 0 |
| 3 | 1 | 1 | |

# Backmarking

## PCSP, maximal solution




FirstMark

| | 1 | 2 | 3 |
|---|---|---|---|
| 2 | 1 | | |
| 3 | 1 | 1 | |

BackTo

| | 1 | 2 | 3 |
|---|---|---|---|
| 2 | 1 | | |
| 3 | 2 | | |

LastMark

| | 1 | 2 | 3 |
|---|---|---|---|
| 2 | 1 | | |
| 3 | 1 | 1 | |

Inconsistencies

| | 1 | 2 | 3 |
|---|---|---|---|
| 2 | 0 | | |
| 3 | 1 | 1 | |

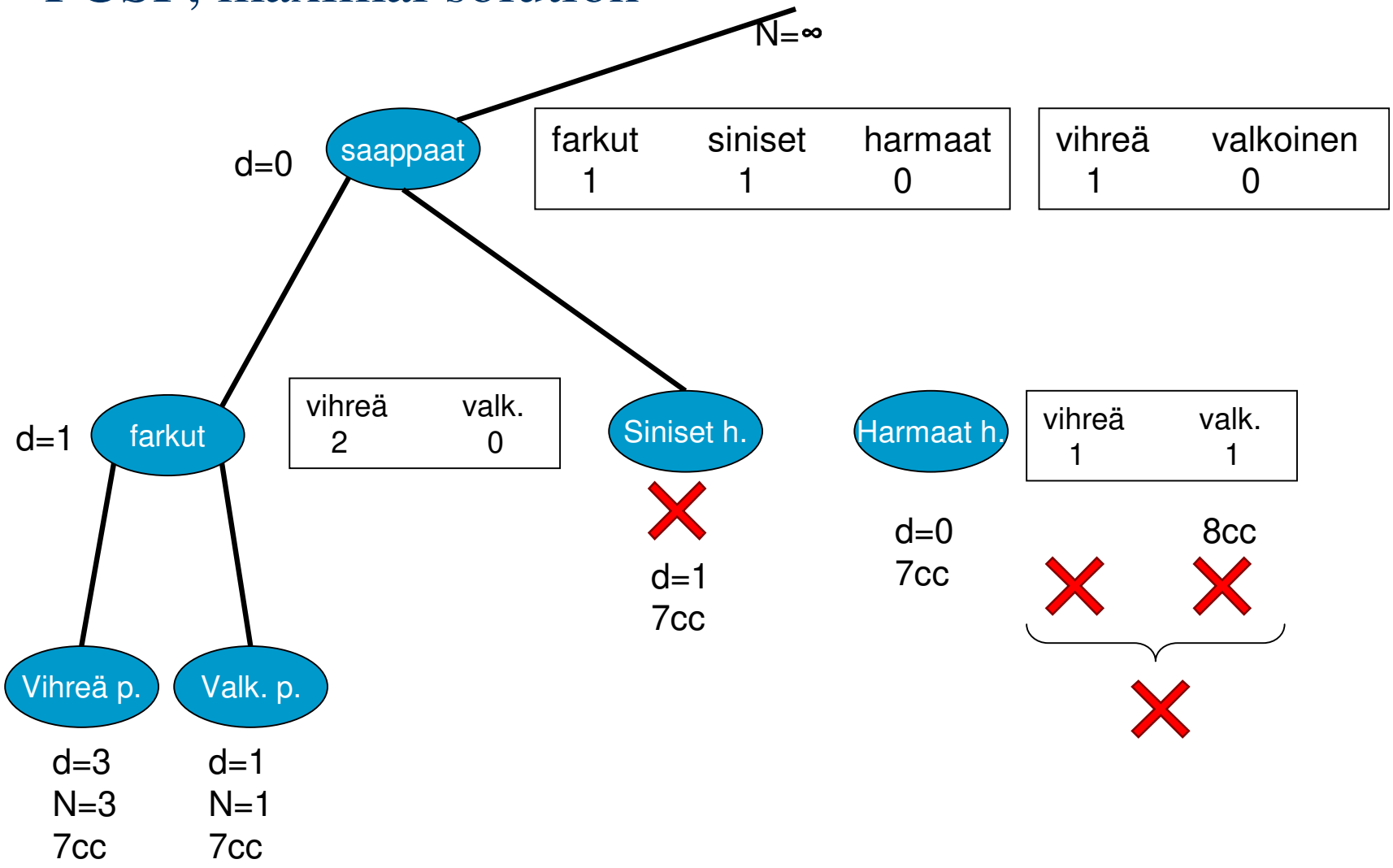
# Prospektiiviset tekniikat

# PCSP Forward Checking

- Karsinnan sijaan lasketaan inkonsistenssien lukumäärää
- Vasta kun lukumäärä ylittää raja-arvon hakupuun haara voidaan karsia

# Forward Checking

## PCSP, maximal solution



# Muuttujien järjestäminen

- Haku tehostuu, jos hyvä, mahdollisimman tiukan rajan N asettava ratkaisu löytyy mahdollisimman aikaisin.
- Muuttujien läpikäynti mahdollisimman vähän rajoittavasta rajoittavampaan päin aikaistaa hyvän ratkaisun löytymistä, jonka jälkeen N alkaa karsia tehokkaasti.

# Muuttujien järjestäminen

- Järjestämisehtona voidaan käyttää mm. kaari-inkonsistenssien arvoja.
- Nämä voidaan laskea ennalta ajassa $O(cd^2)$, missä n on muuttujien määrä ja d suurimman domainin koko.
- Käytännössä lasketaan jokaisen muuttujan jokaiselle arvolle *a* jokaisen rajoitteen suhteen jossa *a*:n muuttuja on osallisena, niiden domainien määrä, jotka instantiaatio *a* tyhjentää

# Yhteenveto

- PCSP menetelmiä käytetään kun halutaan osaratkaisuja
  - kun laskenta-aika on aikarajoitettu
  - kun osaratkaisukin riittää
  - kun ongelma on liian laaja
  - kun ongelmalla ei ole lainkaan ratkaisua CSP ongelmana

# Yhteenveto

- Tavallisten rajoiteongelmien ratkaisumenetelmillä on olemassa analogisia menetelmiä PCSP:ien ratkaisemiseen
- Sekä retrospektiivisia, että prospektiivisia tekniikoita
- PCSP tekniikoissa sallitaan inkonsistensseja hakupolussa. Puuta karsitaan kun konsistenssien määrä ylittää määritellyn tai saavutetun raja-arvon.
- Hakupolun sisältämät inkonsistenssit aiheuttavat muutoksia siihen, kuinka hakupuussa voidaan liikkua. Käytännössä tehokkuus yleensä kärsii CSP:ihin nähden.