4.2. Rajoiteongelmat [RN95, luvut 3.7, 4.2 ja 4.4] [Nil98, luvut 11.3-4]

Rajoiteongelmissa luvun 3 maaliehdon tiedetään jakautuvan konjunktioksi paikallisia ehtoja (”ehto 1 **ja** ehto 2 **ja** . . . ”) joita voidaan tarkastella osin erikseen.

Kuvan 38 mukainen 8 kuningattaren ongelma voidaan muotoilla *rajoiteongelmana (Constraint Satisfaction Problem, CSP)* seuraavasti:

1. Tavoitteena on etsiä sopivat arvot *muuttujille* $X_1, \ldots, X_8$.
   Muuttujan X_i arvo kertoo silloin sarakkeelle i asetetun kuningattaren i rivin.
   Muuttujia on aina äärellinen määrä.
2. Kukin muuttuja saa arvokseen rivinumeron
   $$X_i \in D_i = \{1, \ldots, 8\} .$$
   Joukkoa D_i kutsutaan muuttujan X_i *arvoalueeksi (domain)*.
   Arvoalue voi yleisessä tapauksessa olla myös ääretön kuten $\mathbb{Q}$ tai jopa jatkuva kuten $\mathbb{R}$.
   Jokaista tällaista *yhtä* muuttujaa koskevaa ehtoa ”$X_i \in D_i$” kutsutaan *unaariseksi rajoitteeksi (unary constraint)*.
3. Kaikille $i < j$ pitää päteä, että kuningattaret ruuduissa X_i ja X_j eivät uhkaa toisiaan.
   Jokaista tällaista *kahta* muuttujaa koskevaa ehtoa kutsutaan *binääriseksi* rajoitteeksi.
   Sitä voidaan ajatella karteesisen tulon $D_i \times D_j$ sellaisena osajoukkona $R_{i,j}$, joka luettelee kaikki sallitut parit.

Rajoitteet voivat yleisessä tapauksessa käsittää myös 3 tai useampia muuttujia kerrallaa.

Jos kuitenkin kaikki arvoalueet ovat äärellisiä, niin useampimuuttujaiset rajoitteet voidaan korvata sopivilla uusilla unaari- ja binäärirajoitteilla.

Rajoite voi olla

ehdoton ***(absolute)*** jolloin ratkaisun on pakko toteuttaa se (kuten 8 kuningattaren ongelmassa)

suositus ***(preference)*** jolloin voidaan hyväksyä myös sellainen ratkaisu, joka ei sitä täytä, jos parempaan ei pystytä.

Esimerkiksi laitoksen opetusohjelmaa laadittaessa

ehdoton rajoite on, että samaan saliin ei voi sijoittaa samaan aikaan kahta eri luentoa

suositus on, että cum laude -tason pakollisten kurssien luentoja ei sijoitettaisi samaan aikaan.

Keskitytään ratkomaan *ehdottomista äärellisistä unaari- ja binäärirajoitteista koostuvia ongelmia* oppimillamme menetelmillä.

Luvun 3.1 systemaattisia etsintämenetelmiä voidaan soveltaa seuraavasti:

- Tavoitteena on löytää arvot äärelliselle määrälle muuttujia $X_1, \ldots, X_N$.
  Tilannetta voi ajatella etsintäpuuna, jossa syvyydellä d etsitään arvoa muuttujalle X_{d+1}, ja mahdolliset teot ovat D_{d+1}.
  Voidaan käyttää luvun 3.1.3 *syvyyssuuntaista etsintää*:
  nyt se pysähtyy itsestään viimeistään syvyydellä N (eli on täydellinen).
  Kuningataresimerkissämme $N = 8$.
- Olkoon syvyysetsintä jo antanut arvot muuttujille X_i, missä $1 \leq i \leq j$.
  Valitaan seuraavalle muuttujalle X_{j+1} sopivaa arvoa $V \in D_{j+1}$.

Ensin voidaan tutkia kaikki binäärirajoitteet muuttujien X_i ja X_{j+1} välillä, ja *karsia* joukosta D_{j+1} pois ne ehdokkaat V jotka rikkoisivat jonkin niistä.

Jos yhtään ehdokasta ei jää jäljelle, voidaan *peruuttaa (backtrack)* etsinnässä jo nyt: koko solmusta X_{j+1} alkava alipuu voidaan jättää tutkimatta, koska jo siihen vievä polku on havaittu mahdottomaksi.

Näin rajoiteongelmissa voimme kysyä väliaikatietoja "Onko tämä keskeneräinen ratkaisu oikein tähän saakka?". Se tehostaa etsintää yleisiin ongelmiin verrattuna.

Esimerkissämme tämä strategia tarkoittaa, että kuningatar $j + 1$ laitetaan aina sellaiseen ruutuun, jota mikään jo laitetuista kuningattarista $1, \ldots, j$ ei uhkaa.

- Kääntäen voimme kuningatarta $j + 1$ asettaessa merkitä samalla ne ruudut, joita se uhkaa.

  Silloin voimme katsoa, jääkö laudalle tarpeeksi uhattomia ruutuja lopuille kuningattarille $j + 2, \ldots, 8$, kun kuningatarten $1, \ldots, j + 1$ uhkaamat ruudut on poistettu.

  Tätä strategiaa kutustaan *eteenpäin tarkistamiseksi (forward checking)*.

  Sen avulla voidaan käytännössä ratkoa esimerkkiongelmamme laajennuksia aina noin tapaukseen $N = 30$ saakka (eli sovittaa N kuningatarta $N \times N$ ruudun laajennetulle shakkilaudalle).

  Kun siis valitaan jokin $X_{j+1} = V$, niin samalla poistetaan kaikista binäärirajoitteista $R_{j+1,j+k}$, missä $k > 1$, kaikki parit jotka eivät ole muotoa $\langle V, W\rangle$ (missä siis $W \in D_{j+k}$). Jos jokin niistä tyhjenee, niin voidaan taas peruuttaa.

- Tämä poistostrategia on *kaarikonsistentiksi (arc consistency) pakottamista*:

  Etsinnän tila on kaarikonsistentti jos jokaisen muuttujan X_i arvoalueessa D_i on edelleen jäljellä jokin sellainen arvo V_i, joka kelpaa kaikille muuttujaan X_i liittyville rajoitteille.

  Tai kääntäen, $V_i \notin D_i$ jos yksikin muuttujaan X_i liittyvä rajoite kieltää muuttujalta X_i arvon V_i.

  Toisin sanoen, D_i sisältää joka hetki vain ne muuttujan X_i

arvot V_i jotka ovat yhä *mahdollisia, kun ongelmaa katsellaan vain muuttujan $X_i$lähistöltä*, ei kokonaisuutena.

– Monimutkaisemmissa ongelmissa lähistöksi voidaan ottaa yhden sijasta $k = 2, 3, 4, \ldots$ muuttujaa.
– Eräs tapa ratkoa rajoiteongelma(luokka)a onkin analysoida tarvittavan lähistön kokoa k, ja pakottaa ongelmat k-konsistenteiksi. Tarpeeksi suurella k voidaan jopa *välttää peruutus*.
– Toisaalta k on pakottamisen aikavaatimuksen eksponentissa. Lisäksi "tarpeeksi suuri" k voi lähestyä muuttujien lukumäärää N.
– Yleensä tyydytäänkin 2- eli *polku*konsistenssiin *(path c.)*.

- Kaarikonsistentiksi pakottaminen tapahtuu poistamalla mahdottomia arvoja alueista D_i ja mahdottomia pareja rajoitteista $R_{i,j}$ niin pitkään kuin mahdollista.

  Kukin poisto voi mahdollistaa uusia poistoja.

  Jos jokin joukko tyhjenee, ei kyseisellä rajoiteongelmalla ole ratkaisuja.

  Tämä pakottamismenetelmä on eräs *rajoitteiden levittämisen (constraint propagation)* erikoistapaus.

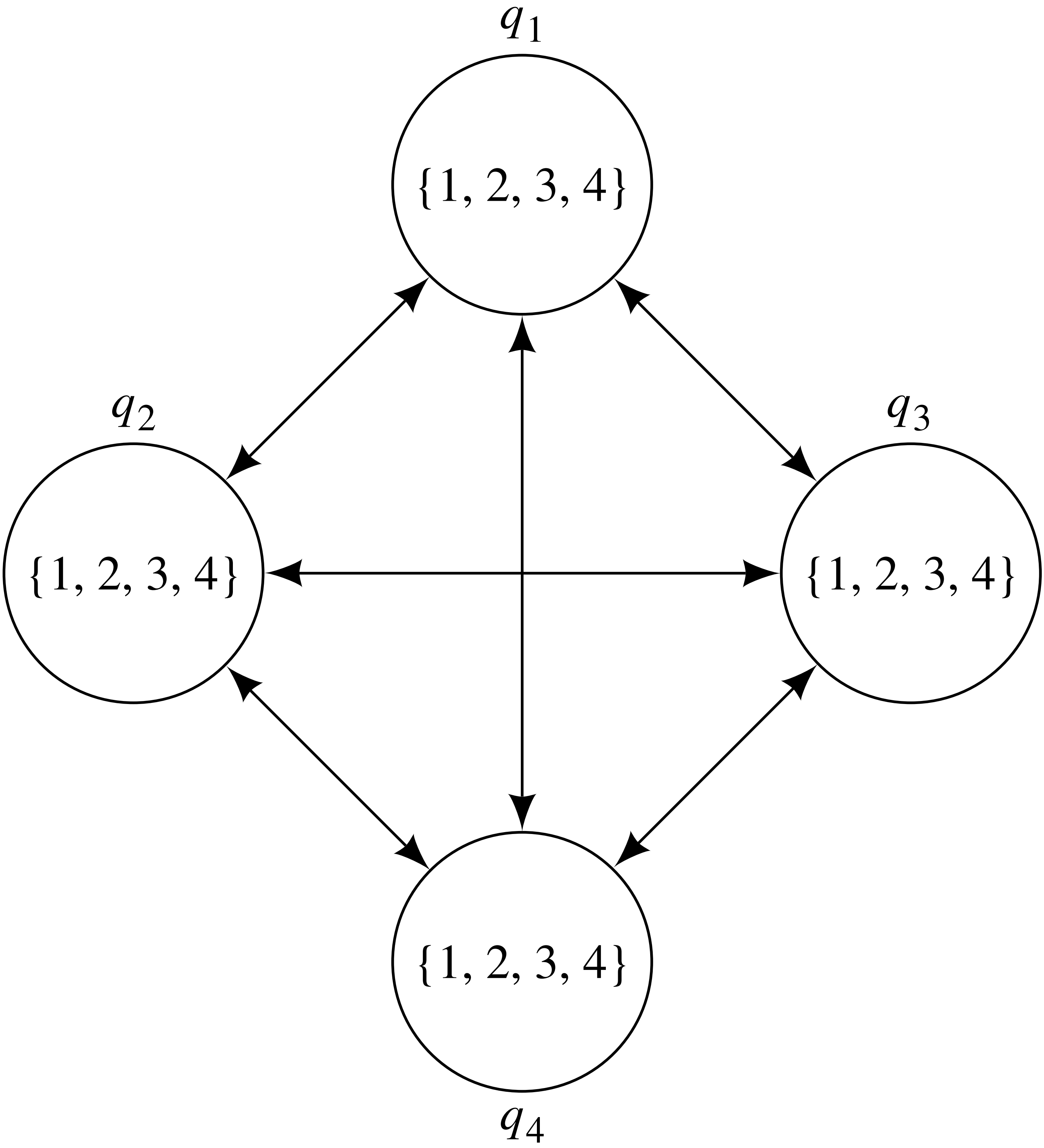
- Yleisesti voidaan rajoiteongelmaa ratkoa rajoitteita levittämällä vaikkapa seuraavasti:
  1. Aloitetaan ongelmasta, jossa tiedetään vain unaarirajoitteet, eli kukin $R_{i,j} = D_i \times D_j$.
  2. Lisätään muuttujien X_i ja X_j välinen binäärirajoite $R'_{i,j} \subseteq D_i \times D_j$ poistamalla pareista $R_{i,j}$ kaikki ne jotka eivät kuulu rajoitteeseen $R'_{i,j}$.
  3. Poistoja jatketaan valituksi k-konsistentiksi saakka.
  4. Sama toistetaan kaikille binäärirajoitteille.
  5. Lopuksi suoritetaan syvyyssuuntainen etsintä jäljelle jääneessä rajoiteongelmassa.

- Rajoitteiden levittämistä käytetään myös muiden etsintämenetelmien
  – esivaiheena
  – osana

  karsimaan etsintäavaruutta ennen raskasta hakua.

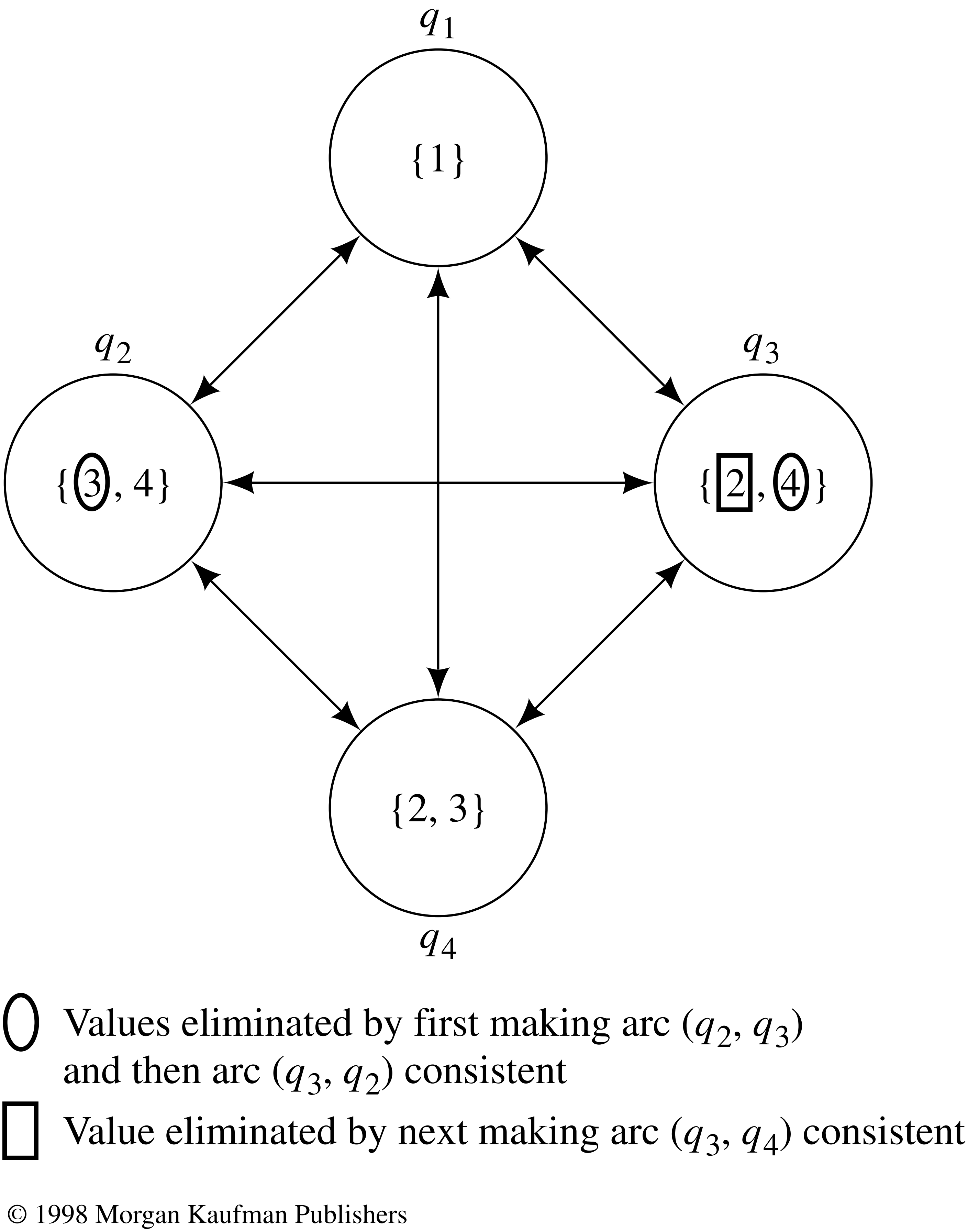
- Tarkastellaan esimerkkinä 4 kuningattaren sijoittelua 4×4-shakkilaudalle.

Alkutilanne on kuvassa 41: kuningattaret ovat muuttujina $q_1, \ldots, q_4$ joille sallitaan rivit $1, \ldots, 4$.



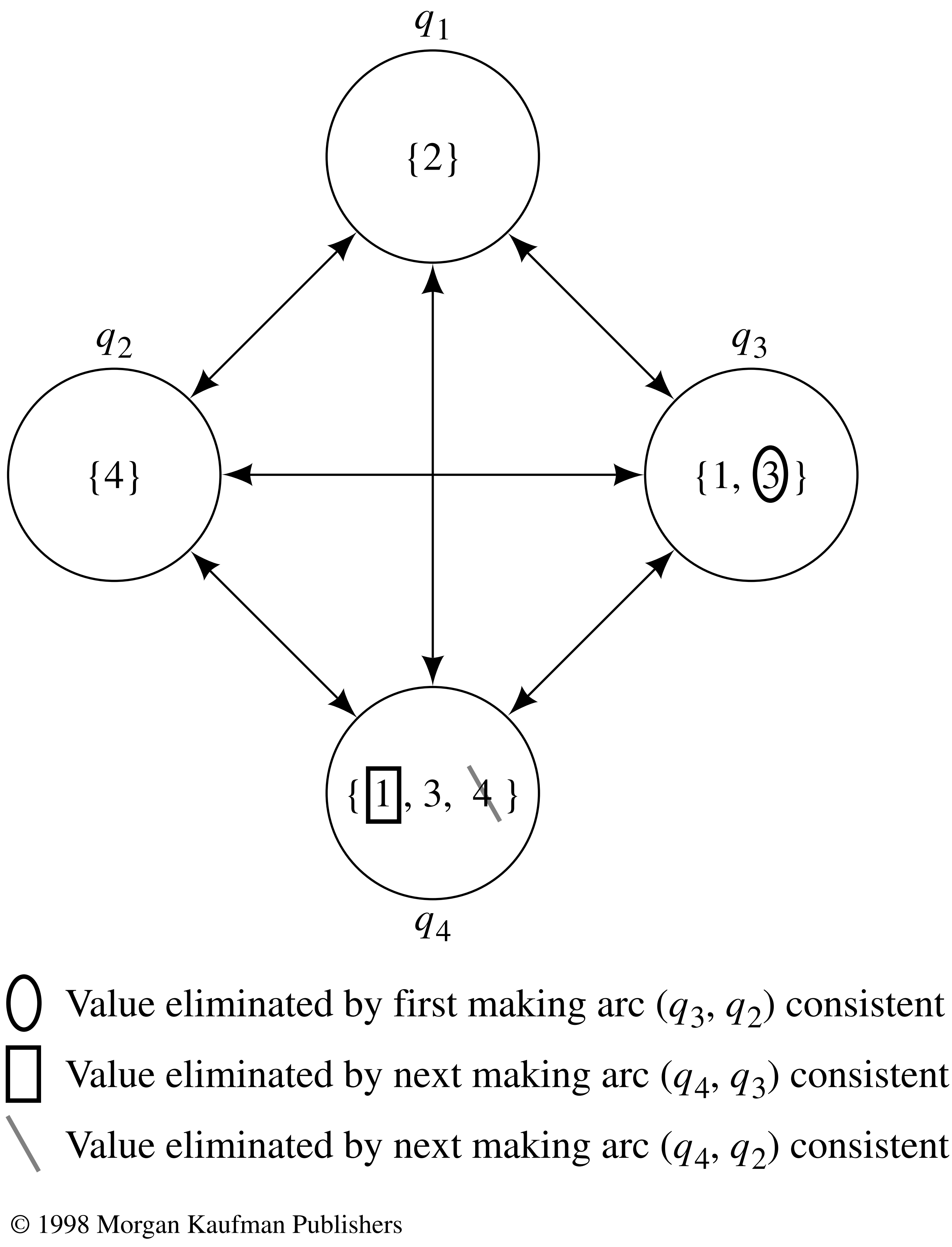
Kuva 41: 4 kuningattaren rajoiteongelman alkutila [Nil98, kuva 11.4].

Jos valitaan ensin $q_1 = 1$, niin päädytään kuvan 42 kautta umpikujaan.



Kuva 42: 4 kuningattaren rajoiteongelma kun $q_1 = 1$ [Nil98, kuva 11.5].

Peruutetaan valintaan $q_2 = 2$, josta kuvan 43 kautta ratkaisuun.

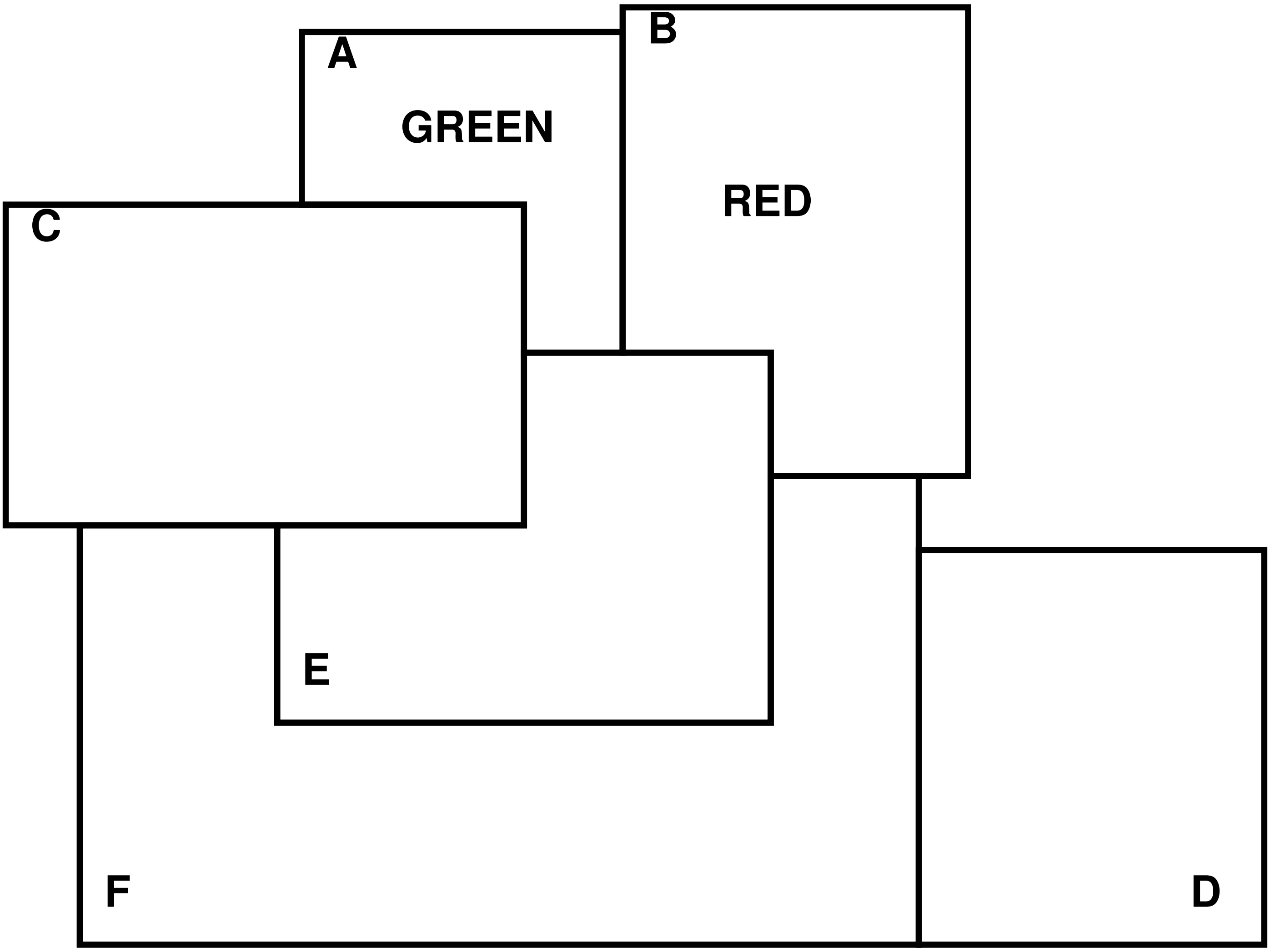


Kuva 43: 4 kuningattaren rajoiteongelma kun $q_1 = 2$ [Nil98, kuva 11.6].

- Rajoiteongelmissa käytetään heuristiikkoja *valitsemaan seuraava muuttuja jolle määrätään arvo*. (Muuttujia ei siis enää käsitellä ennalta kiinnitetyssä järjestyksessä.)

  Tämä on erilaista heuristista etsinnän ohjausta kuin luvussa 3.2.

- Tarkastellaan esimerkiksi ongelmaa, jossa seuraava kuvio pitää värittää 3 värillä siten, että naapuruksilla on aina eri värit. Kaksi aluetta A ja B on jo väritetty vihreäksi (green) ja punaiseksi (red).



Kuva 44: Kolmiväriongelma [RN95, kuva 4.9].

- Seuraavaksi kannattaa värittää alue E siniseksi (blue), koska sillä ei ole enää muita vaihtoehtoja. Sitten:

  1. C voi olla vain punainen.
  2. F voi olla vain vihreä.
  3. D voi olla joko sininen tai punainen muttei vihreäksi.

  Tämä on ääriesimerkki *eniten rajoitettu muuttuja (most-constrained-variable) -heuristiikasta*:

Valitse käsittelyyn sellainen muuttuja X_j, jonka arvoalueessa D_j on *vähiten mahdollisia vaihtoehtoja* jäljellä.

Tämä valinta voidaan tehdä luontevasti eteenpäin tarkistettaessa.

Intuitiivisesti tämä heuristiikka pyrkii minimoimaan etsinnän *välitöntä* haarautumista valitsemalla aina vähiten haaroja synnyttävän vaihtoehdon.

Esimerkiksi kuvan 38 kuningatarongelmassa päästään aina tapauksiin $N = 100$ saakka, kun valitaan aina sellainen sarake j, jolla on vähiten vielä mahdollisia ruutuja.

- Toinen haarautumista ehkäisevä heuristiikka on *eniten rajoittava (most-constraining)* muuttuja -heuristiikka: valitse käsittelyyn sellainen muuttuja, joka on mukana mahdollisimman monessa binäärirajoittessa vielä käsittelemättömien muuttujien kanssa.

  Intuitiivisesti tämä heuristiikka pyrkii minimoimaan etsinnän *tulevaa* haarautumista valitsemalla aina sellaisen vaihtoehdon, jonka jälkeen myöhemmillä vaihtoehdoilla on vähän haaroja jäljellä.

- Kun muuttuja X_j on valittu jommalla kummalla heuristiikalla, on vielä valittava sille jokin mahdollinen arvo $V \in D_j$.

  Vähiten rajoittava arvo (least constraining value) -heuristiikka valitsee muuttujalle X_j sellaisen arvon $V \in D_j$, joka poistaa vähiten mahdollisia arvoja vielä käsittelemättömiltä muuttujilta.

  Valitaankin kuvassa 44 ensin käsittelyyn alue C (eikä E). Jos se väritetään

  siniseksi niin alueelle E ei jää yhtään mahdollista väriä.

  Alueelle F jää vain vihreä.

  punaiseksi niin alueelle E jää sininen.

  Alueelle F jää vihreä ja sininen (jonka E kieltää vasta myöhemmin).

  Valitaan siis punainen väri.

  Kuvan 38 kuningatarongelmassa päästään aina tapauksiin $N = 1\ 000$ saakka, jos valitaan sarakkeen j ruuduista se, joka synnyttää vähiten uusia uhkauksia vielä käsittelemättömille riveille.

Luvun 4.1 iteratiiviset parannusmenetelmät antavat *heuristisia korjausmenetelmiä (heuristic repair methods)*: ne ratkovat rajoiteongelmia