Inteligentni sistemi

Problemi zadovoljenja ograničenja

Constraint Satisfaction Problems (CSP)

Uvod

- Stanje ne posmatramo kao male crne kutije – nove mogućnosti
- Bolje razumevanje strukture i složenosti problema
- Za svako stanje: skup promenljivih, od kojih svaka ima vrednost
- Problem je rešen kada svaka promenljiva ima vrednost koja zadovoljava sva postavljena ograničenja
- Problemi zadovoljenja ograničenja - Constraint Satisfaction Problems
- Koriste prednosti strukture stanja i koriste heuristiku opšte namene
- Eliminacija velikih delova oblasti pretraživanja – primena ograničenja

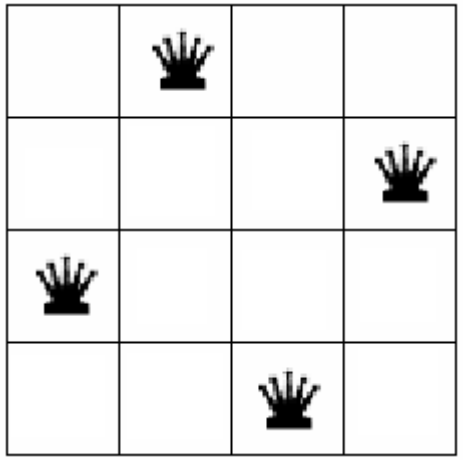
Definisanje problema

- Problem se sastoji od 3 komponente X, D_o i C
- X je skup promenljivih $\{X_1, \ldots, X_n\}$
- D_o je skup domena, $\{D_{01}, \ldots, D_{0n}\}$, po jedan za svaku promenljivu
- C je skup ograničenja koja određuju dozvoljene kombinacije vrednosti
- Svaki domen D_{oi} sastoji se od skupa dozvoljenih vrednosti, $\{v_1, \ldots, v_k\}$, za promenljivu X_i
- Svako ograničenje se sastoji od para (opseg, rel)
  - opseg – n-torka promenljivih koje učestvuju u ograničenju
  - rel – relacija koja definiše vrednosti koje te vrednosti mogu da prime

Definisanje problema

- Relacija se može predstaviti kao eksplicitna lista svih n-torki vrednosti ili kao apstraktna relacija koja podržava testiranje da li je n-torka pripadnik relacije i nabraja pripadnike relacije
- X_1 i X_2 imaju isti domen {A, B}, ali su različite
- (X_1, X_2), [(A, B), (B, A)] ili (X_1, X_2), $X_1 \neq X_2$
- Potrebno je da definišemo oblast stanja i notaciju rešenja
- Dodela koja ne krši nijedno ograničenje naziva se konzistentna ili legalna dodela
- Potpuna dodela – svakoj promenljivoj dodeljena vrednost
- Rešenje – konzistentna potpuna dodela
- Delimična dodela – dodate vrednosti samo nekim promenljivama

Primer N-kraljica



1. način:

Svakom polju na tabli odgovara jedna promenljiva (X_{ij}).

X_{ij} označava šta se dešava na datom polju. Domen:{0,1}

Uslovi:

$$\forall i, j, k \ \left(X_{i,j}, X_{i,k}\right) \in \{(0,0),(0,1),(1,0)\}$$

$$\forall i, j, k \ \left(X_{i,j}, X_{k,j}\right) \in \{(0,0),(0,1),(1,0)\}$$

$$\forall i, j, k \ \left(X_{i,j}, X_{i+k,j+k}\right) \in \{(0,0),(0,1),(1,0)\}$$

$$\forall i, j, k \ \left(X_{i,j}, X_{i+k,j-k}\right) \in \{(0,0),(0,1),(1,0)\}$$

$$\sum_{i,j} X_{i,j} = N$$

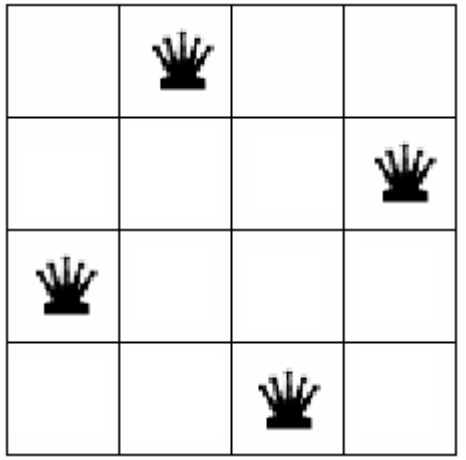
Primer N-kraljica

2. metoda:

Promenljive: Q_k (jedna promenljiva po vrsti)

Domen:{1, 2, 3, ..., N}

Svaka vrsta sadrži kraljicu (Q_1 je promenljiva za prvu vrstu, označava gde se kraljica nalazi u prvoj vrsti).



Uslovi:

Implicitno: $\forall i, j \ \ (Q_i, Q_j)$ ne napadaju se

Eksplicitno: $(Q_1, Q_2) \in \{(1,3),(1,4),(2,4),\ldots\}$

Primer: Bojenje mape

- Australija – svaka oblast jednom bojom, nikada susedne istom
- Promenljive: WA, NT, Q, NSW, V, SA, T (Zapadna, Severna, Kvinslend, Novi Južni Vels, Viktorija, Južna, Tasmanija)
- Domeni: D={crveno, zeleno, plavo}
- Uslovi: susedne teritorije ne mogu biti iste boje. Devet mesta gde se regioni graniče, pa ima devet ograničenja

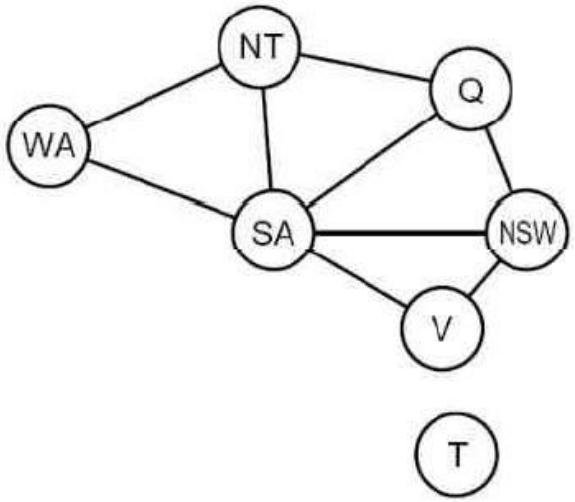
  Implicitno: WA ≠ NT

  Eksplicitno: (WA, NT) € {(crveno,zeleno), (crveno,plavo),...}
- Rešenje je kad su svim promenljivama dodelje vrednosti tako da svi uslovi budu zadovoljeni, npr.:

  {WA=crveno, NT=zeleno, Q=crveno, NSW=zeleno, V=crveno, SA=plavo, T=zeleno}
- Rešenje nije jednoznačno, postoji i drugo ispravno dodeljivanje.

Graf ograničenja

- Čvorovi odgovaraju promenljivama
- Veze povezuju bilo koje 2 promenljive koje učestvuju u ograničenju
- Zašto CSP?
  - Prirodno predstavljanje za širok spektar problema
  - Ako je već razvijen lakše rešiti problem sa tim sistemom
  - Brže, jer se eliminišu veliki uzorci prostora pretraživanja
  - Kada smo odabrali SA=plava, nijedna od 5 susednih ne može biti plava
  - Umesto 3^5=243 dodele dobijamo 2^5=32 dodele, ušteda 87%
  - Moguće je da se fokusira na promenljive koje su bitne, koje krše ograničenje



Varijacije

- Diskretni konačni domeni (bojenje mapa, 8 kraljica, ...)
- Diskretni beskonačni domeni (skup celih brojeva) – jezik ograničenja, linearna ograničenja
- Kontinualni domeni (kod operacionih istraživanja)
  - raspoređivanje eksperimenata na svemirskom teleskopu Hubble
  - početak i kraj svakog posmatranja i manevrisanja su promenljive sa kontinualnim vrednostima
  - Ograničenja astronomska i prioriteta i snage
- Linearno programiranje – ograničenja su linearne jednačine ili nejednačine
- Tipovi ograničenja:
  - Unarno, ograničava se vrednost samojedne promenljive, SA ne vole zelenu boju
  - Binarno, svaki od uslova obuhvata (najviše) dve promenljive, graf mape
  - Globalno, proizvoljan broj promenljivih, Svirazličiti – sve promenljive različite vrednosti, Sudoku ili kriptoaritmetička slagalica, Hipergraf ograničenja
- Svako ograničenje konačnog domena može se redukovati na skup binarnih ograničenja, ako se uvede dovoljno pomoćnih promenljivih

Ostali tipovi ograničenja

- Ograničenje nvalue ($\{e_1, \ldots, e_n\}$, e) je zadovoljeno ako izrazi $e_1, \ldots, e_n$ uzimaju broj različitih vrednosti jednak vrednosti izraza e
  - nvalue($\{x_1, x_2, x_3\}$, 2) je zadovoljeno ako promenljive x_1, x_2 i x_3 uzimaju ukupno dve različite vrednosti.
  - Specijalan slučaj kada je e = n predstavlja Svirazličiti ograničenje
- Ograničenje count ($\{e_1, \ldots, e_n\}$, e) R e' predstavlja uslov da broj pojavljivanja vrednosti nekog izraza e u skupu vrednosti izraza $e_1, \ldots, e_n$ bude u odgovarajućoj aritmetičkoj relaciji R (=, ≠, ≤,<,>,≥) sa nekim izrazom e'
  - count($\{x_1, x_2, x_3, x_4\}$, 5) > 3 je zadovoljeno ako se vrednost 5 pojavljuje više od 3 puta u skupu promenljivih $\{x_1, x_2, x_3, x_4\}$
  - Sve promenljive x_1, x_2, x_3, x_4 moraju da uzmu vrednost 5.

Primer: kriptoaritmetika

- Promenljive: F, T, U, W, R, O, C_{10}, C_{100}, C_{1000}
- Domeni: {0,1,2,3,4,5,6,7,8,9}
- Uslovi: Svirazličiti (F,T,U,W,R,O)

```
  T W O
+ T W O
-------
F O U R
```

$O + O = R + 10 * C_{10}$

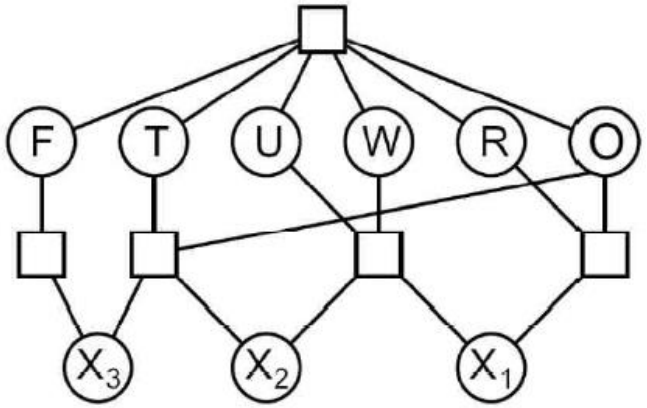
$C_{10}+W+W=U+10* C_{100}$

$C_{100}+T+T=O+10* C_{1000}$

$C_{1000}=F$

Kvadrati označavaju n-arna ograničenja

```
  867
+ 867
-----
 1734
```



Ograničenja preferensi

- Sve do sada primeri apsolutnih ograničenja
- U realnom svetu problemi ukazuju koja su rešenja poželjnija
- Profesori – nijedan ne može više časova istovremeno (apsolutno)
- Profesor D voli predavanja ujutru, profesor B voli popodne
- Profesor B predavanje od 16h, a profesor D od 14h - dozvoljeno
- Troškovi pojedinačnih dodela promenljivih
- Problemi optimizacije ograničenja – linearno programiranje

Zaključivanje

- Algoritam može da:
  - Pretražuje (izabere novu dodelu vrednosti)
  - Zaključuje (prostiranje ograničenja)
- Prostiranje ograničenja može zajedno sa pretraživanjem ili kao predobrada
- Lokalna konzistentnost – eliminisanje nekonzistentnih vrednosti kroz graf:
  - **Konzistentost čvora** – sve vrednosti u domenu promenljive zadovoljavaju unarna ograničenja promenljive
  - **Konzistentnost grane** – svaka vrednost u domenu promenljive zadovoljava binarna ograničenja promenljive
  - **Konzistentnost putanje** – pooštrava binarna ograničenja koristeći implicitna ograničenja koja su zaključena na osnovu gledanja trojki promenljivih

Konzistencija grane (arc consistency)

- X_i je grana-konzistentna u odnosu na X_j ako za svaku vrednost u trenutnom domenu D_{oi} postoji neka vrednost u domenu D_{oj} koja zadovoljava binarno ograničenje na grani (X_i, X_j)
- Mreža je grana-konzistentna ako je svaka promenljiva grana-konzistentna sa svakom drugom promenljivom
- $Y = X^2$ - (X,Y), {(0,0), (1,1), (2,4), (3,9)}; X/{0,1,2,3}, Y/{0,1,4,9}

- Grana X→Y je konzistentna ako i samo ako za *svako x* iz repa *postoji y* u vrhu kojem možemo dodeliti vrednost tako da ne prekršimo uslov.

Algoritam AC-3

- Održava red čekanja grana koje će se razmatrati
- U početku su sve grane, zatim se izbaci proizvoljna grana (X_i, X_j) i učini X_i grana-konzistentnom u odnosu na X_j
- Ako se ne promeni D_{oi} algoritam prelazi na sledeću granu
- Ali ako se smanji D_{oi} onda se redu dodaju sve grane (X_k, X_i), X_k sused X_i
- Ako je D_{oi} obradom smanjen na prazan – nema konzistentnog rešenja
- U suprotnom se nastavlja da se proverava pokušavajući da sklonimo vrednosti iz domena promenljivih, sve dok nema grana u redu
- Ostaje nam grana-konzistentan prostor - manji

Algoritam AC-3

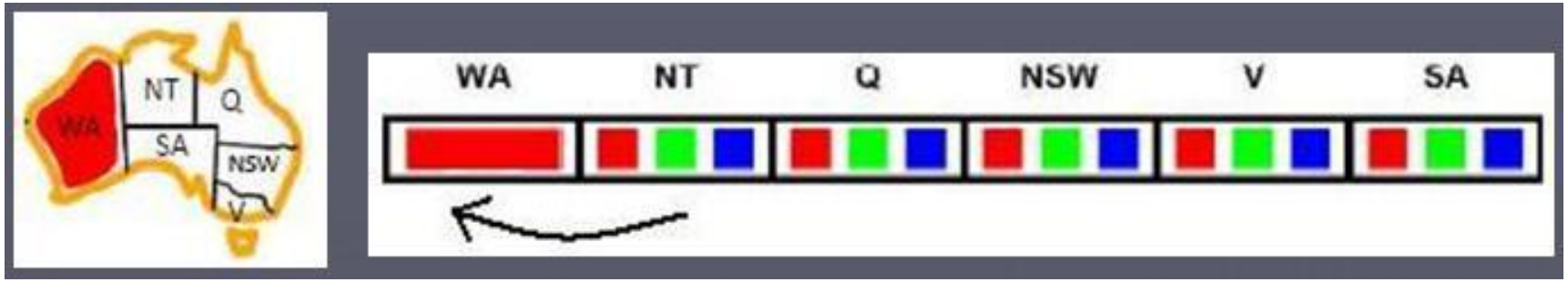
function AC-3(*csp*) **returns** the CSP, possibly with reduced domains
  inputs: *csp*, a binary CSP with variables $\{X_1, X_2, \ldots, X_n\}$
  local variables: *queue*, a queue of arcs, initially all the arcs in *csp*

  while *queue* is not empty
    $(X_i, X_j) \leftarrow$ REMOVE-FIRST(*queue*)
    if REMOVE-INCONSISTENT-VALUES(X_i, X_j) **then**
      for each X_k **in** NEIGHBORS[X_i] **do**
        add (X_k, X_i) to *queue*

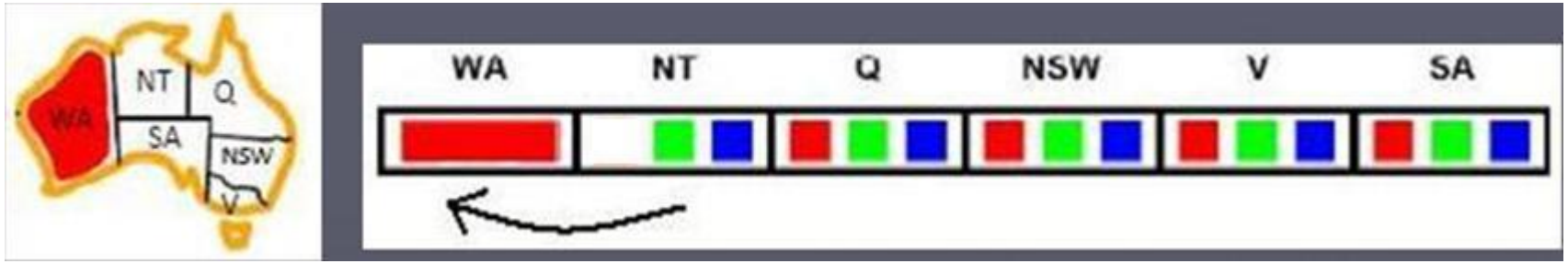
---

function REMOVE-INCONSISTENT-VALUES(X_i, X_j) **returns** true iff succeeds
  removed $\leftarrow$ *false*
  for each x **in** DOMAIN[X_i]
    if no value y in DOMAIN[X_j] allows (x,y) to satisfy the constraint $X_i \leftrightarrow X_j$
      then delete x from DOMAIN[X_i]; *removed* $\leftarrow$ *true*
  return *removed*

AC-3

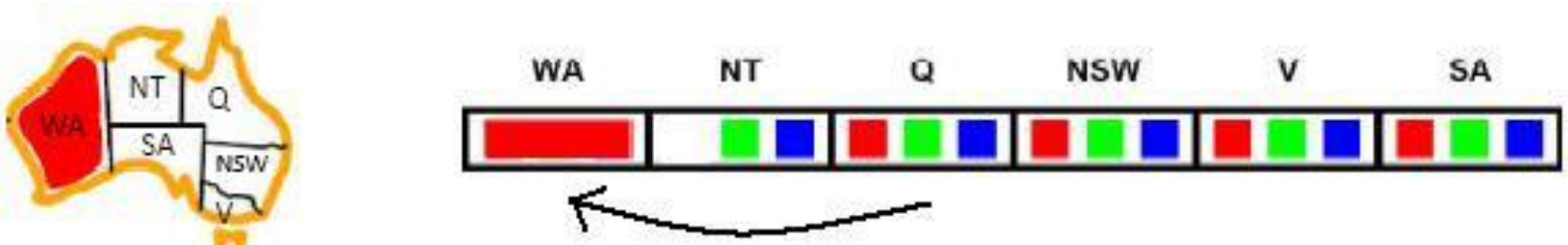


Označena grana nije konzistentna (crveno), ali je možemo učiniti konzistentnom (obrišemo crveno).



Uvek brišemo iz repa!!!

AC-3



- Označena grana je konzistentna (ne treba je učiniti konzistentnom) jer ni za jednu vrednost iz repa nijedan uslov neće biti povređen.

Algoritam AC-3

- Ako ima n promenljivih i svaka veličinu domena d i c binarnih ograničenja (grana)
- Svaka grana (X_k, X_i) može biti umetnuta u red samo d puta
- Provera konzistentnosti je $O(d^2)$ – vreme najgoreg slučaja $O(cd^3)$
- Moguće je proširiti konzistentnost grane da obrađuje n-arna ograničenja

Pretraživanje sa vraćanjem

- Mnogi problemi zadovoljenja ograničenja ne mogu se rešiti samo zaključivanjem
- Algoritmi pretraživanja sa vraćanjem sa delimičnim dodelama
- Algoritmi lokalnog pretraživanja sa potpunim dodelama
- Standardno pretraživanje sa ograničenom dubinom-broj listova $n!*d^n$
- Komutativnost – redosled izvršavanja skupa akcija ne utiče na rezultat
- CSP su komutativni – broj listova d^n

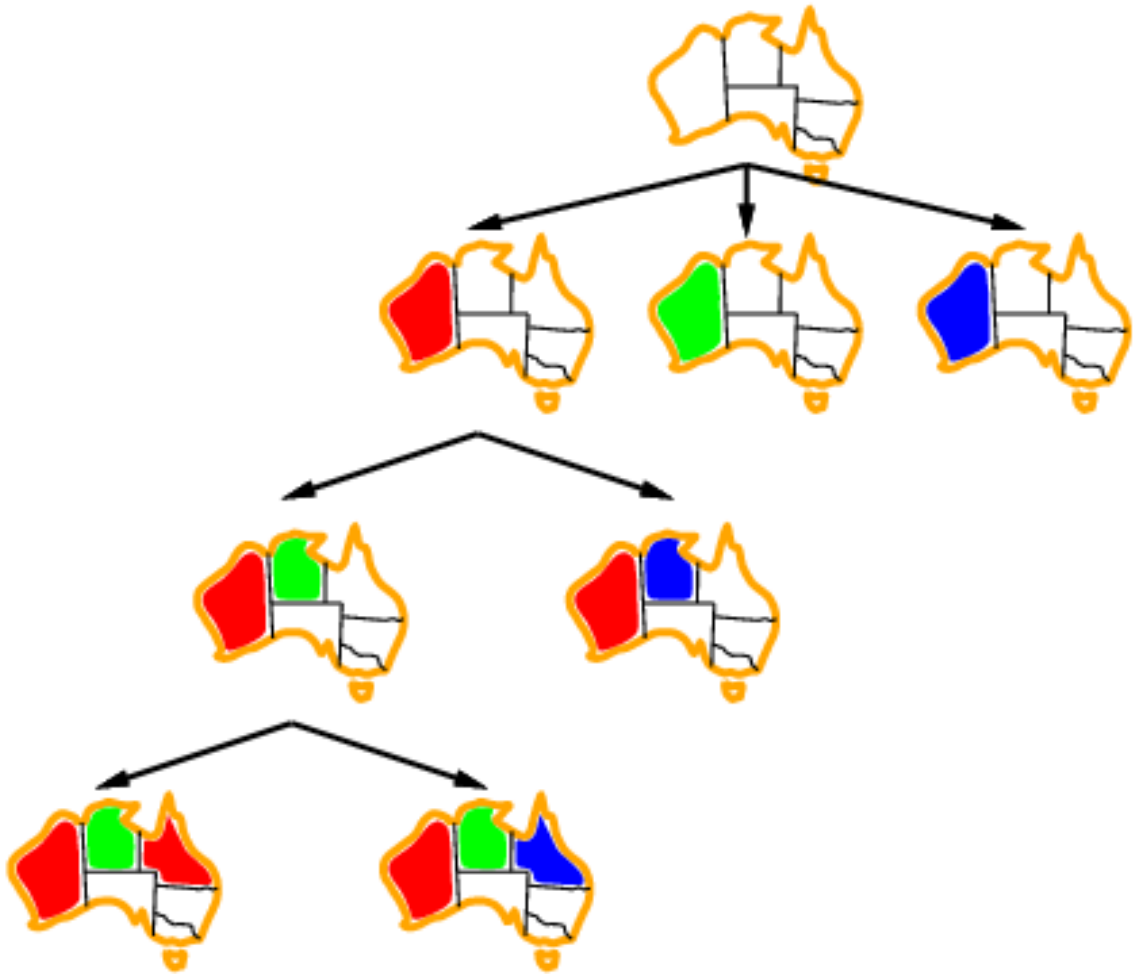
Pretraživanje sa vraćanjem

```
function RECURSIVE-BACKTRACKING(assignment, csp)
    if assignment is complete then return assignment
    var ← SELECT-UNASSIGNED-VARIABLE(VARIABLES[csp], assignment, csp)
    for each value in ORDER-DOMAIN-VALUES(var, assignment, csp)
        if value is consistent with assignment given CONSTRAINTS[csp]
            add {var = value} to assignment
            result ← RECURSIVE-BACKTRACKING(assignment, csp)
            if result ≠ failure then return result
            remove {var = value} from assignment
    return failure
```

Pretraživanje sa vraćanjem

- Rekurzivno pretraživanje po dubini
- Ponavljano bira promenljivu kojoj nije dodeljena vrednost i isprobava po redu sve vrednosti u domenu
- Ako se prepozna nekonzistentnost vraća se neuspeh i prelazi se na drugu vrednost
- Promenom funkcija Izaberi-Nedodeljenu-Promenljivu i Uredi-Domenske-Vrednosti može da se realizuje heuristika opšte namene

Pretraživanje sa vraćanjem



Pretraživanje sa vraćanjem - pitanja

- Koja promenljiva je sledeća za dodelu
- Kojim redosledom ispitati njene vrednosti
- Kada pretraživanje doživi neuspeh, može li pretraživanje da izbegne ponavljanje ovog neuspeha

Redosled promenljivih

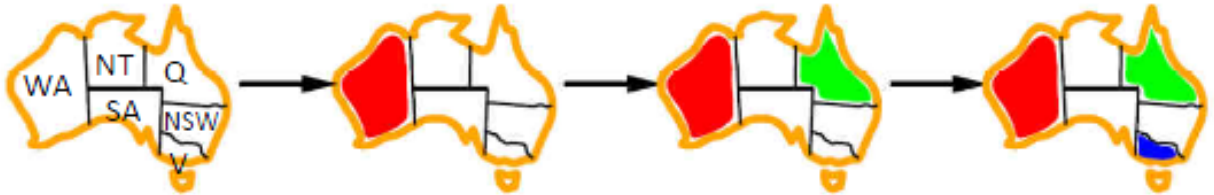
- Minimum preostalih vrednosti (MPV) – bira se promenljiva sa najmanje vrednosti (heuristika najograničenije promenljive)
  - Neuspeh prvo – ona koja će najverovatnije dovesti do neuspeha, odsecanja stable
  - Ako neka promenljiva nema preostalih dozvoljenih vrednosti izabraće se i odmah se otkriva neuspeh
  - Bolje performanse od nasumičnog
- Stepenasta heuristika (SH) – bira se promenljiva koja se pojavljuje u najvećem broju ograničenja za ostale nedodeljene promenljive
  - Kod mapa – SA ima stepen 5, ostale 2 ili 3, T ima 0 – kada se izabere SA, dalje bez pogrešnog koraka
- MPV je bolji putokaz ka rešenju, ali SH korisna za rešavanje nedoumica

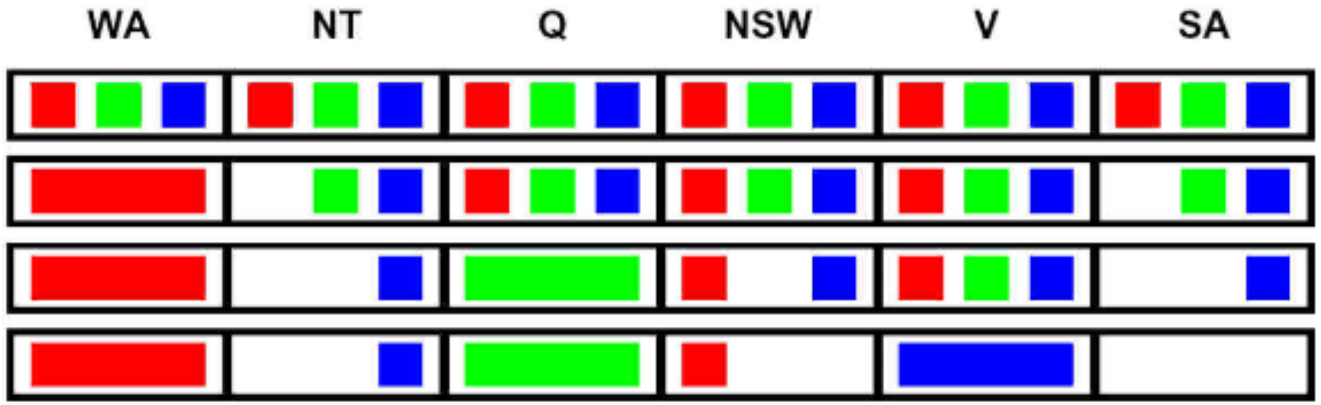
Redosled vrednosti

- Kada se izabere promenljiva – redosled ispitivanja vrednosti
- Vrednost najmanjeg ograničavanja
- Prednost onoj vrednosti koja najmanje sužava izbor susednih promenljivih u grafu ograničenja
- ZA=crveno, ST=zeleno, razmatramo za Kvinslend - plavo?
- Loš izbor, jer ukida poslednju dozvoljenu vrednost za JA
- Heuristika kaže crvena bolja od plave za KV – maksimalna fleksibilnost
- Za promenljive neuspeh-prvo, za promenljive neuspeh-poslednji?
  - Promenljive – minimiziranje broja čvorova u stablu, vrednosti – interesuje nas samo jedno rešenje, najverovatnije

Proveravanje unapred

- X se dodeli vrednost, ustanovi se konzistentnost grana za nju:
  - Za svako Y bez dodeljene vrednosti, koja je sa X povezana ograničenjem, briše se iz domena Y svaka vrednost koja je nekonzistentna sa vrednošću X
  - Nema potrebe da se sprovodi ako je obavljena konzistentnost



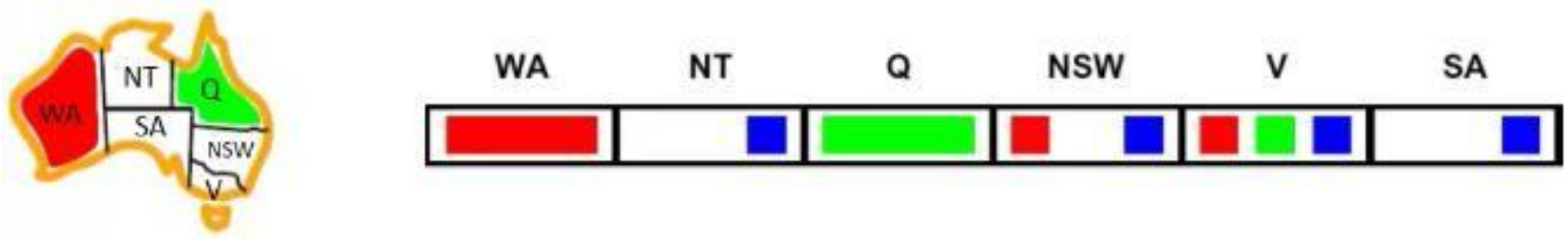


Proveravanje unapred

- Nakon dodele ZA=crveno i KV=zeleno, ST i JA svedeni samo na jednu vrednost – potpuno eliminisano grananje
- Nakon dodele V=plavo domen za JA ostaje prazan
- Zaključak da je delimična dodela nekonzistentna – vraćanje unazad
- Kombinacija MPV sa proveravanjem unapred
- Nakon ZA=crvena – ST i JA imaju po 2 vrednosti, one su sledeće
- Zatim KV, NJV, pa V, na kraju T sa 3 vrednosti
- Proveravanje unapred je efikasan način da inkrementalno proračunamo informaciju koja je potrebna MPV heuristici

Proveravanje unapred

- Nije otkrivena svaka nekonzistentnost
- Trenutno aktivna promenljiva se čini grana-konzistenta, ali ne gleda unapred za sve ostale
- ST i JA moraju da budu plavi
- Algoritam ne gleda dovoljno unapred da zaključi da je nekonzistentno – ST i JA su susedne i ne mogu biti iste boje



MAC

- Maintaining Arc Consistency
- Kada X_i dodeli vrednost poziva se AC-3
- Umesto reda čekanja svih grana počinje se samo sa granama (X_j, X_i)
- Za sve X_j koje su bez dodeljenih vrednosti i susedi sa X_i
- Ako se bilo gde dobije prazan domen AC-3 vraća neuspešno i vraćamo se
- Bolji rezultati, jer se vrši prostiranje ograničenja rekurzivno kada se dese promene u domenima promenljivih

Inteligentno vraćanje: gledanje unazad

- Do sada hronološko vraćanje (najskorija tačka odluke se prepravlja)
- Vrati se na prethodnu promenljivu i probaj drugu vrednost
- KV=crveno, NJV=zeleno, V=plavo, T=crveno – JA nema vrednost, vraćamo se na T – loše
- Vratiti se na neku promenljivu koja može da ispravi problem
- Pratimo skup dodela koje su u konfliktu sa nekom vrednošću
- KV=crveno, NJV=zeleno, V=plavo - konfliktni skup za JA, preskačemo T i biramo drugu vrednost za V
- Ako nije pronadjena dozvoljena vrednost, algoritam vraća prvi prethodni element iz konfliktnog skupa

Inteligentno vraćanje: gledanje unazad

- Kada god X=x izbriše vrednost iz domena Y dodaje se u konfliktni skup
- Ali svaka grana koju seče skakanje unazad takođe odseca proveravanje unapred
- Jednostavno skakanje unazad je redudantno sa svakim pretraživanjem koje koristi jače provere konzistentnosti (MAC)
- ZA=crveno, NJV=zeleno, T=crveno – zatim dodeljujemo ST, KV, V, JA
- ST ima vrednosti konzistentne sa prethodnim, ali ST, KV, V, JA zajedno dovode do neuspeha – zajedno sa sledećim promenljivama
- Skakanje unazad usmereno konfliktom

Skakanje unazad usmereno konfliktom

- Ako je X_j trenutna promenljiva, a konf(X_j) njen konfliktni skup
- Ako svaka moguća vrednost za X_j dovodi do neuspeha, skočiti nazad na prvu prethodnu promenljivu X_i iz skupa konf(X_j)

  konf(X_i)← konf(X_i) u konf(X_j) – {X_i}
- Kada dođe do kontradikcije, skakanje unazad će reći koliko da se vrati
- Obučavanje ograničenja – da se pronađe minimalan skup promenljivih iz konfliktnog skupa koji prouzrokuje problem
- Skup promenljivih sa vrednostima se naziva no-good
- Beležimo no-good ili dodavanjem novog ograničenja ili održavanjem zasebnog keša sa no-good elementima

Lokalno pretraživanje

- U početnom stanju se svakoj promenljivoj dodeljuje vrednost, a pretraživanje menja vrednost jednoj po jednoj promenljivoj
- Heuristika min-konflikata:
  - Prilikom biranja nove vrednosti za jednu promenljivu bira se vrednost koja će dovesti do najmanjeg broja konflikta sa ostalim promenljivama
- Min-konflikti su iznenađujuće efikasni za mnoge probleme
- Problem n-kraljica – vreme izvršavanja (bez početnog postavljanja kraljica) je nezavisno od veličine problema
- Zato što su rešenja gusto raspoređena u celom prostoru stanja
- Min-konflikti su korišćeni i kod teleskopa Hubble – izrada rasporeda za nedelju dana posmatranja sa tri nedelje smanjena na 10 minuta

Ponderisanje ograničenja

- Pomaže da se pretraživanje koncentriše na bitna ograničenja
- Svakom ograničenju se daje numerička težina W_i, u početku svima 1
- U svakom koraku pretraživanja bira se par promenljiva/vrednost koja se menja, a dati najmanju ukupnu težinu svih narušenih ograničenja
- Težine se prilagođavaju uvećavanjem težine svakog ograničenja koje je narušeno trenutnom dodelom
  - Dodaje se topografija platoima
  - Dodaje se težina na ograničenja koja su teška za rešavanje

Lokalna pretraživanja

- Prednost je i podešavanje u hodu kada se problem menja
- Značajno za probleme izrade rasporeda
  - Nedeljni raspored avio kompanije – hiljade letova i desteine hiljade zadataka
  - Vremenska nepogoda na jednom aerodromu – neizvodljivo
  - Algoritam lokalnog pretraživanja koji kreće od trenutnog rasporeda
  - Pretraživanje sa vraćanjem unazad uz novi skup ograničenja obično zahteva mnogo više vremena
  - Cilj je i da se ima što manje izmena