I007 Osnove umjetne inteligencije

Tema: Problemi zadovoljavanja ograničenja.

31. 3. 2021.

predavač: Darija Marković

asistent: Darija Marković

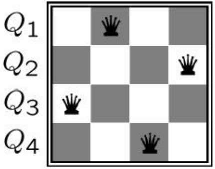
1. Problemi zadovoljavanja ograničenja (PZO)

- Problem zadovoljavanja ograničenja sastoji se od 3 komponente
  - X je skup varijabli $\{X_1, \ldots, X_n\}$
  - D je skup domena $\{D_1, \ldots, D_n\}$, gdje je svaka domena vezana za jednu varijablu
  - C je skup ograničenja koja određuju dozvoljene kombinacije vrijednosti
- Svako stanje u PZO definirano je dodjelom vrijednosti nekim ili svim varijablama
- Dodjela koja ne krši niti jedno ograničenje naziva se konzistentna ili legalna (zakonita) dodjela
- Potpuna dodjela je ona u kojoj svaka varijabla ima dodjeljenu vrijednost
- Rješenje za PZO je konzistentna, potpuna dodjela

Primjer bojenje zemljopisnih karata

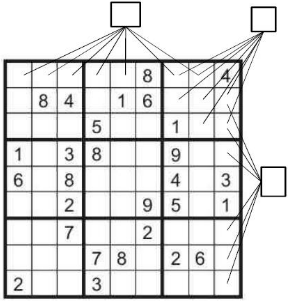
- varijable: $X = \{WA, NT, Q, NSW, V, SA, T\}$
- domena: $D_i = \{c, z, p\}$
- ograničenja: susjedne regije su različitih boja
  Implicitno: $WA \neq NT$
  Eksplicitno: $(WA, NT) \in \{(c, z), (c, p), \dots\}$

Primjer N - kraljica



- varijable: Q_k
- domena: $D_i = \{1, 2, \ldots, N\}$
- ograničenja:
  Implicitno: $\forall i, j \ (Q_i, Q_j)$ bez napada
  Eksplicitno: $(Q_1, Q_2) \in \{(1,3), (1,4), \ldots\}$

Primjer sudoku



- varijable: svaki prazni kvadrat
- domena: $D = \{1, 2, \dots, 9\}$
- ograničenja:
  U svakom stupcu svi različiti
  U svakom retku svi različiti
  U svakoj 3×3 regiji svi različiti

Vrste PZO

- Diskretne varijable
  - konačne domene (svi navedeni primjeri)
  - beskonačne domene (raspored poslova, varijable su početna i završna vremena za svaki posao)
- Kontinuirane varijable (problemi linearnog programiranja)

Vrste ograničenja

- unarna ograničenja: ograničavaju vrijednost samo jedne varijable
- binarna ograničenja: povezuju dvije varijable
- ograničenja višeg reda (globalna ograničenja): ograničenje koje obuhvaća 3 ili više varijabli

Preferense (prednosti/blaga (slaba) ograničenja)

- ograničenje koje nije apsolutno/jako, odnosno nije ga nužno zadovoljiti
- često predstavljaju cijenu postavljanja varijabli
- na ovaj način dolazimo do problema uvjetovane optimizacije

Zaključivanje u PZO

- algoritam može pretraživati ili raditi određenu vrstu zaključivanja zvanu prostiranje ograničenja, tj. koristeći ograničenja smanjiti domenu jedne varijable, što može izazvati smanjenje domene druge varijable itd.
- ključna ideja je lokalna konzistentnost. Ako svaku varijablu promatramo kao čvor u grafu, a svako binarno ograničenje kao granu (brid), onda nametanje lokalne konzistentnosti u svakom dijelu grafa dovodi do eliminiranja nekonzistenthih vrijednosti kroz graf
- tipovi lokalne konzistentnosti su
  - konzistentnost čvora
  - konzistentnost grane (brida)
  - konzistentnost putanje
  - K-konzistentnost

Standardna formulacija pretrage za PZO

- stanje je definirano sa svim do tada dodjeljenim vrijednostima
  - početno stanje: prazna dodjela
  - funkcija sljedbenik: dodjeljuje vrijednost nedodjeljenoj varijabli
  - test cilja: trenutna dodjela je potpuna i zadovoljava sva ograničenja
- naivan pristup je raditi pretrživanje u dubinu

Vraćanje unatrag (Backtracking)

- osnovni neinformirani algoritam za PZO

U njega su uklopljene sljedeće dvije ideje

- Jedna po jedna varijabla
  - pridruživanje vrijednosti varijablama je komutativno, tj. ne ovisi o redosljedu na koji dodjeljujemo vrijednosti varijablama
  - u svakom koraku ćemo dodjeljivati vrijednost samo jednoj varijabli
- Provjera ograničenja u svakom koraku
  - razmatraju se samo vrijednosti koje nisu u konfliktu s prije dodjeljenim
  - u svakom koraku ćemo potrošiti neko vrijeme za provjeru ograničenja
  - “postepeni test cilja”

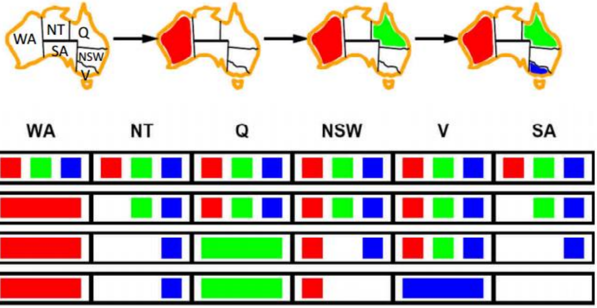
Poboljšanja za vraćanje unatrag

Za poboljšanje možemo iskoristiti sljedeće ideje

- Filtriranje: postoji li način za ranije otkrivanje neizbježnog neuspjeha?
- Redosljed:
  - Kojim redosljedom biramo varijable?
  - Kojim redosljedom biramo vrijednosti?
- Struktura: možemo li iskoristiti strukturu problema?

Filtriranje: Provjera unaprijed

- Pratimo domene nedodjeljenih varijabli i iz njih nakon svake dodjele izbacujemo loše izbore, tj. izbacujemo vrijednosti koje narušavaju ograničenja s obzirom na trenutnu dodjelu
- u trenutku kada se pojavi varijabla s praznom domenom vraćamo se unatrag



Filtriranje: Provjera unaprijed

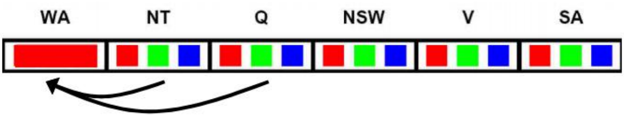
- Provjera unaprijed prosljeđuje informacije o varijablama s dodjeljenim vrijednostima prema varijablama s nedodjeljenim vrijednostima, ali ne omogućava rano otkrivanje svih neuspjeha

Iako znamo da NT i SA ne mogu obje biti plave, to ovdje nismo iskoristili!

Filtriranje: Konzistentnost grane

Grana $X \to Y$ je konzistentna ako i samo ako za svaki izbor x za X postoji neki y u domeni varijable Y koji ne narušava ograničenja.

- opet izbacujemo sve što stvara probleme (iz domene početka grane)
- provjera unaprijed osigurava konzistentnost onih grana koje završavaju u varijabli koju smo posljednju dodjelili



Filtriranje: Konzistentnost grafa

- tražimo da su sve grane konzistentne
- važno: ako X izgubi vrijednost, nanovo moramo provjeriti konzistentnost svih susjeda Y s X ($Y \to X$)
- provjera konzistentnosti grana otkriva neuspjehe prije od provjere unaprijed
- možemo provjeravati konzistentnost u predprocesiranju ili nakon svakog pridruživanja vrijednosti novoj varijabli
- produžuje se izvođenje svakog koraka
- nakon postizanja konzistentnosti grana moguće je
    - dobiti rješenje
    - imati nekoliko rješenja
    - nemati rješenje (i ne znati to)

Redosljed

Možemo gledati redosljed izbora za varijable ili za vrijednosti

- kod varijabli: minimum preostalih vrijednosti (Minimum Remaining Values (MRV))
  - izaberemo varijablu kojoj je preostalo najmanje vrijednosti u domeni
  - iako se to čini kao teža opcija, sve varijable na kraju moraju imati pridružene neke vrijednosti, pa je bolje teže izbore isprobati ranije
  - naziva se još i redosljed brzog neuspjeha
- kod vrijednosti: vrijednost najmanjeg ograničenja (Least Constraining Value (LCV))
  - izabiremo vrijednost koja izbacuje najmanje vrijednosti iz preostalih domena
  - to zahtjeva dodatni račun, ali omogućava dobre izbore
  - za razliku kod varijabli, sve vrijednosti ne moraju biti dodjeljene, pa možemo započeti s onima koje najvjerojatnije vode do rješenja

Struktura problema

S obzirom na izgled grafa, ponekad su dostupne tehnike za vrlo efikasno rješavanje

- ekstremni slučaj: nezavisni potproblem
  - nezavisne potprobleme možemo otkriti kao povezane komponente grafa ograničenja
  - uz pretpostavku da se graf s n varijabli može podijeliti na potprobleme sa samo c varijabli, pri čemu s d označimo veličinu domene, u najgorem slučaju imamo $\mathcal{O}(\binom{n}{c}\,d^c)$– linearno u n

PZO sa strukturom stabla

U grafu ograničenja nemamo ciklusa

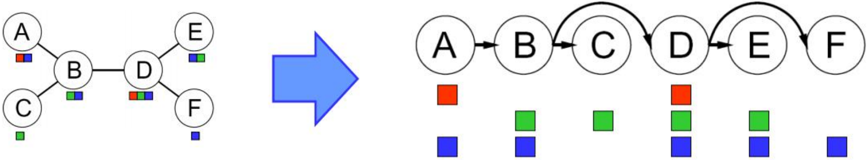
Teorem

Ukoliko graf ograničenja nema ciklusa, tada se PZO može rješiti u $\mathcal{O}(nd^2)$

Algoritam za PZO sa strukturom stabla

- odabere se proizvoljna varijabla kao korijen stabla, i nakon toga se ostale varijable poslažu na način da se svaka varijabla javlja nakon svog roditelja
- prvo se koristi strategija uklanjanja od natrag (remove backward): krenuvši od posljednjeg čvora uklanjamo sve nekonzistentne vrijednosti u granama $Roditelj \to Dijete$
- nakon toga se koristi strategija dodjeljivanja od naprijed (assign forward): krenuši od korijenskog čvora, redom se dodjeljuju vrijednosti

PZO sa strukturom stabla



Poboljšanje strukture

- ponekad uklanjanjem jedne ili više varijabli dolazimo do grafa ograničenja koje je stablo. U takvim slučajevima razmotrimo sljedeće:
    - napravimo sva moguća dodjeljivanja za te varijable, za preostale varijable smanjimo domenu, i nakon toga primjenimo algoritam za rješavanje PZO sa strukturom stabla
- moguće je također uvesti mega-varijable i na taj način stvoriti strukturu stabla, gdje svaka mega-varijabla sadrži dio originalnog problema
    - nakon što se riješe potproblemi, dodatno se zahtjeva da se dodjele varijablama koje su zajedničke u nekoliko mega-varijabli podudaraju

Algoritmi iterativnog poboljšavanja

- počinjemo s netočnim rješenjem i pokušavamo PZO riješiti uklanjajući konflikte
  - imamo potpunu dodjelu koja ne zadovoljava ograničenja
  - nekoj varijabli ponovno dodjeljujemo vrijednost
  - nemamo frontu!

Algoritam

Dok nije riješeno:

- odaberi varijablu: slučajno se odabire bilo koja varijabla koja je konfliktna
- odabir vrijednosti: heuristika minimalnog konflikta
  - odabire se vrijednost koja narušava najmanji broj ograničenja