Programování s omezujícími podmínkami

5

Roman Barták, KTIML

bartak@ktiml.mff.cuni.cz
http://ktiml.mff.cuni.cz/~bartak

Co bylo minule

Prohledávání a heuristiky
– Limited Discrepancy Search

Základní konzistenční techniky
odstraňování nekonzistentních hodnot použitím podmínek

Vrcholová konzistence (NC)
– unární podmínky

Hranová konzistence (AC)
– binární podmínky
– algoritmy AC-1, AC-2, AC-3, AC-4 (AC-5, AC-6, AC-7)

Mají smysl techniky slabší než AC?
– *směrová hranová konzistence (DAC)*

Stačí nám AC? Lze navrhnout něco silnějšího než AC?
– *konzistence po cestě (PC)*

Směrová hranová konzistence (DAC)

Pozorování 1: AC má směrový charakter, ale CSP není směrové.

Pozorování 2: AC musí opakovat revize hran, počet opakování závisí nejen na počtu hran ale i na velikosti domén (while cyklus).

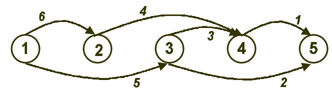
Můžeme AC nějak oslabit, abychom každou hranu revidovali pouze jedenkrát?

Definice: CSP je *směrově hranově konzistentní (directional arc consistent)* při daném uspořádání proměnných, právě když každá hrana (i,j), kde i<j je hranově konzistentní.

Opět kontrolujeme každou hranu, tentokrát v jednom směru

Algoritmus DAC-1

1) Konzistenci hrany požadujeme jen v jednom směru
2) Proměnné jsou uspořádané
   ↳ žádný orientovaný cyklus hran!



Pokud hrany projdeme v dobrém pořadí, nemusíme revize opakovat

Algoritmus DAC-1

```
procedure DAC-1(G)
   for j = |nodes(G)| to 1 by -1 do
      for each arc (i,j) in G such that i<j do
         REVISE((i,j))
      end for
   end for
end DAC-1
```

Použití DAC

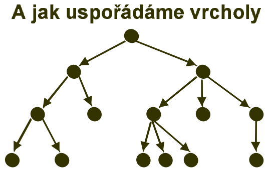
AC zřejmě zahrnuje DAC (je-li CSP AC, pak je i DAC)

Je DAC k něčemu dobré?
– DAC-1 je nepochybně efektivnější než AC-x
– existují problémy, kde DAC stačí

Příklad: Pokud graf CSP tvoří strom, potom stačí aplikovat DAC, abychom problém vyřešili bez navracení.

Jak uspořádáme vrcholy pro DAC?

A jak uspořádáme vrcholy pro ohodnocování?



1. Aplikujeme DAC v uspořádání vrcholů od kořene.

2. Ohodnocujeme vrcholy v pořadí od kořene.

DAC garantuje, že lze vždy najít hodnotu potomka kompatibilní s rodičem.

Vztah DAC k AC

Pozorování: CSP je hranově konzistentní, jestliže pro dané uspořádání proměnných je směrově hranově konzistentní v obou směrech.

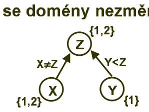
Můžeme AC dosáhnout tak, že aplikujeme DAC v obou směrech?

Obecně NE, ale…

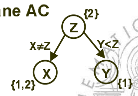
Příklad:

X in {1,2}, Y in {1}, Z in {1,2}, X≠Z, Y<Z

při uspořádání X,Y,Z se domény nezmění



při uspořádání Z,Y,X se změní pouze Z, ale nedostane AC



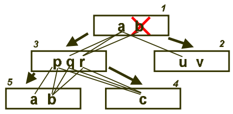
Pokud ale jako první zvolíme uspořádání Z,Y,X, dostaneme AC.

Od DAC k AC pro stromové CSP

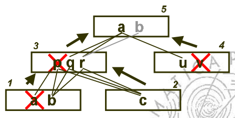
Pokud je DAC aplikováno na stromové CSP nejprve pro uspořádání vrcholů od kořene a po té v obráceném pořadí, potom získáme (plnou) hranovou konzistenci.

Důkaz:

po prvním běhu DAC **zajistíme, že pro každou hodnotu rodičovského vrcholu najdeme podporu (konzistentní hodnotu) u všech potomků**



pokud je *při druhém běhu DAC* (v opačném pořadí) vyřazena nějaká hodnota, potom tato hodnota nebyla podporou žádné hodnoty rodičovského uzlu (tj. hodnoty v rodičovském uzlu neztratily své podpory)



***dohromady*: každá hodnota má podporu v potomkovi (první běh) i v rodiči (druhý běh), tj. jedná se o AC**

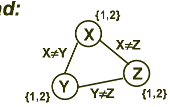
Je hranová konzistence dostatečná?

Použitím AC odstraníme mnoho nekompatibilních hodnot

- **dostaneme potom řešení problému?**
- **víme alespoň zda řešení existuje?**

NE a NE!

Příklad:



CSP je hranově konzistentní a přesto nemá žádné řešení

K čemu tedy AC je?

někdy dá řešení přímo

- **nějaká doména se vyprázdní → řešení neexistuje**
- **všechny domény jsou jednoprvkové → máme řešení**

v obecném případě alespoň zmenší prohledávaný prostor

Konzistence po cestě (PC)

Jak posílit konzistenci?

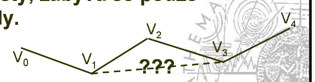
Budeme se zabývat několika podmínkami najednou!

Definice:

- ***Cesta*** **$(V_0,V_1,\dots,V_m)$ je *konzistentní* (*path consistent*), právě když pro každou dvojici hodnot $x \in D_0$ a $y \in D_m$ splňující binární podmínky na V_0,V_m existuje ohodnocení proměnných $V_1,\dots,V_{m-1}$ takové, že všechny binární podmínky mezi sousedy V_i,V_{i+1} jsou splněny.**
- **CSP je *konzistentní po cestě*, právě když všechny cesty jsou konzistentní.**

Pozor!

Definice PC nezaručuje, že jsou splněny všechny podmínky nad vrcholy cesty, zabývá se pouze podmínkami mezi sousedy.



PC a cesty délky 2 (Montanari)

Zjišťovat konzistenci všech cest není moc praktické

naštěstí nám stačí prozkoumat pouze cesty délky 2!

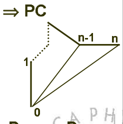
***Tvrzení:* CSP je PC, právě když každá cesta délky 2 je PC.**

Důkaz:

1) PC ⇒ cesty délky 2 jsou PC

2) cesty délky 2 jsou PC ⇒ ∀N cesty délky N jsou PC ⇒ PC

indukcí podle délky cesty



a) N=2 triviálně platí

b) N+1 (za předpokladu, že N platí)

i) vezmeme libovolných N+1 vrcholů $V_0,V_1,\dots,V_n$

ii) vezmeme libovolné dvě kompatibilní hodnoty $x_0 \in D_0$ a $x_n \in D_n$

iii) podle a) najdeme hodnotu $x_{n-1} \in D_{n-1}$ tž. $P_{0,n-1}$ a $P_{n-1,n}$ platí

iv) podle indukčního kroku najdeme zbylé hodnoty na cestě $V_0,V_1,\dots,V_{n-1}$

Vztah PC a AC

Je AC pokryto pomocí PC (je-li CSP PC, potom je i AC)?

- **hrana (i, j) je konzistentní (AC), pokud je cesta (i,j,i) konzistentní (PC)**
- **PC tedy implikuje AC**

Je PC silnější než AC (existuje CSP, které je AC a není PC)?

***Příklad*: X in {1,2}, Y in {1,2}, Z in {1,2}, X≠Z, X≠Y, Y≠Z**

je AC, ale není PC (X=1, Z=2 nelze rozšířit po cestě X,Y,Z)

AC vyřazuje nekompatibilní prvky z domén proměnných, co bude dělat PC?

- **PC vyřazuje dvojice hodnot**
- **PC dělá všechny relace implicitní (A<B,B<C ⇒ A+1<C)**
- **unární podmínka = doména proměnné**

Reprezentace podmínek maticí

V PC potřebujeme vyřazovat jednotlivé dvojice hodnot

↳ je potřeba pamatovat si podmínky explicitně

Binární podmínka = 0,1-matice

0 - hodnoty nejsou kompatibilní

1- hodnoty jsou kompatibilní

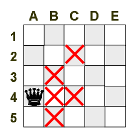
Příklad:

problém 5-ti dam

podmínka mezi dámami *i* a *j* má tvar: $r(i) \neq r(j)$ & $|i-j| \neq |r(i)-r(j)|$

maticový záznam pro dámy A(1) a B(2)

```
0 0 1 1 1
0 0 0 1 1
1 0 0 0 1
1 1 0 0 0
1 1 1 0 0
```



maticový záznam pro dámy A(1) a C(3)

```
0 1 0 1 1
1 0 1 0 1
0 1 0 1 0
1 0 1 0 1
1 1 0 1 0
```

Operace s podmínkami

Průnik podmínek R_{ij} & R'_{ij}

AND po odpovídajících polích

A<B	&	*A≥B-1*	→	*B-1≤A<B*
011		110		010
001	&	111	=	001
000		111		000

Složení podmínek $R_{ik} * R_{kj} \rightarrow R_{ik}$

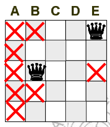
binární násobení matic

A<B	*	*B<C*	→	*A<C-1*
$\boxed{011}$		01$\boxed{1}$		001
001	*	00$\boxed{1}$	=	000
000		00$\boxed{0}$		000

Indukovaná podmínka se spojuje s původní podmínkou

R_{ij} & $(R_{ik} * R_{kj}) \rightarrow R_{ij}$

R_{25}	&	(R_{21}	*	R_{15})	→	R_{25}
01101		00111		01110		01101
10110		00011		10111		10110
$\boxed{11011}$	&	10001	*	11011	=	$\boxed{01010}$
01101		11000		11101		01101
10110		11100		01110		10110



Poznámky:

$R_{ij} = R^T_{ji}$, R_{ii} je diagonální matice reprezentující doménu

REVISE((i,j)) z AC algoritmů dělá $R_{ii} \leftarrow R_{ii}$ & $(R_{ij} * R_{jj} * R_{ji})$