

CSP - Constraint satisfaction programming

CSP problém je sadou krajicí (X, D, C) , kde:

$X = \{x_1, \dots, x_n\}$ je konečná množina proměnných, konára

$D = \{D_1, \dots, D_n\}$ je množina domén. Doména D_i je množina hodnot, kterých může nabývat proměnná x_i .

$C = \{C_1, \dots, c\}$ je konečná množina omezení. Omezení se dvojice (S_j, R_j) , kde $S_j \subseteq X$ a R_j je relace nad množinou S_j .

CSP je neobsaditelný úloha. Řešení je půjčené hodnot z D_i proměnným x_i tak, aby všechny proměnné měly nějakou hodnotu a všechna omezení byla splňová.

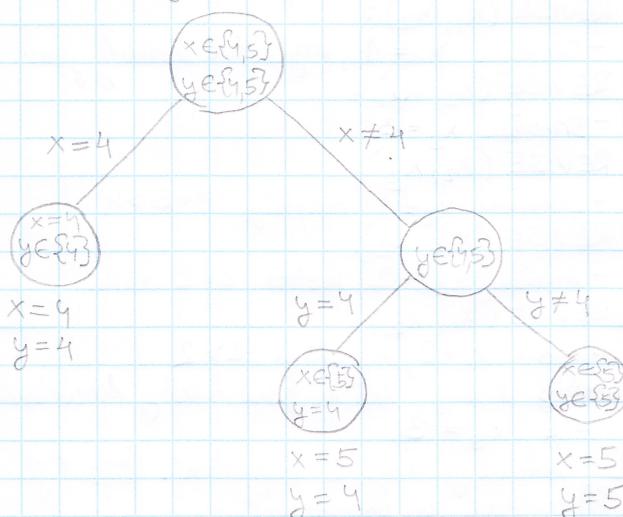
Existuje i optimizační verze CSP nazvaná CSOP, která je dana čtvrticí $(X, D, C, f(x))$, kde $f(x)$ je hledanou funkcií.

Sudoku jako CSP problém:

- každá řadka je proměnná, když máme 81 proměnných
- každá proměnná bere hodnoty z domény $D_i = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$
- pro každý řádek, sloupec i čtverec máme omezení, že každá proměnná musí mít mikadlní hodnotu.

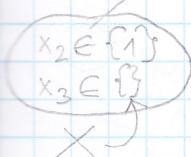
Př: $X = \{x, y\}, D = \{\{3, 4, 5\}, \{3, 4, 5\}\}, x \geq y, y > 3$

- provedeme omezení $y > 3$: $x \in \{3, 4, 5\}, y \in \{4, 5\}$
- — — $x \geq y$: $x \in \{4, 5\}, y \in \{4, 5\}$
- omezení máme upraveno, ale stále měníme výběr libovolnou dvojkou, třeba $x = 4, y = 5$ je replika!

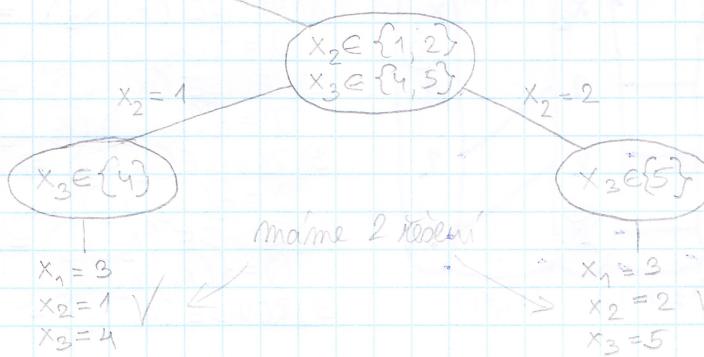


Algoritmus Search and propagation: $x_1 \in \{2, 3\}, x_2 \in \{1, 2, 3\}, x_3 \in \{4, 5\}$
 $x_1 > x_2 \wedge x_1 + x_2 = x_3$

Když konou vyřadíme, položíme $x_1 > x_2$



Když některá nic
nemá k x_3 , položí
me dohodit



Máme 2 řešení

Binánní CSP: v každém omezení jsou pouze 2 proměné (v němž CSP má totiž jde považovat). Problem reprezentujeme grafem, kde proměné jsou uzelohy a za každý omezení (x_i, x_j) uvedeme všechny brány mezi uzelohy x_i a x_j oběma směry.

Hrana (x_i, x_j) je konzistentní (AC), iif $\forall a \in D_i: \exists b \in D_j$ tak, že ochranné $x_i = a, x_j = b$ vyhovuje všem omezením pro x_i, x_j .

CSP je konsistentní, pokud jsou všechny brány konsistentní.

Procedura REVISE(D_i, D_j)

`deleted = False`

`for a in D_i :`

`b_found = False`

`for b in D_j :`

`if $x_i = a, x_j = b$ splňuje všechny constraints:`

`b_found = True`

`break`

`if not b_found:`

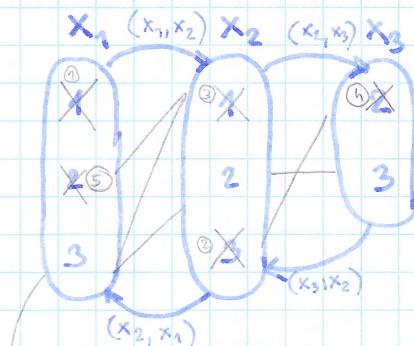
`D_i .remove(a)`

`deleted = True`

`return deleted`.

Smaře z domény všechny a, která nejsou klasifikovány, a mění hodnotu vzdálosti AC brány, a mění hodnotu vzdálosti, jestliže se něco změnilo.

Př.: $X = \{x_1, x_2, x_3\}, D = \{\{1, 2, 3\}, \{1, 2, 3\}, \{2, 3\}\}$
 $x_1 > x_2, x_2 \neq x_3, x_2 + x_3 > 4$



1) REVISE((x_1, x_2)):

- smařu $x_1 = 1$, když obě uzelohy jsou ok

2) REVISE((x_2, x_1)):

- smařu $x_2 = 3$

3) REVISE((x_2, x_3)):

- smařu $x_2 = 1$, neplatí $1 + 3 > 4$

4) REVISE((x_3, x_2)):

- mění $x_3 = 2$, protože $2 \neq 2$ nelze.

5) REVISE((x_1, x_2)):

- mění $x_1 = 2$, neplatí $2 > 2$.

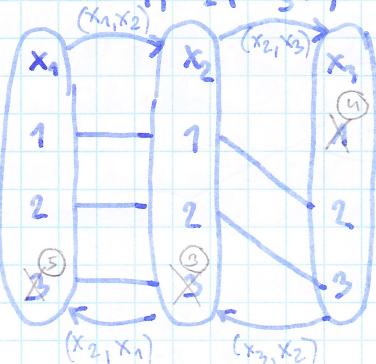
- REVISE se opakují novou na všechny brány, když už jsou všechny konsistentní

- přečtené řešení, $x_1 = 3, x_2 = 2, x_3 = 3$

→ brana mení hodnotu je když pokud je dogice považována. Řešení je spojeno bránami 3-2-3. Brány jsou jen pomocné?

Algoritmus AC3: Udržuje si frontu bran (x_i, x_j) k proklesání. Na každém kroku je řešení ve frontě všechny brány. Po spracování se odstraňuje a do fronty se přidají brány, které mohly být spracovány v dalších krocích.

Př.: $X = \{x_1, x_2, x_3\}, D = \{\{1, 2, 3\}, \{1, 2, 3\}, \{1, 2, 3\}\}, x_1 = x_2, x_2 + 1 = x_3$.



$Q = \{(x_1, x_2), (x_2, x_3), (x_3, x_1)\}$

1) REVISE((x_1, x_2)) \Rightarrow ne ok

$Q = \{(x_2, x_1), (x_2, x_3), (x_3, x_1)\}$

2) REVISE((x_2, x_1)) \Rightarrow ne ok

$Q = \{(x_2, x_3), (x_3, x_1)\}$

3) REVISE((x_2, x_3)) \Rightarrow mění $x_2 = 3$, vložit do (x_1, x_2)

$Q = \{(x_3, x_2), (x_1, x_2)\}$

4) REVISE((x_3, x_2)) \Rightarrow mění $x_3 = 1$

$Q = \{(x_1, x_2)\}$

5) REVISE((x_1, x_2)) \Rightarrow mění $x_1 = 3, Q = \emptyset$, konc