# Automatizované řešení úloh s omezeními

Martin Kot

Katedra informatiky, FEI,
Vysoká škola báňská – Technická universita Ostrava
17. listopadu 15, Ostrava-Poruba 708 33
Česká republika

1. listopadu 2012

# Zpětné hledání

# Konfliktní množiny

- Backjumping je jeden ze základních způsobů omezení tendence backtrackingu opakovaně prohlednávat stejné dead-end situace
- **Vinná proměná** (culprit variable) - její instanciace je odpovědná za dead-end (žádná možná kombinace následujících proměnných nevede k řešení)
- Když dokážeme identifikovat vinnou proměnnou, je backjump k nové instanciaci této proměnné lepší než postupné procházení prostoru backtrackingem
- Identifikace vinných proměnných je založena na konfliktních množinách
- Definice: Nechť $\bar{a} = (a_{i_1}, \ldots, a_{i_k})$ je konzistentní instanciace libovolné podmnožiny proměnných a $x$ je dosud neinstanciovaná proměnná. Pokud v doméně $x$ neexistuje hodnota $b$ taková, aby $(\bar{a}, x = b)$ bylo konzistentní, nazveme $\bar{a}$ **konfliktní množinou** (conflict set) proměnné $x$. Pokud navíc $\bar{a}$ neobsahuje žádnou podmnožinu konfliktní s $x$, nazveme $\bar{a}$ **minimální konfliktní množinou** proměnné $x$.

# Konfliktní množiny

- Definice: Nechť $\bar{a}_i = (a_1, \ldots, a_i)$ je konzistentní $i$tice. Když je $\bar{a}$ v konfliktu s $x_{i+1}$ nazveme situaci listový (leaf) dead end a proměnná $x_{i+1}$ je listová dead-end proměnná
- Definice: Pro síť $\mathcal{R} = (X, D, C)$ každou částečnou instanciaci $\bar{a}$, která není součástí žádného řešení, nazýváme **nedobrou** (no-good).
- Každá konfliktní množina je nedobrá, ale nedobrá množina nemusí být konfliktní pro jednu proměnnou
- Definice: Nechť $\bar{a}_i = (a_1, \ldots, a_i)$ je listová dead-end situace. Řekneme, že $x_j$, jde $j \leq i$ is **bezpečná** (safe) vzhledem k $\bar{a}$, když částečná instanciace $\bar{a}_j = (a_1, \ldots, a_j)$ je nedobrá.

# Backjumping styly

- **Gaschnig's backjumping** - skáče zpět na vinnou proměnnou jen při listové dead-end situaci.
- **Graph-based backjumping** - vytáhne z grafu omezení informaci o zbytečných krocích zpět a skáče i ve vnitřních (ne listových) dead-end situacích
- **Conflict-directed backjumping** - kombinuje skoky v listových i vnitřních dead end situacích a neomezuje se jen na informace z grafu

# Gaschnig's backjumping

- Definice: Nechť $\bar{a}_i = (a_1, \ldots, a_i)$ je listový dead-end. Index viny (culprit index) vzhledem k $\bar{a}_i$ je definován jako $b = min\{j \leq i \mid \bar{a}_j$ je v konfliktu s $x_{i+1}\}$. Vinná proměnná instanciace $\bar{a}_i$ je $x_b$.
- Tvrzení: Když je $\bar{a}_i$ listový dead-end a $x_b$ je vinná proměnná, tak $\bar{a}_b$ je bezpečná destinace pro backjump a $\bar{a}_j, j < b$ není.
- Určení vinné proměnné pro $\bar{a}_i$ je relativně snadné - je potřeba otestovat maximálně $i$ podmnožin na konzistentnost s $x_{i+1}$
- Navíc může být určena v průběhu hledání udržováním nějakých informací při přiřazování $\bar{a}_i$

# Graph-Based backjumping

- Když algoritmus skáče zpět na proměnnou $x_j$ z listového dead-endu a $x_j$ nemá další hodnoty k vyzkoušení, označujeme ji **vnitřní dead-end proměnnou** a $\bar{a}_{j-1}$ **vnitřní dead-end stav**
- Gaschnigův algoritmus provádí větší skoky jen z listových dead-endů, když skočí na vnitřní, tak se zachová jako standardní backtracking
- Algoritmus **graph-Based backjumping** umožňuje větší skoky i ve vnitřních dead-endech
- Informaci o možných konfliktních množinách získává pouze z grafu omezení (nezkoumá tedy typ omezení nebo třeba domény)
- Při dead-endu skáče na naposledy přiřazenou proměnnou, která je s aktuální proměnnou spojena hranou v grafu omezení
- Pokud je po skoku znovu dead-end, opět skáče na naposledy přiřazenou proměnnou, která je v grafu spojena s některou z proměnných, při kterých na dead-end narazil

# Conflict-Directed Backjumping

- Spojuje dva předchozí postupy
- Pracuje podobně jako graph-based backjumping, ale nedívá se jen na graf omezení
- Pro každou proměnnou si udržuje **jumpback set**
- Definice: Při daném uspořádání proměnných nazveme omezení $R$ dřívější než omezení $Q$, když nejpozdější proměnná z $scope(R) - scope(Q)$ předchází nepozdější proměnnou z $scope(Q) - scope(R)$

# Conflict-Directed Backjumping

- Relace dřívějšího omezení definuje úplné uspořádání na omezeních
- Definice: Nechť pro síť $\mathcal{R} = (X, D, C)$ s uspořádáním proměnných $d$ je $\bar{a}_i$ ntice jejíž potenciální dead-end proměnná je $x_{i+1}$. **Nejdřívější minimální konfliktní množina** pro $\bar{a}_i$ (nebo pro $x_{i+1}$), označovaná $emc(\bar{a}_i)$ může být vytvořena takto: uvažujme omezení uspořádáné podle relace dřívějšího, pro $j \in \{1, \ldots, c\}$ (počet omezení), pokud existuje $b \in D_{i+1}$ tž. $R_j$ je porušeno přiřazením $\bar{a}_i, x_{i+1} = b$ a žádné dřívější omezení tímto porušeno není, potom do $var - emc(\bar{a}_i)$ přidáme $S_j$ (rozsah $R_j$), $emc(\bar{a}_i)$ potom získáme projekcí $emc(\bar{a}_i) = a_i[var - emc(\bar{a}_i)]$
- Definice: **Jumpback set** $J_{i+1}$ listového dead-endu $x_i$ je jeho $var - emc(\bar{a}_i)$. Jumpback set vnitřního stavu $\bar{a}_i$ (nebo proměnné $x_{i+1}$) obsahuje všechny $var - emc(\bar{a}_j)$ všech relevantních dead-endů $\bar{a}_j, j \geq i$, které nastaly v současné řešení proměnné $x_i$

# Conflict-Directed Backjumping

- Tvrzení: Pro danou ntici $\bar{a}_i$ je nejpozdější proměnná v jumpback množině $J_i$ nejdřívější proměnnou, kam je bezpečné skočit.
- Algoritmus - viz slidy R. Dechter, chapter 6, slide 21

# Učící se algoritmy

- Při construkci minimální konfliktní množiny se ujasní nedobré instanciace a použijí se k rozhodnutí skoku zpět
- Stejně tak se takové instanciace dají přidat formou nových omezení, takže je algoritmus nebude v budoucnu znovu procházet díky testování konzistentnosti
- Tím se může "prosekat" strom prohledávaného prostoru
- Techniku nazýváme **constraint recording** ne **learning**
- Příležitost k naučení nového omezení je při každém dead-endu
- Pokud $\bar{a}_i$ je konfliktní množinou pro $x_{i+1}$, nedává moc smysl si přimo mezi omezení zakázat $\bar{a}_i$
- Pokud $\bar{a}_i$ obsahuje podmnožiny konfliktní s $x_{i+1}$, může se vyplatit zaznamenat tyto podmnožiny.
- Výhodné je identifikovat co nejmenší konfliktní množiny a ty si zapamatovat
- Jedním z kandidátů je earliest minimal conflict set (emc) z konflikty řízeného backjumpingu

# Graph-Based Learning

- **Graph-based learning** využívá stejné metody jako graph-based backjumping k určení nedobrých instanciací
- Informace o konfliktech vyvozuje pouze z grafu omezení
- Pro listový dead-end $\bar{a}_i$ jsou hodnoty přiřazené předkům $x_{i+1}$ zaznamenány v uložené konfliktní množině.
- Příklad viz slidy R. Dechter, chapter 6, slide 30

# Deep learning

- **Deep learning** znamená, že se zaznamenávají jen minimální konfliktní množiny
- **Shallow learning** - netrvá na tom, aby zaznamenané množiny byly minimální
- Deep learning využívá nejvíce informace k prosekání prohledávaného stromu, ale výpočet minimálních konfliktních množin je náročný (až exponenciální)

# Další varianty učících se algoritmů

- Jumpback Learning - místo všech minimálních konfliktních množin si pamatuje jen jednu, většinou jumpback množinu z conflict-directed backjumping
- Bounded and Relevance-Bounded Learning - každý učící algoritmus může být doplněn o omezení na velikost konfliktních množin
- Nonsystematic Randomized Backtrack Learning - při pravděpodobnostních algoritmech (náhodný výběr hodnot) se pamatují všechny nedobré instanciace, aby po restartu nebyly znovu procházeny