Řešiče SAT založené na DPLL

Radek Miček

Obsah prezentace

- algoritmus DPLL
- učení vynucujících klauzulí
- rozhodovací strategie VSIDS
- restarty

Vybrané definice (1)

- rozhodovací problém SAT
  - vstup: formule F v CNF
  - výstup: SAT je-li formule splnitelná, jinak UNSAT
- ohodnocení v přiřazuje hodnotu 0/1 některým proměnným z F
  - ohodnocení přirozeně rozšíříme na literály, klauzule a formule
- úplné ohodnocení je ohodnocení, které přiřazuje hodnotu všem proměnným v F

Vybrané definice (2)

- všechny literály v konfliktní klauzuli jsou ohodnoceny 0
- vyřešená/splněná klauzule obsahuje alespoň jeden literál ohodnocený 1
- jednotková klauzule obsahuje právě jeden neohodnocený literál, ostatní literály jsou ohodnoceny 0
- pozn.: klauzule chápeme jako množiny literálů

Algoritmus DPLL

DPLL(F, v)

- proveď propagaci a uprav ohodnocení v
  - napřed jednotkové klauzule, pak čisté literály
- je-li F splněná ohodnocením v, vrať SAT
- obsahuje-li F konfliktní klauzuli, vrať UNSAT
- vyber neohodnocenou proměnnou x
- vrať SAT, jestliže DPLL(F, v ∪ {x = 0}) vrací SAT
- vrať DPLL(F, v ∪ {x = 1})

proměnné ohodnocené propagací se nazývají implikované proměnné

tyto proměnné se nazývají rozhodovací proměnné a jejich počet udává rozhodovací úroveň (RÚ)

Další definice pro DPLL

- předchůdce implikované proměnné je klauzule, která způsobila jednotkovou propagaci dané proměnné
  - předchůdce proměnné x značíme A(x)
  - rozhodovací proměnné nemají předchůdce
- stopa je zásobník ohodnocených proměnných (naposledy ohodnocená proměnná je na vrcholu)

Problémy DPLL

- žádné poučení z konfliktů
- chronologický backtracking

} opravíme pomocí učení vynucujících klauzulí

- opakování stejné práce v různých větvích

Úvod do učení klauzulí

- idea: na základě informace odvozené z konfliktu přidáme novou klauzuli (i více), která zabrání podobným konfliktům
- často se přidává vynucující klauzule obsahující FUIP
- moderní SAT řešiče se snaží přidávat malé klauzule

Vynucující klauzule

- klauzule C je vynucující (asserting), pokud
  - $F \models C$,
  - $v(C) = 0$,
  - a obsahuje právě jednu proměnnou z aktuální RÚ
- fakt: snížením RÚ se z vynucující klauzule stane jednotková
  - skáčeme do nejnižší RÚ, kde je C ještě jednotková
  - po skoku pokračujeme jednotkovou propagací

Implikační graf

- implikační graf je orientovaný graf:
  - vrcholy odpovídají ohodnoceným proměnným
  - do implikované proměnné x vedou hrany z proměnných v množině $A(x) \setminus \{x, \neg x\}$ a jsou ohodnoceny klauzulí $A(x)$
  - je-li C konfliktní klauzule, pak do grafu přidáme vrchol K, do nějž povedou hrany vedoucí z proměnných v C, které ohodnotíme klauzulí C
- fakt: do rozhodovací proměnné nevede žádná hrana
- fakt: graf je acyklický (plyne z konstrukce)

Unikátní implikační bod (UIP)

- unikátní implikační bod (UIP) je proměnná přes kterou prochází všechny cesty z rozhodovací proměnné na aktuální RÚ do vrcholu K
- fakt: UIP je proměnná z aktuální RÚ
  - hrany vedou pouze do proměnných se stejnou nebo vyšší rozhodovací úrovní a my jsme na nejvyšší rozhodovací úrovni
- fakt: v dané vzdálenosti od K je nejvýše jeden UIP
  - sporem: vezmeme nejkratší cestu do K
- první UIP (FUIP) je UIP, který je nejblíž k vrcholu K
- fakt: FUIP je právě jeden

Řezy v implikačním grafu

- řezem v implikačním grafu budeme rozumět rozklad vrcholů na dvě disjunktní množiny, kde:
  - strana důvodu je množina obsahující všechny proměnné z nižších RÚ, rozhodovací proměnnou z aktuální RÚ a proměnné, z nichž nevede cesta do K
  - strana konfliktu je množina obsahující vrchol K
  - hrany vedou pouze ze strany důvodu na stranu konfliktu
- klauzule určená řezem je konfliktní klauzule obsahující právě ty proměnné, z nichž vedou rozříznuté hrany

Řezy a vynucující klauzule s FUIP

- jestliže FUIP je na straně důvodu a jeho následníci jsou na straně konfliktu, pak řez určuje vynucující klauzuli s FUIP
- důkaz:
  - klauzule obsahuje FUIP
  - klauzule je vynucující: pokud v klauzuli byla jiná proměnná než FUIP z aktuální RÚ, pak by přes ni vedla cesta do K ⇨ spor s definicí FUIP

Nalezení vynucující klauzule s FUIP

- máme konfliktní klauzuli C
  - je-li RÚ ≥ 1: konfliktní klauzule má alespoň 2 proměnné přiřazené v této RÚ
- opakuj, dokud C není vynucující klauzule:
  - buď x implikovaná proměnná z C, která je nejblíže vrcholu stopy
  - C := rezoluce(C, A(x), x)
- C je vynucující klauzule
- konečnost: v každém kroku se proměnné v C vzdálí od vrcholu stopy

Algoritmus Fast backjumping

- proveď propagaci, a pokud nastal konflikt, vrať UNSAT
- opakuj:
  - rozhodni neohodnocenou proměnnou
    - jsou-li všechny proměnné ohodnocené, vrať SAT
  - proveď propagaci
  - opakuj, dokud při propagaci nastává konflikt:
    - je-li RÚ = 0: vrať UNSAT
    - je-li RÚ > 0: spočítej a přidej vynucující klauzuli a skoč na příslušnou RÚ (nejnižší RÚ, že vynucující klauzule je jednotková)
    - proveď propagaci

Důkaz nesplnitelnosti rezolucí

- pokud je formule F splnitelná, může řešič založený na DPLL jednoduše vrátit splňující ohodnocení
- co v případě, kdy formule F není splnitelná?
- nastane-li konflikt v 0. RÚ, pak rezolucí dokážeme, že formule F není splnitelná
- C je konfliktní klauzule
- opakuj dokud se klauzule C nevyprázdní:
  - buď x proměnná z C, která je nejblíže vrcholu stopy
  - C := rezoluce(C, A(x), x)
- stejný princip jako při konstrukci vynucující klauzule

Problémy s učením klauzulí

- mnoho klauzulí zabírá mnoho paměti a zpomaluje hledání jednotkových a konfliktních klauzulí
  - chytrá implementace
    - př.: 2 sledované literály
  - promazávání klauzulí
    - př.: dlouhé klauzule, které se málo používají v jednotkové propagaci
  - minimalizace naučených klauzulí
    - př.: z konfliktní klauzule C lze odstranit literál l s implikovanou proměnnou x, pokud $A(x) \setminus \{\neg l\} \subseteq C$

Rozhodovací strategie VSIDS (1)

- Variable State Independent Decaying Sum
- pro každý literál udržuje váhu
- váhy literálů určují váhy proměnných:
  - váha proměnné je rovna maximu z vah příslušných literálů
- vybírá proměnnou s nejvyšší váhou
- proměnné přiřadí 1 právě, když je váha pozitivního literálu vyšší než váha negativního literálu

Rozhodovací strategie VSIDS (2)

- udržování vah:
  - váhy jsou inicializovány počty výskytů
  - při přidávání naučené klauzule zvýší skóre literálů, jenž jsou v klauzuli, o 1
  - po určitém pevně daném počtu konfliktů (např. 1000) se skóre všech literálů vydělí 2 (Decay)
- vlastnosti:
  - nezávislá na aktuálním ohodnocení (State Independent)
  - preferuje splňování naučených klauzulí
  - není náročná na výpočetní zdroje

Restarty

- motivace: únik ze špatných větví
- řešící proces se spustí od začátku, naučené klauzule si však ponechá
- existují různé restartovací strategie
  - např. restartuje se po určitém počtu konfliktů a prvních několik rozhodovacích proměnných se vybere náhodně
- pozor: restarty spolu s promazáváním naučených klauzulí mohou vést ke ztrátě úplnosti

Literatura

- J. P. Marques-Silva, K. A. Sakallah. GRASP – a new search algorithm for satisfiability. 1996
- L. Zhang, C. F. Madigan, M. W. Moskewicz, S. Malik. Efficient Conflict Driven Learning in a Boolean Satisfiability Solver. 2001
- N. Eén, N. Sörensson. An extensible SAT-solver. 2003
- N. Eén, N. Sörensson. MiniSat: A SAT solver with conflict-clause minimization. 2005
- M. W. Moskewicz, C. F. Madigan, Y. Zhao, L. Zhang, S. Malik. Chaff: Engineering an efficient SAT solver. 2001