Algorithmisches Beweisen LAB

CDCL - Entscheidungsheuristiken

Luc Spachmann

FSU Jena

06.06.2024

Ziele

- Implementierung von SAT-Lösern
  - 2-SAT
  - Hornformeln
  - DPLL
  - CDCL
    - watched literals
    - clause learning
    - decision heuristics
    - restart strategy

CDCL Pseudocode

Eingabe: CNF φ

1: decision-level $\leftarrow 0$
2: **while** Es existieren nicht belegte Variablen **do**
3: decision-level $\leftarrow$ decision-level $+ 1$
4: decide()
5: $C_{\text{conflict}} \leftarrow$ propagate()
6: **while** C_{conflict} is not null **do**
7: **if** decision-level $= 0$ **then return** UNSAT
8: C_{learned}, new_dec_lvl $\leftarrow$ analyze-conflict(C_{conflict})
9: $\varphi \leftarrow \varphi \wedge C_{\text{learned}}$
10: backtrack(new_dec_lvl)
11: $C_{\text{conflict}} \leftarrow$ propagate()
12: apply-restart-policy()
13: **return** SAT

Entscheidungen

- Gute Entscheidungen sind zentral für schnelles Lösen
- Nur bei idealer Literalauswahl ist CDCL p-äquivalent zu Res
- Für praktische Heuristiken: Exponentielle Trennung

VSIDS

- “Variable State Independent Decaying Sum“
- Hängt nicht von Belegung ab
- Priorisiert Variablen aus kürzlichen Konflikten
- Sehr verschiedene Implementierungen, aber gleiche Idee

VSIDS Implementierung

- Einfacher Algorithmus
  - Ein Zähler pro Variable (double/float)
  - Initialisierung aller Zähler auf 0
  - Bei Konflikt Erhöhung aller involvierten Variablen um Parameter b
  - Nach jedem Konflikt b auf $b \cdot c$ erhöhen
  - Alle k Konflikte zufällige Entscheidung
  - Um overflow zu verhindern, bei Bedarf neu skalieren
- Entschiedene Variable: Variable mit größtem Zähler
- Bei Gleichheit egal
- Variable involviert in Konflikt: In Konfliktanalyse berührt
- Beispielparameter (MiniSAT '03): $b = 2, c = 1.05, k = 200$
- Experimentieren mit unterschiedlichen Werten!

Phase Saving

- Bisher: Welche Variable wird gewählt
- Phase Saving: Wie wird die Variable gesetzt
- Speichern und Verwendung der letzten Belegung jeder Variable
- Egal ob von Propagation oder voriger Entscheidung
- Falls noch nicht belegt: Irgendwie (z.B. immer auf 0)

Neustarts

- Belegung wird vollständig gelöscht
- Gelernte Klauseln bleiben erhalten
- VSIDS und Phase Savings bleiben ebenfalls erhalten
- Wann: Nach bestimmter Anzahl Konflikten:
  - Anzahl Konflikte sollte ansteigen
  - Viele Funktionen denkbar, z.B. Geometrische Reihe
  - Hier: Luby Restarts

Luby

- Luby Folge: $1, 1, 2, 1, 1, 2, 4, 1, 1, 2, 4, 8, 1, \ldots$
- Formal:

$$t_i = \begin{cases} 2^{k-1} & \text{falls } i = 2^k - 1 \\ t_{i-2^{k-1}+1} & \text{falls } 2^{k-1} \leq i < 2^k - 1 \end{cases}$$

- i-ter Neustart nach $c \cdot t_i$ Konflikten für Konstante c
- Beispielswert $c = 100$

Aufgabe: CDCL - VSIDS und Restarts

- Implementierung von VSIDS und Restarts
- Vergleichen Sie die Performance
- Ausgabe einiger Statistiken:
  - Zeit
  - Speicherbedarf
  - Anzahl Unit Propagations
  - Anzahl Entscheidungen
  - Anzahl Konflikte
  - Anzahl Restarts
  - etc.

Evaluierung

- Bitte Evaluierung ausfüllen!
  `https://www.evaluation.uni-jena.de/lve/?e=34938&c=8262cf`