A Review of Stochastic Local Search Techniques for Theorem Provers

Termpaper presentation for

CSE814–Formal Methods in Software Development Course

Michigan State University, East Lansing, USA

Presenter: Khaled

December 11, 2014
What’s all about this *Theorem Prover* and *Local Search*?

- **Automated Theorem Prover (ATP):** Computer programs that can do axioms $\Rightarrow$ conjecture, i.e. find $\mathcal{M}$ such that $\mathcal{M} \models \phi_1, \phi_2, \ldots, \phi_k$.

- **Local Search (LS):** Metaheuristics for solving optimization problem. How to optimization $\rightarrow$ decision?

- **Stochastic Local Search (SLS):** Uses probability models/sampling for Local Search, if the search space is *HUGE* (NP-Hard). Closely related to *Approximation Algorithms*. 
Motivation

Why are you doing this review?

- **ATP ⇔ CAV:**
  - ATPs are *kernel* for many CAV softwares
  - sat4j → Alloy-Analyzer
  - Alt-Ergo → GNATprove

- **Popularity of LS/SLS:**
  - *Lots* of papers/research work, started around 60’s.
  - Easy to implement and tweak – flexible.
  - Good news – Random Walk Hypothesis.
  - Bad news – No Free Launch theorem.

- **Interesting:**
  - It’s interesting.
A Review Problem

Most Recent Earliest Common Ancestor Problem

- Big volume – makes it tough to trace back to the initial work.
- Too many variants.
What are you going to talk about?

Outlines

- Basic SLS Algorithm and problems.
- Basic SLS $\rightarrow$ WalkSAT.
- WalkSAT $\rightarrow$ \textit{Novelty} and \textit{Novelty}⁺.
- Discrete Lagrangian Methods (DLM).
- A Possible Enhancement to SLS.
Basic SLS Framework

\( \pi \) is a problem (i.e. SAT problem) instance from \( \Pi \)

Definitions & Terms

- **Search Space**: set of points \( s \in S_\pi \)
- **Solution Set**: \( S' \subseteq S_\pi \)
- **Neighbourhood**:
  - \( N \subseteq S_\pi \times S_\pi \) on \( S_\pi \).
  - \( s_j \in S_\pi \) is reachable from \( s_i \in S_\pi \) if \( \{s_i, s_j\} \in N \).
  - \( |N| \leq \rho \).
- **Initialization Function**: \( I : \emptyset \mapsto (S_\pi \mapsto \mathbb{R}) \), start with some random \( s \).
- **Step Function**: \( \sigma : S_\pi \mapsto (N \mapsto \mathbb{R}) \), jump from \( s_i \) to \( s_j \).
- **Objective Function**: \( f : S_\pi \mapsto \mathbb{R} \), how good is \( s \)?
Basic SLS Framework

Incompleteness & Stagnation

- Incompleteness: SLS algorithms are not guaranteed to find a solution even if a model $M$ exists.
- Stagnation: **Local optima**, non-determinism in search.
A Generic SLS Algorithm

Require: CNF formula $\phi$, Maximum number of flips $N_f$, Maximum number of trials $N_t$

Ensure: There exists a truth assignment $t \in M$ such that $M \models \phi$ or there is no such $t$.

1: repeat
2: truth assignment $t \leftarrow$ initialize()
3: repeat
4: if $t$ satisfies $\phi$ then
5: return $t$
6: else
7: $v \leftarrow \text{choose-variable}(\phi)$
8: $t \leftarrow \text{flip}(v, t)$
9: end if
10: until $N_f$ flips are done
11: until $N_t$ trials are done
12: return Fail

Algorithm 1: Generic SLS
WalkSAT algorithm

Require: CNF formula $\phi$, Maximum number of flips $N_f$, Maximum number of trials $N_t$
Ensure: There exists a truth assignment $t \in M$ such that $M \models \phi$ or there is no such $t$.
1: repeat
2: truth assignment $t \leftarrow$ initialize()
3: repeat
4: if $t$ satisfies $\phi$ then
5: return $t$
6: else
7: $v \leftarrow$ choose-variable-walksat($\phi$)
8: $t \leftarrow$ flip($v, t$)
9: end if
10: until $N_f$ flips are done
11: until $N_t$ trials are done
12: return Fail

Algorithm 2: WalkSAT
WalkSAT: choose-variable-walksat()

Tries to be *safe* – minimum damage flip.

**Require:** CNF formula $\phi$

**Ensure:** A variable $v$ to be flipped

1. $c \leftarrow$ choose-random-clause($\phi$) such that $c$ is already falsified.
2. $v_r \leftarrow \emptyset$
3. **for all** $v_i \in c$ **do**
4.   **if** ($v_i \leftarrow \neg v_i$) does not falsify a satisfied clause **then**
5.     $v_r \leftarrow v_i$
6.   **else if** ($v_i \leftarrow \neg v_i$) falsifies minimum number of clauses **then**
7.     $v_r \leftarrow v_i$
8.   **end if**
9. **end for**
10. **if** $v_r = \emptyset$ **then**
11.   $v_r \leftarrow$ choose-random-variable($\phi$) with probability $p$
12. **end if**
13. **return** $v_r$

**Algorithm 3:** choose-variable-walksat
WalkSAT: Evaluation

- Minimum damage flip.
- Flips what needs to be flipped.
- Greedy search – local optima stagnation.
- Incompleteness – may not terminate even if a model exists.
- Comes with a knob $p$ – problem dependant.
- Introduced other algorithms into the field.
**Novelty algorithm**

**Require:** CNF formula \( \phi \), Maximum number of flips \( N_f \), Maximum number of trials \( N_t \)

**Ensure:** There exists a truth assignment \( t \in M \) such that \( M \models \phi \) or there is no such \( t \).

1: repeat
2: truth assignment \( t \leftarrow \text{initialize()} \)
3: \( v_p \leftarrow \emptyset \)
4: repeat
5: if \( t \) satisfies \( \phi \) then
6: return \( t \)
7: else
8: \( v \leftarrow \text{choose-variable-novelty}(\phi, v_p) \)
9: \( t \leftarrow \text{flip}(v, t) \)
10: \( v_p \leftarrow v \)
11: end if
12: until \( N_f \) flips are done
13: until \( N_t \) trials are done
14: return Fail

**Algorithm 4:** Novelty
Novelty: choose-variable-novelty()

**Require:** CNF formula $\phi$, most recent flipped variable $v_p$

**Ensure:** A variable $v$ to be flipped next

1. $c \leftarrow$ choose-random-clause($\phi$) such that $c$ is already falsified.
2. **for all** $v_i \in c$ **do**
3. \[ s_i \leftarrow \text{total number of satisfied clauses if } v_i \text{ were flipped} \]
4. **end for**
5. $v_r \leftarrow$ find the $v_i \in c$ that maximize $s_i$
6. **if** $v_r \neq v_p$ **or** $v_p \neq \emptyset$ **then**
7. \[ \text{return } v_r \]
8. **else**
9. \[ r \leftarrow U(\mu, \sigma): U \text{ is a uniform distribution} \]
10. **if** $r \leq 1 - p$ **then**
11. \[ \text{return } v_r \]
12. **else**
13. \[ c' \leftarrow c - \{v_r\} \]
14. $v_r \leftarrow$ find the $v_i \in c'$ that maximizes $s_i$
15. \[ \text{return } v_r \]
16. **end if**
17. **end if**

**Algorithm 5:** choose-variable-novelty
Novelty/Novelty\(^{+}\): Evaluation

- Avoid repeat – does not flip the same bit back and forth.
- Incompleteness – could be solved by adding another knob \( p_w \).
- Avoids revisits but still greedy.
- Adaptive Novelty\(^{+}\) – change \( p \) and \( p_w \) by observing the convergence rate.
- Stagnation can be avoided but introduces more parameters.
Discrete Lagrangian Methods (DLM)

LP problem

\[
\begin{align*}
\text{min (or max)} & \quad N(t) = \sum_{i=1}^{k} C_i(t) \\
\text{s.t.} & \quad C_i(t) = \prod_{j=1}^{n} Q_{i,j}(v_j) \\
& \quad \phi = \{c_1, c_2, \ldots, c_k\} \\
& \quad t \in \{0, 1\}^n
\end{align*}
\]

Lagrange Multiplier

\[
\begin{align*}
L(t, \lambda) &= N(t) + \lambda^T C(t) \\
t^{k+1} &= t^k - \Delta_t L(t^k, \lambda^k) \\
\lambda^{k+1} &= \lambda^k + C(t^k)
\end{align*}
\]
A Basic DLM

Require: CNF formula $\phi$
Ensure: There exists a truth assignment $t \in \mathcal{M}$ such that $\mathcal{M} \models \phi$ or there is no such $t$.

1: $t \leftarrow$ a random $n$-dimensional $0 - 1$ vector
2: $\lambda \leftarrow [0, 0, \ldots, 0]$
3: construct $C(t)$ from $\phi$
4: repeat
5: \ $t \leftarrow t - \Delta_t L(t, \lambda)$
6: if update condition for $\lambda$ is satisfied then
7: \ $\lambda \leftarrow \lambda + \gamma C(t)$
8: end if
9: until $N(t) = 0$ (or $N(t) = k$ for max)
10: return $t$

Algorithm 6: DLM
DLM: Evaluation

- $\gamma$ is the convergence rate, smaller is better.
- Exploration-Exploitation trade-off can be controlled.
- Expose to lots of LP solving algorithms – simplex, interior point, ellipsoid ...
- Approximate LP solver $\rightarrow$ near global optima $\rightarrow$ DPLL.
Which one is better?

- No one knows.
- Depends on the problem.
- Different methods have different downsides and upsides.
List of some interesting papers –

H. H. Hoos and T. Stützle.
Local search algorithms for SAT: An empirical evaluation.

B. Selman, H. Kautz, and B. Cohen.
Local search strategies for satisfiability testing.

D. McAllester, B. Selman, and H. Kautz.
Evidence for invariants in local search.

Y. Shang and B. Wah.
A discrete lagrangian-based global-search method for solving satisfiability problems.

M. Davis, G. Logemann, and D. Loveland.
A machine program for theorem-proving.

A genetic programming approach to automated software repair.
Is there any other way to improve?

May be, what if two close neighbouring truth assignment \( t_i \) and \( t_j \) with large difference in objective function values??

- \( \Delta t = \| t_1 - t_2 \| \)
- \( f_\phi(t_1) \gg f_\phi(t_2) \)

Linkage Learning??
Feedback

Questions?

Suggestions?