Stochastic Local Search

Steffen Hölldobler
International Center for Computational Logic
Technische Universität Dresden
Germany

- Stochastic Local Search Algorithms
- Uninformed Random Walk
- Iterative Improvement
- Local Minima and Escape Strategies
- Randomized Iterative Improvement
- Tabu Search
- The GSAT Architecture
- The Walksat Architecture

"Logic is everywhere ..."
Stochastic Local Search Algorithms

A probability distribution for a finite set $S$ is a function $D : S \mapsto [0, 1]$ with

$$\sum_{s \in S} D(s) = 1$$

Let $\mathcal{D}(S)$ denotes the set of probability distributions over a given set $S$

Given a (combinatorial) problem $\Pi$, a stochastic local search algorithm for solving an arbitrary instance $\pi \in \Pi$ is defined by the following components:

- the search space $S(\pi)$, which is a finite set of candidate solutions $s \in S(\pi)$
- a set of solutions $S'(\pi) \subseteq S(\pi)$
- a neighbourhood relation on $S(\pi)$: $N(\pi) \subseteq S(\pi) \times S(\pi)$
- a finite set of memory states $M(\pi)$
- an initialization function $\text{init}(\pi) : \to \mathcal{D}(S(\pi) \times M(\pi))$
- a step function $\text{step}(\pi) : S(\pi) \times M(\pi) \to \mathcal{D}(S(\pi) \times M(\pi))$
- a termination predicate $\text{terminate}(\pi) : S(\pi) \times M(\pi) \to \{\bot, \top\}$
Some Notation

- We often write \( \text{step}(\pi, s, m) \) instead of \( \text{step}(\pi)(s, m) \) and, likewise, for terminate and other functions.
- We omit \( M(\pi) \) and the parameter \( m \) if no memory is used.
General Outline of a Stochastic Local Search Algorithm

procedure \texttt{SLSDecision}(\pi)
\begin{itemize}
  \item input \( \pi \in \Pi \)
  \item output \( s \in S'(\pi) \) or “no solution found”
\end{itemize}
\( (s, m) = \texttt{selectRandomly}(S(\pi) \times M(\pi), \text{init}(\pi)) \);
\begin{itemize}
  \item while not \texttt{terminate}(\pi, s, m) do
  \item \( (s, m) = \texttt{selectRandomly}(S(\pi) \times M(\pi), \text{step}(\pi, s, m)) \);
  \end{itemize}
\begin{itemize}
  \item if \( s \in S'(\pi) \) then
  \item return \( s \)
  \item else
  \item return “no solution found”
  \end{itemize}
\end{itemize}

where \texttt{selectRandomly} gets a pair \((S(\pi) \times M(\pi), D)\) as input
and yields the result of a random experiment selecting an element of
\( S(\pi) \times M(\pi) \) wrt the probability distribution \( D \in \mathcal{D}(S(\pi) \times M(\pi)) \)
A Simple SLS Algorithm for SAT: Uninformed Random Walk (URW)

- Let $F$ be a CNF-formula with variables $1, \ldots, n$.
- The search space $S(F)$ is the set of all interpretations for $F$.
- The set of solutions $S'(F)$ is the set of models for $F$.
- The neighbourhood relation on $S(F)$ is the one-flip neighbourhood $N(F, I, I')$ if there exists $A \in \{1, \ldots, n\}$ such that $A_I \neq A'_I$ and for all $A' \in \{1, \ldots, n\} \setminus \{A\}$ we find $A'_I = A''_I$.
- We will not use memory.
- The initialization function yields the uninformed random distribution
  \[
  \text{init}(F, I) = \frac{1}{|S(F)|} = \frac{1}{2^n} \text{ for all } I \in S(F).
  \]
- The step function maps any $I$ to the uniform distribution over all its neighbours
  \[
  \text{step}(F, I, I') = \frac{1}{|\{I' | N(F, I, I')\}|} = \frac{1}{n} \text{ for all } I' \text{ with } N(F, I, I').
  \]
- $\text{terminate}(F, I)$ holds iff $I \models F$. 
Evaluation Functions

Given a (combinatorial) problem $\Pi$ and let $\pi \in \Pi$; an evaluation function $g(\pi) : S(\pi) \mapsto \mathbb{R}$ is a function which maps each candidate solution to a real number such that the global optima of $g(\pi)$ correspond to the solutions of $\pi$.

Optima are usually minima or maxima.

$g(\pi)$ is used to rank candidate solutions.

Concerning SAT: Let $F$ be a CNF-formula and $I$ an interpretation.

- Often, $g(F)(I) = g(F, I)$ is the number of clauses of $F$ not satisfied by $I$, i.e.,
  \[ g(F, I) = |\{C \in F \mid I \not\models C\}| \]

- Consequently, $g(F, I) = 0$ iff $I \models F$.
Iterative Improvement

Given $\Pi, \pi \in \Pi, S(\pi), N(\pi)$ and $g(\pi)$

We assume that the solutions of $\pi$ correspond to global minima of $g(\pi)$

Iterative improvement (II) starts from a randomly selected point in the search space and tries to improve the current candidate solution wrt $g(\pi)$

Initialization function

$$\text{init}(\pi, s) = \frac{1}{|S(\pi)|} \quad \text{for all } s \in S(\pi)$$

Neighbouring candidate solutions

$$N'(s) = \{ s' \mid (s, s') \in N(\pi) \text{ and } g(\pi, s') < g(\pi, s) \} \quad \text{for all } s \in S(\pi)$$

Step function

$$\text{step}(\pi, s, s') = \begin{cases} \frac{1}{|N'(s)|} & \text{if } s' \in N'(s) \\ 0 & \text{otherwise} \end{cases} \quad \text{for all } s, s' \in S(\pi)$$
Local Minima and Escape Strategies

▶ The step function in the definition of iterative improvement is ill-defined!

▶ Given $\Pi, \pi \in \Pi, S(\pi), N(\pi)$ and $g(\pi)$

▶ A local minimum is a candidate solution $s \in S(\pi)$ such that for all $(s, s') \in N(\pi)$ we find $g(\pi, s) \leq g(\pi, s')$

▶ A local minimum $s \in S(\pi)$ is strict if for all $(s, s') \in N(\pi)$ we find $g(\pi, s) < g(\pi, s')$

▷ If II encounters a local minimum which does not correspond to a solution, then it “gets stuck”; step$(\pi, s)$ is not a probability distribution!

▶ Escape Strategies

▷ Restart re-initialize the search whenever a local minimum is encountered

▷ Random Walk perform a randomly chosen non-improving step

▷ Tabu List forbid steps to recently visited candidate solutions

▶ Even with these escape strategies there is no guarantee that an SLS-algorithm does eventually find a solution
Randomized Iterative Improvement – Preliminaries

- We want to escape local minima by selecting non-improving steps
- Walk Probability $wp \in [0, 1]$  
- $\text{stepURW}$ the step function of uninformed random walk  
- $\text{stepII}$  
  a variant of the step function used in the iterative improvement algorithm, which differs only in that a minimally worsening neighbour is selected if $N'(s) = \emptyset$
The Step Function of Randomized Iterative Improvement

procedure stepRII(π, s, wp)
    input π ∈ Π, s ∈ S(π), wp ∈ [0, 1]
    output s′ ∈ S(π)
    u = random([0, 1]);
    if u ≤ wp then
        s′ = stepURW(π, s);
    else
        s′ = stepII(π, s);
    end
    return s′
end
The Randomized Iterative Improvement Algorithm

► Termination

▷ after limit on the CPU time
▷ after limit on the number of search steps, i.e., iterations of the while loop or
▷ after a number of search steps have been performed without improvement

► Properties

▷ Arbitrarily long sequences of random walk steps may occur
▷ The algorithm can escape from any local minimum
▷ Solutions can be (provably) found with arbitrarily high probability
GUWSAT

- Randomized iterative improvement algorithm for SAT, but
  - instead of step II
  - a best improvement local search algorithm is applied, i.e.,
    - in each step a variable is flipped that leads to a maximal increase in the evaluation function
- The algorithm does not terminate in a local minima
  - The maximally improving variable flip is a least worsening step in this case
- The search in step URW is still uninformed
Tabu Search

- Iterative improvement algorithm using a form of short-term memory
- It uses a best improvement strategy
- Forbids steps to recently visited candidate solutions
  - by memorizing recently visited solutions explicitly or
  - by using a parameter \( tt \) called tabu tenure
The Step Function of Tabu Search

procedure stepTS(π, s, tt)
    input π ∈ Π, s ∈ S(π), tt
    output s' ∈ S(π)
    \[ N' = \text{admissableNeighbours}(\pi, s, tt); \]
    s' = selectBest(N');
    return s'
end
The GSAT Architecture

- GSAT was one of the first SLS algorithms for SAT
  Selman, Levesque, Mitchell:
  A New Method for Solving Hard Satisfiability Problems
  In: Proc. AAAI National Conference on Artificial Intelligence, 440-446: 1992

- Given CNF-formula $F$ and interpretation $I$, GSAT uses
  - the one-flip neighbourhood relation
  - the evaluation function

$$g(F, I) = |\{C \in F \mid I \not\models C\}|$$

- the score $g(F, I) - g(F, I')$ of a variable $A$ under $I$
  where $I'$ is obtained from $I$ by flipping $A$

- At the time of its introduction GSAT outperformed the best systematic search algorithms for SAT
The Basic GSAT Algorithm

procedure $\text{gsat}(F, \text{maxtries}, \text{maxsteps})$

input $F \in \mathcal{L}(\mathcal{R})$, $\text{maxtries}, \text{maxsteps} \in \mathbb{N}^+$

output model of $F$ or "no solution found"

for $\text{try} = 1$ to $\text{maxtries}$ do

$I = \text{randomly chosen interpretation of } F$;

for $\text{step} = 1$ to $\text{maxsteps}$ do

if $I \models F$ then

return $I$

end

$A = \text{randomly selected variable with maximal score}$;

$I = I$ with $A$ flipped;

end

end

return "no solution found"
end
GSAT with Random Walk (GWSAT)

- Consider GSAT, but use a randomised best-improvement search method

- **Conflict-directed random walk steps**  
  In a random walk step do
  - randomly select a currently unsatisfied clause $C$
  - randomly select a variable $A$ occurring in $C$
  - Flip $A$

- **GWSAT**
  - Use the basic GSAT algorithm
  - At each local step decide with fixed walk probability $wp$ whether to do
    - a standard GSAT step or
    - a conflict-directed random walk step

- In contrast to GUWSAT, GWSAT performs informed random walk steps

- GWSAT achieves substantially better performance than basic GSAT
GSAT with Tabu Search (GSAT/Tabu)

- Consider GSAT, but after $A$ has been flipped, it cannot be flipped back within the next $tt$ steps.
- With each variable $A$ a tabu status is associated as follows:
  - Let $t$ be the current search step number.
  - Let $tt \in \mathbb{N}$.
  - Let $t_A$ be the search step number, when $A$ was flipped for the last time.
  - Initialize $t_A = -tt$.
  - Every time variable $A$ is flipped set $t_A = t$.
  - Variable $A$ is \textit{tabu} iff $t - t_A \leq tt$. 

Steffen Hölldobler
Stochastic Local Search
procedure WalkSAT($F$, maxtries, maxsteps, select)
    input $F \in \mathcal{L}(\mathcal{R})$, maxtries, maxsteps $\in \mathbb{N}^+$
    heuristic function select
    output model of $F$ or "no solution found"
    for try = 1 to maxtries do
        $I$ = randomly choosen interpretation of $F$;
        for step = 1 to maxsteps do
            if $I \models F$ then
                return $I$
            end
            $C$ = randomly selected clause unsatisfied under $I$;
            $A$ = variable selected from $C$ according to select;
            $I$ = $I$ with $A$ flipped;
        end
    end
    return "no solution found"
end
Application of a Solver

► Consider **walksat**
  ▶ Check out the internet for **walksat**
  ▶ Walksat accepts .cnf-files and attempts to find a model
  ▶ E.g., `walksat -sol < axioms.cnf`

► WalkSAT as well as GSAT and GWSAT are sound but incomplete
Novelty

- Considers variables in the selected clauses sorted according to their score.
- If the best variable is not the most recently flipped one, it is flipped, otherwise, it is flipped with a probability $1 - p$, while in the remaining cases, the second-best variable is flipped,
  - where $p \in [0, 1]$ is a parameter called noise setting.
- Is in many cases substantially better than WalkSAT.
- It suffers from essential incompleteness.
Novelty +

- In each search step, with a user-specified probability $w_p$, the variable to be flipped is randomly selected from the selected clause, otherwise, the variable is selected according to the heuristics from Novelty

- Is probabilistically approximately complete

- In practice, $w_p = 0.01$ is sufficient

- Hoos: On the run-time behavior of stochastic local search algorithms for SAT. Proc. 16th National Conference on Artificial Intelligence (AAAI), 661-666: 1999
Adaptive Novelty

- Optimal value for noise $p$ varies significantly between problem instances

- **Idea**  Adapt $p$
  - Initially $p = 0$
  - Rapid improvement typically leading to stagnation
  - Increase the value of $p$ until escape from stagnation
  - Gradually decrease the value of $p$
  - Repeat this process until solution is found


- Implemented in the UBCSAT framework
Final Remarks

- **This section is based on** Hoos, Stützle: Stochastic Local Search. Morgan Kaufmann/Elsevier, San Francisco: 1998

- So far: stochastic local search
  - Sound but usually incomplete
  - Often quite fast

- Alternative: systematic search
  - Decides SAT problems
  - Sound and complete
  - May be too slow
  - In real applications it is often known that the problem is solvable