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 $\text{init}(\pi) : \mapsto \mathcal{D}(S(\pi) \times M(\pi))$
  - a step function $\text{step}(\pi) : S(\pi) \times M(\pi) \mapsto \mathcal{D}(S(\pi) \times M(\pi))$
  - a termination predicate $\text{terminate}(\pi) : S(\pi) \times M(\pi) \mapsto \{\bot, \top\}$

Steffen Hölldobler
Stochastic Local Search
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 SLSDecision(\(\pi\))
  input \(\pi \in \Pi\)
  output \(s \in S'(\pi)\) or “no solution found”
  \((s, m) = \text{selectRandomly}(S(\pi) \times M(\pi), \text{init}(\pi))\);
  while not \text{terminate}(\pi, s, m) do
    \((s, m) = \text{selectRandomly}(S(\pi) \times M(\pi), \text{step}(\pi, s, m))\);
  end
  if \(s \in S'(\pi)\) then
    return \(s\)
  else
    return “no solution found”
  end
end

where \text{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

- 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 $N(F, I, I')$ iff 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
  $$\text{step}(F, I, I') = \frac{1}{|\{I' \mid N(F, I, I')\}|} = \frac{1}{n} \text{ for all } I' \text{ with } N(F, I, I')$$
- $\text{terminate}(F, I)$ 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 $I \models F$. 

Steffen Hölldobler
Stochastic Local Search
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** $wp \in [0, 1]$
- **stepURW** the step function of uninformed random walk
- **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(\(\pi\), \(s\), \(wp\))

input \(\pi \in \Pi\), \(s \in S(\pi)\), \(wp \in [0, 1]\)

output \(s' \in S(\pi)\)

\(u = \text{random}([0, 1])\);

if \(u \leq wp\) then

\(s' = \text{stepURW}(\pi, s)\);

else

\(s' = \text{stepII}(\pi, s)\);

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
Randomized iterative improvement algorithm for SAT, but instead of stepII
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 stepURW 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 $t_t$ called tabu tenure
- Sometimes, an aspiration criterion is used to override the tabu status
The Step Function of Tabu Search

procedure stepTS(π, s, tt)
    input π ∈ Π, s ∈ S(π), tt
    output s′ ∈ S(π)
    N′ = admissibleNeighbours(π, s, tt);
    s′ = selectBest(N');
    return s'
end
The GSAT Architecture

- GSAT was one of the first SLS algorithms for SAT

- 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 available at that time
The Basic GSAT Algorithm

procedure \text{gsat}(F, \text{maxtries, maxsteps})
\begin{align*}
\text{input} & \quad F \in \mathcal{L}(\mathcal{R}), \text{maxtries, maxsteps} \in \mathbb{N}^+ \\
\text{output} & \quad \text{model of } F \text{ or ”no solution found”}
\end{align*}
\begin{align*}
\text{for } \text{try} = 1 \text{ to maxtries do} \\
& \quad I = \text{randomly chosen interpretation of } F; \\
& \quad \text{for } \text{step} = 1 \text{ to maxsteps do} \\
& \quad \quad \text{if } I \models F \text{ then} \\
& \quad \quad \quad \text{return } I \\
& \quad \quad \text{end} \\
& \quad \quad A = \text{randomly selected variable with maximal score}; \\
& \quad \quad I = I \text{ with } A \text{ flipped}; \\
& \quad \text{end} \\
& \quad \text{end} \\
& \quad \text{return ”no solution found”} \\
& \text{end}
\end{align*}
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 \( w_p \) 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 tabu iff $t - t_A \leq tt$. 

Steffen Hölldobler
Stochastic Local Search
procedure \texttt{WalkSAT}(F, \texttt{maxtries}, \texttt{maxsteps}, \texttt{select})

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

heuristic function \texttt{select}

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

for \texttt{try} = 1 to \texttt{maxtries} do

\hspace{2em} $I$ = randomly chosen interpretation of $F$;

for \texttt{step} = 1 to \texttt{maxsteps} do

\hspace{2em} if $I \models F$ then

\hspace{4em} return $I$

\hspace{2em} end

\hspace{2em} $C$ = randomly selected clause unsatisfied under $I$;

\hspace{2em} $A$ = variable selected from $C$ according to \texttt{select};

\hspace{2em} $I$ = $I$ with $A$ flipped;

\hspace{2em} end

end

end

return "no solution found"
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 \( wp \), 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, \( wp = 0.01 \) is sufficient.

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