STOCHASTIC LOCAL SEARCH
FOUNDATIONS AND APPLICATIONS

SLS Methods: An Overview

Holger H. Hoos & Thomas Stützle

Outline

1. Iterative Improvement (Revisited)
2. 'Simple' SLS Methods
3. Hybrid SLS Methods
4. Population-based SLS Methods

In II, various mechanisms (*pivoting rules*) can be used for choosing improving neighbour in each step:

- *Best Improvement* (aka *gradient descent*, *greedy hill-climbing*): Choose maximally improving neighbour

  Note: Requires evaluation of all neighbours in each step.

- *First Improvement:* Evaluate neighbours in fixed order, choose first improving step encountered.

  Note: Can be much more efficient than Best Improvement; order of evaluation can have significant impact on performance.

'Simple' SLS Methods

Goal:

Effectively escape from local minima of given evaluation function.

General approach:

For fixed neighbourhood, use step function that permits *worsening search steps*.

Specific methods:

- Randomised Iterative Improvement
- Probabilistic Iterative Improvement
- Simulated Annealing
- Tabu Search
- Dynamic Local Search

Randomised Iterative Improvement

Key idea: In each search step, with a fixed probability perform an uninformed random walk step instead of an iterative improvement step.

Randomised Iterative Improvement (RII):

determine initial candidate solution s
While termination condition is not satisfied:
  With probability *wp*:
    choose a neighbour s' of s uniformly at random
  Otherwise:
    choose a neighbour s' of s such that $g(s') < g(s)$ or,
      if no such s' exists, choose s' such that $g(s')$ is minimal
  $s := s'$

Note:

- No need to terminate search when local minimum is encountered

  Instead: Bound number of search steps or CPU time from beginning of search or after last improvement.

- Probabilistic mechanism permits arbitrary long sequences of random walk steps

  Therefore: When run sufficiently long, RII is guaranteed to find (optimal) solution to any problem instance with arbitrarily high probability.

- A variant of RII has successfully been applied to SAT (GWSAT algorithm), but generally, RII is often outperformed by more complex SLS methods.

Probabilistic Iterative Improvement

Key idea: Accept worsening steps with probability that depends on respective deterioration in evaluation function value: bigger deterioration $\cong$ smaller probability

Realisation:

- Function $p(g, s)$: determines probability distribution over neighbours of s based on their values under evaluation function g.
- Let $step(s)(s') := p(g, s)(s')$.

Note:

- Behaviour of PII crucially depends on choice of p.
- II and RII are special cases of PII.

Simulated Annealing

Key idea: Vary temperature parameter, *i.e.*, probability of accepting worsening moves, in Probabilistic Iterative Improvement according to *annealing schedule* (aka *cooling schedule*).

Inspired by physical annealing process:

- candidate solutions $\cong$ states of physical system
- evaluation function $\cong$ thermodynamic energy
- globally optimal solutions $\cong$ ground states
- parameter T $\cong$ physical temperature

Note: In physical process (*e.g.*, annealing of metals), perfect ground states are achieved by very slow lowering of temperature.

Simulated Annealing (SA):

determine initial candidate solution s
set initial temperature T according to *annealing schedule*
While termination condition is not satisfied:
  probabilistically choose a neighbour s' of s
    using *proposal mechanism*
  If s' satisfies probabilistic *acceptance criterion* (depending on T):
    $s := s'$
  update T according to *annealing schedule*

Note:

- 2-stage step function based on
  - proposal mechanism (often uniform random choice from $N(s)$)
  - acceptance criterion (often *Metropolis condition*)
- Annealing schedule (function mapping run-time t onto temperature $T(t)$):
  - initial temperature T_0
    (may depend on properties of given problem instance)
  - temperature update scheme
    (*e.g.*, geometric cooling: $T := \alpha \cdot T$)
  - number of search steps to be performed at each temperature
    (often multiple of neighbourhood size)
- Termination predicate: often based on *acceptance ratio*, *i.e.*, ratio of proposed steps to accepted steps.

‘Convergence’ result for SA:

Under certain conditions (extremely slow cooling),
any sufficiently long trajectory of SA is guaranteed to end in
an optimal solution [Geman and Geman, 1984; Hajek, 1998].

Note:

- Practical relevance for combinatorial problem solving is very limited (impractical nature of necessary conditions)
- In combinatorial problem solving, *ending* in optimal solution is typically unimportant, but *finding* optimal solution during the search is (even if it is encountered only once)!

Tabu Search

Key idea: Use aspects of search history (memory) to escape from local minima.

Simple Tabu Search:

- Associate *tabu attributes* with candidate solutions or solution components.
- Forbid steps to search positions recently visited by underlying iterative best improvement procedure based on tabu attributes.

Tabu Search (TS):

determine initial candidate solution s
While *termination criterion* is not satisfied:
| *determine set N' of non-tabu neighbours of s*
| *choose a best improving candidate solution s' in N'*
|
| *update tabu attributes* based on s'
| $s := s'$

Note:

- Non-tabu search positions in $N(s)$ are called *admissible neighbours of s*.
- After a search step, the current search position or the solution components just added/removed from it are declared *tabu* for a fixed number of subsequent search steps (*tabu tenure*).
- Often, an additional *aspiration criterion* is used: this specifies conditions under which tabu status may be overridden (*e.g.*, if considered step leads to improvement in incumbent solution).

Note: Performance of Tabu Search depends crucially on setting of tabu tenure tt:

- tt too low $\Rightarrow$ search stagnates due to inability to escape from local minima;
- tt too high $\Rightarrow$ search becomes ineffective due to overly restricted search path (admissible neighbourhoods too small)

Further improvements can be achieved by using *intermediate-term* or *long-term memory* to achieve additional *intensification* or *diversification*.

Examples:

- Occasionally backtrack to *elite candidate solutions*, *i.e.*, high-quality search positions encountered earlier in the search; when doing this, all associated tabu attributes are cleared.
- Freeze certain solution components and keep them fixed for long periods of the search.
- Occasionally force rarely used solution components to be introduced into current candidate solution.
- Extend evaluation function to capture frequency of use of candidate solutions or solution components.

Tabu search algorithms algorithms are state of the art for solving many combinatorial problems, including:

- SAT and MAX-SAT
- the Constraint Satisfaction Problem (CSP)
- many scheduling problems

Crucial factors in many applications:

- choice of neighbourhood relation
- efficient evaluation of candidate solutions (caching and incremental updating mechanisms)

Hybrid SLS Methods

The behaviour and performance of 'simple' SLS techniques can often be improved significantly by combining them with other SLS strategies.

Simple examples:

- Commonly used restart mechanisms can be seen as hybridisations with Uninformed Random Picking
- Iterative Improvement + Uninformed Random Walk = Randomised Iterative Improvement

Iterated Local Search

Key Idea: Use two types of SLS steps:

- *subsidiary local search* steps for reaching local optima as efficiently as possible (intensification)
- *perturbation steps* for effectively escaping from local optima (diversification).

Also: Use *acceptance criterion* to control diversification *vs* intensification behaviour.

Iterated Local Search (ILS):

determine initial candidate solution s
perform *subsidiary local search* on s
While termination criterion is not satisfied:

  $r := s$
  perform *perturbation* on s
  perform *subsidiary local search* on s

  based on *acceptance criterion*,
    keep s or revert to $s := r$

Note:

- *Subsidiary local search* results in a local minimum.
- ILS trajectories can be seen as walks in the space of local minima of the given evaluation function.
- *Perturbation phase* and *acceptance criterion* may use aspects of *search history* (*e.g.*, when the same local optima are repeatedly encountered, stronger perturbation steps may be applied).
- In a high-performance ILS algorithm, *subsidiary local search*, *perturbation mechanism*, and *acceptance criterion* need to complement each other well.

Subsidiary local search:

- More effective subsidiary local search procedures lead to better ILS performance.

  Example: When applying ILS to TSP, performance can be ranked as:

  2-opt $<$ 3-opt $<$ LK (Lin-Kernighan)

  where 3-opt is an iterative improvement algorithm based on the 3-exchange neighbourhood relation

- Often, subsidiary local search $=$ iterative improvement, but more sophisticated SLS methods can be used. (*e.g.*, Tabu Search).

Perturbation mechanism:

- Needs to be chosen such that its effect *cannot* be easily undone by subsequent local search phase.
  (In the simplest case, a random walk step in a larger neighbourhood than the one used by local search may be sufficient for achieving this goal)

  Example: local search = 3-opt, perturbation = 4-exchange steps in ILS for TSP.

- A perturbation phase may consist of one or more perturbation steps.

Perturbation mechanism (continued):

- Weak perturbation $\Rightarrow$ short subsequent local search phase; *but:* risk of revisiting current local minimum.
- Strong perturbation $\Rightarrow$ more effective escape from local minima; *but:* may have similar drawbacks as random restart.
- Advanced ILS algorithms may change nature and/or strength of perturbation adaptively during search.

Acceptance criteria:

- Always accept the *better* of the two candidate solutions

  ⇒ ILS performs Iterative Improvement in the space of local optima reached by subsidiary local search.

- Always accept the *more recent* of the two candidate solutions

  ⇒ ILS performs random walk in the space of local optima reached by subsidiary local search.

- Intermediate behaviour: select between the two candidate solutions based on the *Metropolis acceptance criterion* (*e.g.*, used in *Large Step Markov Chains* [Martin *et al.*, 1991].

- Advanced acceptance criteria take into account search history, such as the number of search steps since the last improvement of the *incumbent candidate solution*.

Iterated local search algorithms . . .

- are typically rather easy to implement (given existing implementation of subsidiary simple SLS algorithms);
- achieve state-of-the-art performance on many combinatorial problems, including the TSP.

Population-based SLS Methods

SLS methods discussed so far manipulate one single candidate solution of the given problem instance in each search step.

Straightforward extension: Use *population* (*i.e.*, set) of candidate solutions instead.

Note:

- The use of populations provides a generic way to achieve search diversification.
- Population-based SLS methods fit into the formal definition of an SLS algorithm from Chapter 1 by treating sets of candidate solutions as search positions.