Introduction to Stochastic Local Search

Steven Giangreco

October 2018

Formal Definition of SLS

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

- search space $S(\pi)$ containing finite set of candidate solutions $s \in S$,
- set of feasible solutions $S'(\pi) \subseteq S(\pi)$,
- neighborhood relation on $S(\pi)$ defined as $N(\pi) \subseteq S(\pi) \times S(\pi)$,
- finite set of memory states $M(\pi)$,
- initialization function $init(\pi) : \emptyset \to D(S(\pi) \times M(\pi))$ which specifies a probability distribution over initial search spaces and memory states,

Formal Definition of SLS (cont.)

- step function $step(\pi) : S(\pi) \times M(\pi) \to D(S(\pi) \times M(\pi))$ mapping each search position and memory state onto a probability distribution over its neighboring search positions and memory states, and
- termination predicate $terminate(\pi) : S(\pi) \times M(\pi) \to D(\{\top, \bot\})$ mapping each search position and memory state to a probability distribution over truth values indicating probability of search terminating on reaching a certain search point in search space and memory state

Neighborhood Relation

- Neighborhood relation can be defined as a function $N(s) := \{s' \in S \mid N(s, s')\} \subseteq S$ mapping each candidate solution to a set of other candidate solutions.
- In SAT problems, this relation is typically k-exchange wherein neighbor s' to candidate solution s is s with at most k variables having flipped truth values.
- For example the solution candidate $(x_1, x_2) = (\top, \top)$ has 1-exchange neighbors $(\top, \bot)$ and $(\bot, \top)$.

Iterative Improvement

- $\forall s \in S, init(s) := 1/|S|$
- Let $I(s) := \{s' \in S \mid s' \in N(s) \mathrm{and} g(s') < g(s)\}$. Then

$$step(s)(s') = \begin{cases} 1/|I(s)| & s' \in I(s) \\ 0 & \mathrm{otherwise} \end{cases}$$

- Always finds a local optimum
- High likelihood of getting bogged down in local optimum instead of finding better solution with backtracking

Escape Strategies

Approaching a local optimum without adequately low objective value is frequently-encountered problem. Can be reduced with a couple of strategies.

- Backtracking - Choose a non-improving step, typically involving randomness. Choosing policy of minimally worsening steps runs risk of plateauing.
- Restart - Re-initialize algorithm at a different initial solution candidate. Best used when there are few local minima and data can be readily re-initialized.

Variable Neighborhood Descent

Let $N_1, N_2, ...N_k$ be neighborhood relations ordered in increasing size. Algorithm works as follows:

- Find local optimum of N_1.
- Continue search for improvement in N_2. If none is found, continue search in N_3, and so on.
- If improvement is found, perform above two steps again. If no improvement is found with N_k, return with local optimum as optimal value.

Randomized Iterative Improvement

- Let $wp \in [0, 1]$ be a parameter determining likelihood of performing random walk instead of improvement step (this is called the walk probability, or noise parameter).
- $step(s)(s') := wp \cdot step_{URW}(s)(s') + (1 - wp) \cdot step_{II}(s)(s')$ where $step_{URW}(s)(s')$ is step function for uninformed random walk and $step_{II}(s)(s')$ is step function for iterative improvement (where least worsening step is chosen if set of strictly improving neighbors $I(s) = \emptyset$.
- Terminates after certain amount of time or steps, or after certain amount of steps without improvement.

Probabilistic Iterative Improvement

- The more a step would worsen the evaluation function at current position, the less likely it is to be performed
- $step(s)(s') = p(g, s)(s')$ where $p(g, s)$ is probability distribution function over neighboring candidate solutions of s depending on their evaluation function values.

Tabu Search

- Focus shifts to using search history instead of probabilistically worsening steps to escape local optimum.
- Typical restrictions are search positions visited or solution components used.
- Introduces parameter tt (called tabu tenure) indicating how long restrictions are memorized.
- Too small tt and search will stagnate; too large tt and search path may be too restricted.
- Aspiration criteria are conditions under which the restrictions may be lifted; often include an improvement in the incumbent candidate solution.

Iterated Local Search

- Steps alternate between approaching local optimum and escaping it.
- First step perturbs current candidate solution s, yielding s'.
- Second step finds the local optimum s'' of s'.
- Third step compares s and s'', starts again with better solution.

Greedy Randomized Adaptive Search Procedure (GRASP)

- Attempts to overcome limited candidate solution pool offered by greedy construction searches (which always add best improvement) by randomizing construction method.
- Randomly chooses solution component to add from list of highly-ranked components (called restricted candidate list, or RCL).
- RCL may be decided by value cutoff or cardinality restriction

Adaptive Iterated Construction Search

- Like GRASP, but weights are adapted to candidate solutions depending on their solution components and quality.
- The weights and a heuristic function h are used to probabilistically select components to be added to the candidate solution.

Ant Colony Optimization

- Multiple instances of the solver are run (called agents).
- Agents are able to communicate indirectly with "pheromone trails".
- At start of each iteration, a population of candidate solutions *sp* is generated with constructive search procedure.
- Solving works like adaptive iterated construction search, but pheromone trail levels are used instead of weights when adding solution components to partial candidate solutions.
- Perturbative local search may be applied to each element of *sp*, yielding set of locally optimum candidate solutions sp'.
- Best candidate solution in sp' is chosen to be incumbent candidate solution.
- Pheromone trail levels for solution components are updated.

Evolutionary Algorithms

- More direct interaction between agents than found in ant colony optimization.
- Three genetic operators (selection, mutation, recombination) replace a set of candidate solutions with a new set after each iteration (usually called a generation).
- Selection operator chooses (probabilistically) which candidate solutions to receive mutation and/or recombination or be added to the next generation.
- Mutation operator performs small, random change to a candidate solution.
- Recombination operator combines some information from multiple candidate solutions (called parents) to produce new candidate solution(s) (called offspring).