Stochastic Greedy Local Search

Chapter 7 @ Constraint Processing by Rina Dechter

Motivation

> A life spent making mistakes is not only more honorable, but more useful than a life spent doing nothing.
>
> *George Bernard Shaw*

- Two main exact methods for CSP solving: inference and search
- Both have exponential worst-case time (and space) complexity

- Stochastic greedy local search is approximation method
  - Appealing for requiring little memory
  - Start with arbitrary full assignment; resolve inconsistencies by local repairs

Stochastic greedy local search

- Greedy local search
  - The algorithm
  - Heuristics
- Random walk strategies
  - WalkSAT
  - Properties
- Hybrids of local search and inference

Greedy local search

- Implements greedy / *hill climbing* traversal of the search space
  - Does not guarantee finding a solution
- Explores search spaces whose states are complete assignments
  - Not necessarily consistent
  - Rather than partially consistent assignments

Greedy local search: the algorithm

- Start with a random complete assignment / instantiation
- Move from one complete assignment to the *next*
- *Cost* function estimates the distance between current assignment and solution
- Most greedy variant: change in value of variable gives best reduction
    - Requires a local change, and so the name local search
    - Stops when cost = 0 (*global minimum*) or no improvement possible changing one variable (*local minimum*)
    - Shortcoming: algorithm *gets stuck* in local minimum

procedure SLS

Input: A constraint network $\mathcal{R} = (X, D, C)$, $\boxed{\text{number of tries MAX\_TRIES}}$. A cost function defined on full assignments.

Output: A solution iff the problem is consistent, "false" otherwise.

1. **for** i = 1 to MAX_TRIES
   - **initialization: let** $\bar{a} = (a_1, \ldots, a_n)$ be a random initial assignment to all variables.
   - **repeat**
     (a) **if** $\bar{a}$ is consistent, return $\bar{a}$ as a solution.
     (b) **else** let $Y = \{\langle x_i, a'_i \rangle\}$ be the set of variable-value pairs that when x_i is assigned a'_i, give a maximum improvement in the cost of the assignment; pick a pair $\langle x_i, a'_i \rangle \in Y$,
     $\bar{a} \leftarrow (a_1, \ldots, a_{i-1}, a'_i, a_{i+1}, \ldots, a_n)$ (just flip a_i to a'_i).
   - **until** the current assignment cannot be improved.
2. **endfor**
3. **return** false

Greedy local search: example

- Propositional formula (0-1 CSP)
  $$\varphi = \{(\neg C), (\neg A \vee \neg B \vee C), (\neg A \vee D \vee E), (\neg B \vee \neg C)\}$$
- Initial assignment: all variables assigned value 1 (true)
  - Cost = 2 (first and last clauses violated)
- Next assignment?
  - Flipping A, D or E will not reduce cost
  - Flipping C to 0 satisfies violated clauses but violates second clause (cost=1)
  - Flipping B to 0 satisfies one violated clause (cost=1)
  - Decision: C=0
- Next assignment?
  - Flip B to 0 and find a solution

Greedy local search: improvements

- In the selection of the initial assignment
- In the nature of the local changes
  - Trying to escape local minima

Heuristics to improve local search (I)

- Plateau search
  - When a local optimum is reached, apply non-improving sideway moves

- Constraint weighting
  - The guiding cost function $F(\bar{a}) = \sum_i w_i * C_i(\bar{a})$ is a weighted sum of the violated constraints, where w_i is the current weight of C_i and $C_i(\bar{a}) = 1$ iff $\bar{a}$ violates C_i (0 otherwise)
  - First select variable/value pair that leads to largest reduction in F
  - Adjust weights
    - Increase +1 the weight of each constraint violated by current assignment
    - Local minima: no longer a local minima!

Heuristics to improve local search (II)

- Tabu search
  - Construct a list of the last n variable-value assignments
  - List forbids picking the same variable-value pairs – are *tabu*
- Other ideas…
  - Tie-breaking rules based on historic information
  - Value propagation over unsatisfied constraints when at local minima
- Automating MAX-TRIES/MAX-FLIPS
  - How to pick a value?
  - Continue search so long as there is progress, i.e. more constraints satisfied

Constraint weighting: example

- Propositional formula (0-1 CSP)
$$\varphi = \{(\neg C), (\neg A \vee \neg B \vee C), (\neg A \vee D \vee E), (\neg B \vee \neg C)\}$$
- Initially all weights $\omega_1, \dots, \omega_4$ are set to 1
- Initial assignment: all variables assigned value 1 (true)
  - Increment by 1 the weights of the violated clauses
  - $\omega_1 = \omega_4 = 2$, i.e. cost = 4
- Next assignment? C=0
  - Flipping C to 0 satisfies violated clauses but violates second clause (cost=1)
  - Flipping B to 0 keeps first clause unsatisfied (cost=2)
- Next assignment?
  - Flip B to 0 and find a solution

Random walk strategies

- Combine random walk with a greedy bias
  - Toward assignments that satisfy more constraints
- Stochastic element for escaping local minima

- WalkSAT
  - Some probability for choosing *worse* assignment
- *Simulated annealing*
  - *Noise model inspired by statistical mechanics*

WalkSAT

- Designed for SAT (0-1 CSP)
- First step: constraint violated randomly selected
- Second step:
  - With probability p: change the value of one variable in violated constraint
  - With probability (1-p): minimize number of constraints that become inconsistent
- Often identify value of p given class of problems

procedure WALKSAT

Input: A network $\mathcal{R} = (X, D, C)$, number of flips MAX_FLIPS, MAX_TRIES, probability p.

Output: "True," and a solution, if the problem is consistent, "false," and an inconsistent best assignment, otherwise.

1. **for** i = 1 to MAX_TRIES **do**
2. **start** with a random initial assignment $\bar{a}$.
3. Compare best assignment with $\bar{a}$ and retain the best.
4. **for** i = 1 to MAX_FLIPS
   - **if** $\bar{a}$ is a solution, **return** true and $\bar{a}$.
   - **else,**
     1. **pick** a violated constraint C, randomly
     2. **choose** with probability p a variable-value pair $\langle x, a' \rangle$ for $x \in scope(C)$, or, with probability $1 - p$, choose a variable-value pair $\langle x, a' \rangle$ that minimizes the number of new constraints that break when the value of x is changed to a' (minus 1 if the current constraint is satisfied).
     3. Change x's value to a'.
5. **endfor**
6. **return** false and the best current assignment.

WalkSAT: example

- Propositional formula (0-1 CSP)
$$\varphi = \{(\neg C), (\neg A \vee \neg B \vee C), (\neg A \vee D \vee E), (\neg B \vee \neg C)\}$$
- Initial assignment: all variables assigned value 1 (true)
  - Cost = 2 (first and last clauses violated)
- Randomly select unsatisfied clause $(\neg B \vee \neg C)$
  - Try to minimize violated constraints? Flip B to 0 (cost=1)
- Only one unsatisfied clause $(\neg C)$
  - Flip C to 0 (cost=0, i.e. solution found!)

Hybrids of local search

- Inference-based methods can help backtracking search
  - Forward checking, arc- and path-consistency
- Can inference-based methods help local search?

Effect of propagation on local search

- Certain classes of structured problems are easy for backtracking and hard for local search
- Problems become trivial for local search with local consistency
  - Local consistency changes the search space
  - Many near solutions are eliminated

Summary

- Stochastic local search
  - Randomized greedy scheme
  - Approximation of systematic search
  - Cannot prove inconsistency
- Start with a random complete instantiation
- Move to the next complete instantiation
  - Random walk in a cost function
- Stop at a glocal / local minimum
  - Escape from local minimum with restart / relaxing the improvement requirement


That's all Folks!