Lecture02b: Local Search II

CS540 1/25/18

Local Search (General Idea)

Goal: find a termination state
- Example: SAT (find a mapping of variables to {T,F})
- Counter-example: Path planning (output is a sequence of states)

Method
- Start with an initial state
- On every step:
  - Evaluate "neighboring" states (small variations of current state)
  - Move to neighboring state with better evaluation score

Variations on strategy
- First improvement
- Steepest ascent
- Evaluations on evaluation
- Evaluation function is the function to be optimized
- Evaluation function is a cheaper heuristic.

Constraints
- Finite memory
- Classic form: "memory-less"
- Modifications: finite (but non-zero) memory

Adding Memory

Add just one state of memory ("best so far")

Random Restart Local Search
- Run local search from random start position
- Save result as "best so far"
- Run local search N more times
- After each, if result is better than "best so far", replace

Note: any local search strategy will do
- Generally run lots of fast searches

Adding N States

Parallel Local Search
- Run N searches in parallel
- Compares N local optima, select best
- Similar to random restart

Beam Local Search
- Initialize with N random states
- Compute N neighborhoods
- Select N best elements from combined neighborhoods
- Repeat

Various heuristic additions to beam search...

Unlimited Memory

Tabu Search (Glover 1986)

Basic idea:
- Keep memory of visited states (tabu table)
- Do not return to tabu states

Deterministic search
- Never return to visited state

Stochastic search
- May be value to returning to visited state
- But odds of improvement are less

Issue: Memory size

Announcements

On-campus students:
You should have access to Canvas now (re-check)


Basic Tabu Search Algorithm

1. Initialize s and best-so-far
2. While termination criteria not satisfied
   1. Update memory (add last state)
   2. Generate neighbor solutions
   3. Prune neighbors that are tabu (in memory)
   4. Set s to best remaining neighbor
   5. Update best-so-far if necessary
3. Return best-so-far

Tabu Search Parameters

- **Size of memory / tabu tenure**
  - Fixed memory -> forgetting
  - Forgetting strategies
  - Age
  - Others...
- **Contents of memory**
  - States
  - State attributes
  - States on path to current best solution
  - States from which the next state was not chosen stochastically

Tabu Search Variants

- **Robust Tabu Search (Taillard 1991)**
  - Repeatedly choose tabu tenure randomly from an interval
- **Reactive Tabu Search (Batitti & Tecchiolli 1994)**
  - Dynamically adjust tabu tenure during search
  - Store step counts with states
  - Best-so-far always in tabu table
  - When neighbor revisited, check interval
  - Increase memory size with repetitions
  - Decrease memory size when no neighbors are repeats
  - Decrease memory size when ALL neighbors are repeats

Dynamic Local Search

- **Main theme**: separate state quality $H(s)$ from Evaluation function $G(s)$ that guides search
- **Modify the evaluation function $G(s)$ during local search**
- $G(s)$ changes at sub-termination local optima

DLS Algorithm

- Initialize state $s$
- Initialize best-so-far to $s$
- Initialize penalties to zero
- Initialize evaluation function $G() = h() – \text{penalties}()$
- While termination criteria not satisfied:
  - Perform local search, starting at $s$ and using $g()$, stopping at $s'$
  - If $H(s') > H(\text{best-so-far})$, best-so-far $\leftarrow s'$
  - Update penalties based on $s'$
  - $G() = h() – \text{penalties}()$.

Adapted from slides for Hoos & Stutzle's "Stochastic Local Search"

Discussion DLS

- What would a good penalty function be?