Metaheuristics
Chapter 2.5-2.8

Johan Falkenjack

NLPLAB
Dept. of Computer and Information Science
Linköping University

Overview

- Tabu Search (TS)
- Iterated Local Search (ILS)
- Variable Neighborhood Search (VNS)
- Guided Local Search (GLS)

Tabu Search (TS)

- Proposed by Glover in 1986
- Like Steepest Descent with a twist
- If all neighboring solutions are nonimproving, accept the best nonimproving solution.
- Problem: We'll get stuck in a loop between local optimum and it's best neighbour.
- Solution: Make the local optimum tabu (or taboo).

TS: Design issues

- Tabu list, representation in a large search space
- Aspiration criterion, when to break tabu
- Intensification (using medium-term memory)
- Diversification(using long-term memory)

TS: Short-Term Memory

- Serves local search
- How to store?
- Limit memory to k latest solutions
- How to increase size of k
  - Store hashed solutions
  - Store only some attributes of latest solutions
  - Store attributes of latest moves
- Static, dynamic or adaptive size?

TS: Medium-Term Memory

- Serves intesification
- Recency memory
- Store number of consequtive solutions with a given feature
- In other terms, store common features of elite solutions
- Create new solutions from these features if no progress is made
- Not always useful

TS: Long-Term Memory

- Serves diversification
- Frequency memory
- Store number of times a feature shows up in a solution
- Create new solutions with features rarely seen
- Strategy
  - Restart diversification, perturb current solution
  - Continous diversification, introduce bias
  - Strategic oscillation, switch between prioritizing infeasible and feasible solutions

Iterated Local Search (ILS)

- or Iterated Decent
- or Large-Step Markov Chains
- or Chained Local Optimization
- Smarter version of multistart local search
- Perturb locally optimal solutions and restart search
- Lin-Kernighan algorithm for TSP

ILS: Design issues

- Local search algorithm, any S-metaheuristic might do
- Perturbation method
  - Large perturbations if the search heuristic is effective for the landscape
  - Too large perturbations and we're doing random restart
  - Strategies
    - Static, Dynamic or Adaptive perturbation size
    - Random (Markov Chain) or Semideterministic (biased by memory) perturbations
- Acceptance criteria, probabilistic or deterministic

Variable Neighborhood Search (VNS)

- Proposed by Hansen & Mladenović in 1997
- Use a number of different neighborhood structures when traversing
- Based on Variable Neighborhood Descent (VND)
  - Define N neighborhoods $N_1 ... N_{max}$
  - Find local optimum in N_1 and set this as current solution
  - Switch to N_2
  - Search from the current solution to local optimum in N_2 and set this as current solution
  - Continue until local optimum in N_{max}

VNS: Stochastic VND

- VND is deterministic if neighborhoods and start point are predefined or generated deterministically
- VNS is not deterministic
  - Define N neighborhoods $N_1 ... N_{max}$
  - In N_k, pick a random neighbor x' to x
  - In N_k, find local optimum x'' from x' with some search algorithm
  - If $f(x'') < f(x)$, restart search in N_1 from x'', otherwise, restart search in N_{k+1}
  - Continue until some stopping criteria

Guided Local Search (GLS)

- Special case of Tabu Search
- Dynamically change the object function based on obtained local optima
- Penalize unfavorable features present in obtained local optium
- Object function:

$$f'(s) = f(s) + \lambda \sum_{i=1}^{m} p_i I_i(s)$$

GLS: Updating penalties

- Calculate utility of penalizing feature i based on local optimum s^*

$$u_i(s^*) = I_i(s^*)\frac{c_i}{1 + p_i}$$

- For the feature with highest utility, increment penalty by 1

GLS: Full procedure

- Set all penalties to 0
- Find initial local optimum s^* using some search algorithm
- Compute utilities for all features i in s^*
- Update penalty for feature with highest utility