DM841

Heuristics for Combinatorial Optimization

Metaheuristics to Enhance Construction Heuristics

Marco Chiarandini

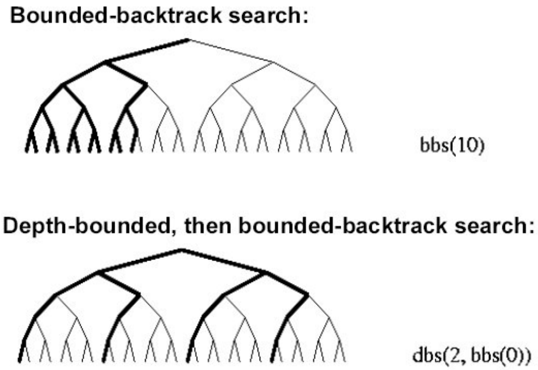
Department of Mathematics & Computer Science
University of Southern Denmark

Outline

1. Bounded backtrack
2. Limited Discrepancy Search
3. Random Restart
4. Rollout/Pilot Method
5. Beam Search
6. Iterated Greedy
7. GRASP

Outline

Bounded backtrack

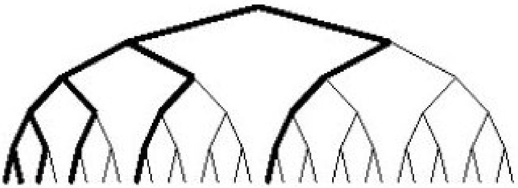


http://4c.ucc.ie/~hsimonis/visualization/techniques/partial_search/main.htm

Outline

Limited Discrepancy Search

Limited Discrepancy Search (LDS)



- Key observation that often the heuristic used in the search is nearly always correct with just a few exceptions.
- Explore the tree in increasing number of discrepancies, modifications from the heuristic choice.
- Eg: count one discrepancy if second best is chosen
  count two discrepancies either if third best is chosen or twice the second best is chosen
- Control parameter: the number of discrepancies

Outline

Randomization in Tree Search

The idea comes from complete search: the important decisions are made up in the search tree (backdoors - set of variables such that once they are instantiated the remaining problem simplifies to a tractable form)
$\leadsto$ random selections + restart strategy

Random selections

- randomization in variable ordering:
  - breaking ties at random
  - use heuristic to rank and randomly pick from small factor from the best
  - random pick among heuristics
  - random pick variable with probability depending on heuristic value
- randomization in value ordering:
  - just select random from the domain

Restart strategy in backtracking

- Example: $S_u = (1, 1, 2, 1, 1, 2, 4, 1, 1, 2, 1, 1, 4, 8, 1, \ldots)$

Outline

Rollout/Pilot Method

Derived from A*

- Each candidate solution is a collection of m components $S = (s_1, s_2, \ldots, s_m)$.
- Master process adds components sequentially to a partial solution $S_k = (s_1, s_2, \ldots s_k)$
- At the k-th iteration the master process evaluates feasible components to add based on an heuristic look-ahead strategy.
- The evaluation function $H(S_{k+1})$ is determined by sub-heuristics that complete the solution starting from S_k
- Sub-heuristics are combined in $H(S_{k+1})$ by
  - weighted sum
  - minimal value

Speed-ups:

- halt whenever cost of current partial solution exceeds current upper bound
- evaluate only a fraction of possible components

Outline

Beam Search

Again based on tree search:

- maintain a set B of bw (beam width) partial candidate solutions
- at each iteration extend each solution from B in fw (filter width) possible ways
- rank each $bw \times fw$ candidate solutions and take the best bw partial solutions
- complete candidate solutions obtained by B are maintained in B_f
- Stop when no partial solution in B is to be extended

Outline

1. Bounded backtrack
2. Limited Discrepancy Search
3. Random Restart
4. Rollout/Pilot Method
5. Beam Search
6. Iterated Greedy
7. GRASP

Iterated Greedy

(aka, Adaptive Large Neighborhood Search)

Key idea: use greedy construction

- alternation of construction and deconstruction phases
- an acceptance criterion decides whether the search continues from the new or from the old solution.

Iterated Greedy (IG):
determine initial candidate solution s
while termination criterion is not satisfied **do**
  $r := s$
  (randomly or heuristically) destruct part of s
  greedily reconstruct the missing part of s
  based on acceptance criterion,
    keep s or revert to $s := r$

Outline

1. Bounded backtrack
2. Limited Discrepancy Search
3. Random Restart
4. Rollout/Pilot Method
5. Beam Search
6. Iterated Greedy
7. GRASP

GRASP

Greedy Randomized Adaptive Search Procedure

Key Idea: Combine randomized constructive search with subsequent local search.

Motivation:

- Candidate solutions obtained from construction heuristics can often be substantially improved by local search.
- Local search methods often require substantially fewer steps to reach high-quality solutions when initialized using greedy constructive search rather than random picking.
- By iterating cycles of constructive + local search, further performance improvements can be achieved.

Greedy Randomized "Adaptive" Search Procedure (GRASP):

while *termination criterion* is not satisfied **do**
  generate candidate solution s using
    subsidiary greedy randomized constructive search
  perform subsidiary local search on s

- Randomization in *constructive search* ensures that a large number of good starting points for *subsidiary local search* is obtained.
- Constructive search in GRASP is 'adaptive' (or dynamic): Heuristic value of solution component to be added to a given partial candidate solution may depend on solution components present in it.
- Variants of GRASP without local search phase (aka *semi-greedy heuristics*) typically do not reach the performance of GRASP with local search.

Restricted candidate lists (RCLs)

- Each step of *constructive search* adds a solution component selected uniformly at random from a restricted candidate list (RCL).
- RCLs are constructed in each step using a *heuristic function* h.
  - RCLs based on cardinality restriction comprise the k best-ranked solution components. (k is a parameter of the algorithm.)
  - RCLs based on value restriction comprise all solution components l for which $h(l) \leq h_{min} + \alpha \cdot (h_{max} - h_{min})$, where h_{min} = minimal value of h and h_{max} = maximal value of h for any l. (α is a parameter of the algorithm.)
  - Possible extension: reactive GRASP (*e.g.*, dynamic adaptation of α during search)

Example: Squeaky Wheel

Key idea: solutions can reveal problem structure which maybe worth to exploit.

Use a greedy heuristic repeatedly by prioritizing the elements that create troubles.

Squeaky Wheel

- Constructor: greedy algorithm on a sequence of problem elements.
- Analyzer: assign a penalty to problem elements that contribute to flaws in the current solution.
- Prioritizer: uses the penalties to modify the previous sequence of problem elements. Elements with high penalty are moved toward the front.

Possible to include a local search phase between one iteration and the other