Good Complete Search Methods

CS410/510CS Lecture 4

Bart Massey <bart@cs.pdx.edu>
PSU CS Department
February 1, 2001

Overview

- Basic Complete Search
- Heuristics
- Heuristic Search
- Applications

Reminder: DFS

Simplest CSP search

- Pick vars, vals
- Bind one var at a time
- Backtrack to try each val for var
- Beware the triangle

Reminder: BFS

Good when

- Soln depth not uniform (CSPs?)
- Need/want shortest soln

BFS for Rubik's Cube: turn faces of cube until solution

BFS uses exponential space

Fixing BFS

Need to

- remove exp memory requirement
- retain BFS order

How to “remember” position?

Iterative Deepening

Idea: abandon systematic, keep complete!

- Do DFS with depth "cutoff" limit
- Evaluate only nodes at cutoff
- Achieves BFS order with poly memory
- Extra search, but not much
  - Node at depth k searched k times
  - Total "extra" work factor $O(d^2)$

Iterative Broadening

For DFS on CSP

- BFS/ID useless
- Problem is “early mistakes”

Try setting *breadth* limit and iterating DFS?

Evaluation Of Iterative Broadening

- Spreads search over tree
- k-th most node searched $b - k$ times, not bad
- Can get “stuck” in deep binary searches
- Useless for SAT w/o help
- Will revisit with heuristics

Iterative Sampling

Give up completeness? Even better:

- Choose random path to leaf:
  Done if goal, repeat if not!
- ISAMP
  - is "statistically complete"
  - adds $O(d)$ expected overhead
  - requires $O(1)$ space(!)
  - covers leaves OK

But not very satisfying...

Review: Heuristic

Heuristic is “rule of thumb” that usually works

Heuristic is “scoring function” $h(x)$ that guides search

Humans appear to operate by heuristic search rather than complete or stochastic search: c.f. “1-samp” later

Ordering Heuristics

- *Static* var ordering, val ordering: select at start of search
- *Dynamic* var ordering, val ordering: select "as you go"
- Search remains complete with dynamic
  - val order
  - var order
- Whether static or dynamic, can convert heuristic score to rank order

Dynamic Variable Ordering Complete

Note dynamic var order still complete: inductive proof

base case: With zero or one unbound variables, complete :-)

inductive case: With k + 1 unbound variables, select any var v; for each val of v, complete search for k remaining vars

Using Heuristics: 1-Samp

Heuristics for CSP? Choose best var, val to bind next via heuristics: give up if stuck (var has no possible val). Aka "1-Samp", does not work real well

Heuristic for optimization problem? Estimate $f(n)$ of best (max) score for node

Can follow *perfect* heuristic to optimum. Good heuristics give OK 1-samp scores

Admissible Heuristics

Special case: always "optimistic" (overestimated) heuristic $f'(n)$ (s.t. $f'(n) \geq f(n)$) is *admissible*

Perfect heuristic admissible. Any constant value better than possible also admissible, but not "tight"

Depth First Branch-and-Bound

Consider maximization with achievable value v and admissible heuristic f. Can prune nodes such that $f(n) \leq v$

Thus, in heuristic-guided DFS, will prune more as new solutions v_i are discovered!

DFBnB is common Operations Research technique, esp. with Integer Programming

Heuristic Iterative Deepening

Why heuristic ID? Can estimate solution depth. Pure ID = Dijkstra's algorithm on trees w/o memory

Admissible heuristic $f(n)$ constructible from $g(n) = \text{depth}(n)$ and optimistic estimate of remaining depth $h(n)$ as simply $f(n) = g(n) + h(n)$

IDA* search: ID with DFBnB on heuristic

Heuristic Iterative Broadening

Why heuristic IB? Good val-order heuristic crucial to IB success.

With good heuristics, IB can help with early mistakes

This kind of argument dangerous: good enough heuristics make 1-samp and variants work. Fine line...

Weighted ISAMP

WISAMP: instead of equal probability of all children n_i, weight using $h(n_i)$, e.g.

$$p_i = \frac{h(n_i)}{\sum_j h(n_j)}$$

Works strangely with depth:

$$p_{\langle n_1 \ldots n_k \rangle} = \prod_{i \in \{1 \ldots k\}} p_i$$

Bad mistakes mean trouble

Searches even with perfect heuristic(!)

Limited Discrepancy Search

Better idea:

- Use val-order heuristic only to choose single best value
- Count all “mistakes” equal
- Iterate on number of deliberate mistakes

Result: LDS

- Treats early, late mistakes identically
- Spreads search across tree
- Tries perfect first

Implementing LDS

Easy implementation; like IB, but instead of breadth limit, use "discrepancy limit"

Do LDS(0) = 1-SAMP, then LDS(1) = best n, then LDS(2) = best n^2. Each iteration repeats all previous (like IB)

Why leftmost = 0 disc. and all other = 1 disc? Empirical, plus small amount of analysis

Graphs?

Again, defer issue of search on graphs. Note that this is big deal: tents problem is effectively graph due to symmetry

Again, basic idea will be to burn storage to avoid re-search, or to give up and iterate. Note that these methods work on graphs; just lose systematicity

Credits

These algorithms do not come out of thin air. Much is due to Judea Pearl and Richard Korf. Will Harvey invented LDS while Matt Ginsberg's Ph.D. student. ISAMP is "obvious," but Pat Langley's 1992 paper revived interest

LOGS BOX STACKING Optimization

Consider optimization version of LOGS BOX STACKING. Goal: fill box as much as possible

Use var = "cell", value = log "filling cell". Note "big vs. little var" case: could use e.g. var = row, value = seq. of logs in row

Heuristics

Order vars left-to-right by row, and vals from largest log to smallest. Adjacency constraints and length constraints are thus satisfied.

Heuristic: box fill is $g(n) =$ *already filled* $+ h(n) =$ *fillable*. How to estimate h in polytime?

- Total - unfillable. Can find some unfillable easily
- Use 1-SAMP to fill remaining

First is admissable, second not. Second is looser, first not

Cole's Law

Student Brian Cole and I proved following: *unit propagation* by exactly filling row with one piece cannot hurt. This gives pruning rule: need not consider partial fill

Such pruning rules important in practice: use mathematics to avoid search within reason. But beware CPU time vs. human time

LOGS BOX STACKING and DFBnB

Can do BnB during search: if admissable $f(n)$ too small, then give up. In this case, if too many gaps in the box, done.

Points up problem with BnB: most configurations "close" to optimal (recall criticality graph). Thus, BnB gets smaller leverage than might expect

Optimizing TENT PACKING

Should now have good ideas about how to search TENT PACKING:

- Heuristics for var order, val order, evaluation
- Dynamic ordering techniques
- Better search algorithms

Homework 1c (not yet assigned) is to get good score on tents problem using these techniques plus techniques of next week