Search: The Core of AI

Keith L. Downing

January 5, 2014
A (preferably methodical) process for finding something.

- Searching for:
  - pre-existing entities (information, objects, etc.)
  - strategies for creating/designing entities.

- Examples:
  - Web search (for just about anything)
  - AI search - creates something, doesn’t just find it.

- Uninformed -vs- Informed: Do points in the search space give information that helps the searcher to determine the next step?

- Partial -vs- Complete solutions (i.e., attempts): Could the current state of search always be considered a complete solution, though not necessarily good or optimal? Or is it often a partial state that must be incrementally enhanced to become a solution?
  - E.g. 2 approaches to origami.
  - Closed-loop -vs- Open-loop control
Incremental Search: The 8-puzzle

Move Up

Move Down

Move Left

Keith L. Downing
Search: The Core of AI
Missionaries and Cannibals: Another Classic AI Puzzle

Popular AI puzzles: M and C, Towers of Hanoi, 8 puzzle, Cryptarithmetic, K-Queens, Knight’s Tour
3 Common Incremental Search Algorithms

Depth-First

Breadth-First

Best-First

Uninformed

Informed

Keith L. Downing
Search: The Core of AI
Properties of Incremental Search

- Search = choosing the next partial solution to expand.
- Requires representations that are easy to extend.
- High space complexity - need to save ALL nodes in the search tree (or graph).
- Time complexity varies - generally high for uninformed versions but potentially much lower for informed approaches.
- The heuristic is the key to A* performance. It needs to give good partial credit.
- With a good \textit{(admissible)} heuristic, optimal solutions are guaranteed to be found first with A*.
- Usually portrayed as the growth of a search tree (or graph).
Local Search
- The path is unimportant; only the final state matters.
- All search-space states = complete solutions (a.k.a. attempts).
- Search ↔ modifying complete solutions.
- Partial credit still very important, but now it’s given to whole (not partial) solutions.
Other Properties of Local Search

- Low space complexity - only need to save one (or a set) of current solutions, NOT paths back to the start state.
- Time complexity varies, though recent work indicates major improvements over incremental search for problems with densely-packed optimal solutions.
- Satisficing - can often find *reasonably good* solutions quickly.
- Requires representations that are easy to *tweak* to generate search-space neighbors.
- Uses an **objective function** to evaluate solutions. Similar to a heuristic but for complete solutions → less guesswork.
- Often portrayed as movement in a **landscape**.
What a Computer Sees during Local Search

Evaluations

0.54
Am I at a local optimum?

0.89

Where's the global optimum?

0.22

Representations

15, 22, 37, 99...
18, 33, 22, 86...
43, 11, 66, 84
Searching for Minimal Cable Networks

Example of the Minimal Spanning Tree (MST) problem.

Keith L. Downing  Search: The Core of AI
Incremental Search for an MST

Kruskal's and Prim's algorithms do this easy. It's not even considered search!
But the problem becomes much more complex if we can add intermediate nodes.

There is no omnipotent algorithm for STP that guarantees optimality (of the solution) and efficiency (of the algorithm). It is an NP-Complete problem.
1. Find longest side of triangle ABC (AC).
2. Draw equilateral triangle AQC opposite to point B.
3. The intersection of line BQ and the circle that circumscribes AQC is the Steiner point (P).
4. If angle ABC \( \geq 120 \) degrees, then P \( \leftarrow \) B.
Max of Steiner points is $|V| - 2$; $V =$ original vertices (Bern, 1989).

But there are $T = \binom{|V|}{3} = \frac{|V|(|V|-1)(|V|-2)}{6}$ possible point triples.

$\Upsilon = \sum_{i=0}^{|V|-2} \binom{|V|}{3}$ - For each possible number $i$ of Steiner points, from 0 to $|V| - 2$, there are $\binom{T}{i}$ ways to pick them.

| $|V|$ | $\Upsilon$ |
|------|------------|
| 3    | 2          |
| 4    | 11         |
| 5    | 176        |
| 10   | 903601306070 |
| 20   | 1466796429263525233849320188625720221593 |

$8 \times 10^{18} \approx$ Number of sand grains on earth.
Node-expansion ops: a) add the next 1, 2 or 3 edges using Kruskal’s algorithm, and b) add 1 or 2 new Steiner points (drawn as stars).

Vertical bar = relative arc cost.

Horizontal triangle = relative heuristic (h) value: estimated dist to goal.
Best-First Search Solutions

Length of MST of 6 original points = 24. Bottom left tree is optimal.
Begin with a set or population (P) of *individuals*solutions.

REPEAT

- Use an objective function (F) to evaluate each individual in P.
- If an individual $p^* \in P$ produces an optimal value for F, return $p^*$ and halt.
- Produce *children* of solutions in P.
- Update P by including some or all children and removing some or all parent solutions in P.

UNTIL a pre-defined maximum number of iterations have been performed.

Return the best (though not necessarily optimal) solution found.

Algorithms differ in how they produce and prioritize children.
Representations in Local Search

**Syntactic Representation**

\[
\{ [f^1, p^1_1, p^1_2, p^1_3], [f^2, p^2_1, p^2_2, p^2_3], \ldots, [f^k, p^k_1, p^k_2, p^k_3] \}
\]

- Encodes a list of point triples = bases for Steiner points.
- \( p^j \) = index of a point.
- \( f^i \) = flag; Only when flag =1 is steiner point created.
- Steiner-point indices can also appear in triples.
- Search occurs in the space of possible syntactic representations.

**Semantic Representation**

All points (original + Steiner) and edges (generated by Kruskal’s or Prim’s algorithm).

- This has *meaning* (i.e. semantics) in the problem domain.
- The evaluations of semantic representations govern search in syntactic space.
Development in Local STP Search

Kruskal's

\[ F = \frac{24.00 - 22.66}{24.00} \]
Using an evolutionary algorithm with 20 individuals and run for 200 generations.
Steiner Brains (Cherniak et. al., 1999)

Constraints of neural development produce **volume**-minimizing Steiner trees.

Top: A Steiner tree for transporting material from A to B and C. Middle: Thicker conduits require smaller branch angles. Bottom: A network that minimizes edge volume.
Intelligent generators in Classic AI (GOFAI), but not in nature nor in Bio-AI.