Recap: Search Problem

- **States**
  - configurations of the world
- **Successor function**:
  - function from states to lists of (state, action, cost) triples
- **Start state**
- **Goal test**

General Graph Search Paradigm

```plaintext
function tree-search(root-node)
  fringe  successors(root-node)
  explored  empty
  while ( notempty(fringe) )
    node  remove-first(fringe)
    state  state(node)
    if goal-test(state) return solution(node)
    explored  explored  {node}
    fringe  fringe  (successors(node) - explored)
  return failure
end tree-search
```

Fringe = priority queue, ranked by heuristic

Often: \( f(x) = g(x) + h(x) \)

Which Algorithm?

- Uniform cost search
- \( A^* \) using Manhattan
- Best-first search using Manhattan
Heuristics

It’s what makes search actually work

Admissible Heuristics

- $f(x) = g(x) + h(x)$
- $g$: cost so far
- $h$: underestimate of remaining costs

Where do heuristics come from?

Relaxed Problems

- Derive admissible heuristic from exact cost of a solution to a relaxed version of problem
  - For transportation planning, relax requirement that car has to stay on road → Euclidean dist
  - For blocks world, distance = # move operations  heuristic = number of misplaced blocks
- What is relaxed problem?

  # out of place = 2, true distance to goal = 3

  Cost of optimal soln to relaxed problem ≤ cost of optimal soln for real problem

Example: Pancake Problem

Action: Flip over the top $n$ pancakes

Cost: Number of pancakes flipped
Goal: Pancakes in size order

Example: Pancake Problem

**BOUNDS FOR SORTING BY PREFIX REVERSAL**

William H. Gates  
Microsoft, Albuquerque, New Mexico

Charitos H. Papadimitriou

Department of Electrical Engineering, University of California, Berkeley, CA 94720, U.S.A.

Received 18 January 1979.  
Revised 20 August 1979.

For a permutation $\sigma$ of the integers from 1 to $n$, let $f(\sigma)$ be the smallest number of prefix reversals that will transform $\sigma$ to the identity permutation, and let $(n)$ be the largest such $f(\sigma)$ for all $\sigma$ in the symmetric group $S_n$. We show that $f(\sigma) \leq \lceil \frac{3n}{2} \rceil$, and that $f(\sigma) \leq \lceil \frac{7n}{5} \rceil$ for $n$ a multiple of 16. Furthermore, each integer is required to participate in at most $n$ prefix reversals, the corresponding function $g(\sigma)$ is shown to obey $3g(\sigma) \leq g(\sigma) + 3n$.
**Example: Pancake Problem**

State space graph with costs as weights

---

**Example: Heuristic Function**

Heuristic: $h(x) = \text{the largest pancake that is still out of place}$

What is being relaxed?

---

**Counterfeit Coin Problem**

- Twelve coins
- One is counterfeit: maybe heavier, maybe light
- Objective:
  - Which is phony & is it heavier or lighter?
  - Max three weighings

---

**Coins**

- State = coin possibilities
- Action = weighing two subsets of coins
- Heuristic?
  - What is being relaxed?

---

**Traveling Salesman Problem**

Path =
1) Graph
2) Degree 2 (except ends, degree 1)
3) Connected

Kruskal's Algo: (Greedily add cheapest useful edges)

---

**Traveling Salesman Problem**

What can be relaxed?

Relax degree constraint
Assume can teleport to past nodes on path

Minimum spanning tree

Kruskal's Algorithm: $O(n^2)$
(Greedily add cheapest useful edges)
Traveling Salesman Problem

What can be relaxed?

Relax connected constraint

Cheapest degree 2 graph

Optimal assignment $O(n^2)$

Automated Generation of Relaxed Problems

- Need to reason about search problems
- Represent search problems in formal language

Planning

I have a plan - a plan that cannot possibly fail.
- Inspector Clousseau

Classical Planning

- Given
  - a logical description of the initial situation,
  - a logical description of the goal conditions, and
  - a logical description of a set of possible actions,
- Find
  - a sequence of actions (a plan of actions) that brings us from the initial situation to a situation in which the goal conditions hold.

Example: BlocksWorld

Planning Input:
State Variables/Propositions

- Types: block --- a, b, c
- (on-table a) (on-table b) (on-table c)
- (clear a) (clear b) (clear c)
- (arm-empty)
- (holding a) (holding b) (holding c)
- (on a b) (on a c) (on b a) (on b c) (on c a) (on c b)

No. of state variables = 16
No. of states = $2^{16}$
No. of reachable states = ?
Planning Input: Actions

- pickup a b, pickup a c, ...
- place a b, place a c, ...
- pickup-table a, pickup-table b, ...
- place-table a, place-table b, ...

Total: 6 + 6 + 3 + 3 = 18 “ground” actions
Total: 4 action schemata

Planning Input: Actions (contd)

- :action pickup ?b1 ?b2
  :precondition
  (on ?b1 ?b2)
  (clear ?b1)
  (arm-empty)

- :effect
  (holding ?b1)
  (not (on ?b1 ?b2))
  (clear ?b2)
  (not (arm-empty))

Planning Input: Initial State

- (on-table a) (on-table b)
- (arm-empty)
- (clear c) (clear b)
- (on c a)

- All other propositions false
  - not mentioned → assumed false
  - “Closed world assumption”

Planning Input: Goal

- (on-table c) AND (on b c) AND (on a b)

- Is this a state?

- In planning a goal is a set of states
  - Like the goal test in problem solving search
  - But specified declaratively (in logic) rather than with code

Specifying a Planning Problem

- Description of initial state of world
  - Set of propositions

- Description of goal:
  - E.g., Logical conjunction
  - Any world satisfying conjunction is a goal

- Description of available actions

Forward State-Space Search

- Initial state: set of positive ground literals
  - CWA: literals not appearing are false

- Actions:
  - applicable if preconditions satisfied
  - add positive effect literals
  - remove negative effect literals

- Goal test: does state logically satisfy goal?
- Step cost: typically 1
Heuristics for State-Space Search

- Count number of false goal propositions in current state
  - Admissible?
    - NO
- Subgoal independence assumption:
  - Cost of solving conjunction is sum of cost of solving each subgoal independently
  - Optimistic: ignores negative interactions
  - Pessimistic: ignores redundancy
- Admissible? No
- Can you make this admissible?

Importance of Heuristics

- $h_1$ = number of tiles in wrong place
- $h_2 = \sum$ distances of tiles from correct loc

Combining Admissible Heuristics

- Can always take max
- Could add several heuristic values
  - Doesn’t preserve admissibility in general

Heuristics for eight puzzle

- Start: 7 2 3 5 1 6 8
- Goal: 1 2 3 4 5 6 7 8

What can we relax?

Importance of Heuristics (contd)

- Delete all preconditions from actions, solve easy relaxed problem, use length
  - Admissible?
    - YES

- :action pickup-table ?b
  - :precondition (and (on-table ?b) (clear ?b) (arm-empty))
  - :effect (and (holding ?b) (not (on-table ?b)) (not (arm-empty)))

Decrease effective branching factor
Performance of IDA* on 15 Puzzle

- Random 15 puzzle instances were first solved optimally using IDA* with Manhattan distance heuristic (Korf, 1985).
- Optimal solution lengths average 53 moves.
- 400 million nodes generated on average.
- Average solution time is about 50 seconds on current machines.

Limitation of Manhattan Distance

- Solving a 24-Puzzle instance,
  - IDA* with Manhattan distance …
  - 65,000 years on average.
- Assumes that each tile moves independently
- In fact, tiles interfere with each other.
- Accounting for these interactions is the key to more accurate heuristic functions.

Example: Linear Conflict

3 1

1 3

Manhattan distance is 2+2=4 moves

Example: Linear Conflict

3 1

1 3

Manhattan distance is 2+2=4 moves

Example: Linear Conflict

3 1

1 3

Manhattan distance is 2+2=4 moves

Example: Linear Conflict

3 1

1 3

Manhattan distance is 2+2=4 moves
Example: Linear Conflict

Manhattan distance is 2+2=4 moves

Example: Linear Conflict

Manhattan distance is 2+2=4 moves

Example: Linear Conflict

Manhattan distance is 2+2=4 moves, but linear conflict adds 2 additional moves.

Linear Conflict Heuristic

- Hansson, Mayer, and Yung, 1991
- Given two tiles in their goal row, but reversed in position, additional vertical moves can be added to Manhattan distance.
- Still not accurate enough to solve 24-Puzzle
- We can generalize this idea further.

Pattern Database Heuristics

- Culberson and Schaeffer, 1996
- A pattern database is a complete set of such positions, with associated number of moves.
- E.g. a 7-tile pattern database for the Fifteen Puzzle contains 519 million entries.

Heuristics from Pattern Databases

31 moves is a lower bound on the total number of moves needed to solve this particular state.
Precomputing Pattern Databases

- Entire database is computed with one backward breadth-first search from goal.
- All non-pattern tiles are indistinguishable,
  - But all tile moves are counted.
- The first time each state is encountered, the total number of moves made so far is stored.
- Once computed, the same table is used for all problems with the same goal state.

Combining Multiple Databases

- Overall heuristic is maximum of 31 moves
- 31 moves needed to solve red tiles
- 22 moves need to solve blue tiles

Drawbacks of Standard Pattern DBs

- Since we can only take \( \text{max} \)
  - Diminishing returns on additional DBs
- Would like to be able to add values

Additive Pattern Databases

- Culberson and Schaeffer counted all moves needed to correctly position the pattern tiles.
- In contrast, we could count only moves of the pattern tiles, ignoring non-pattern moves.
- If no tile belongs to more than one pattern, then we can add their heuristic values.
- Manhattan distance is a special case of this, where each pattern contains a single tile.

Example Additive Databases

- The 7-tile database contains 58 million entries.
- The 8-tile database contains 519 million entries.

Computing the Heuristic

- Overall heuristic is sum, or 20+25=45 moves
- 20 moves needed to solve red tiles
- 25 moves needed to solve blue tiles
Performance

- **15 Puzzle**: 2000x speedup vs Manhattan dist
  - IDA* with the two DBs shown previously solves 15 Puzzles optimally in 30 milliseconds

- **24 Puzzle**: 12 million x speedup vs Manhattan
  - IDA* can solve random instances in 2 days.
  - Requires 4 DBs as shown
    - Each DB has 128 million entries
  - Without PDBs: 65,000 years

© Daniel S. Weld
Adapted from Richard Korf presentation