CS 581 – Advanced Artificial Intelligence

Topic: Search

Mustafa Bilgic

http://www.cs.iit.edu/~mbilgic

https://twitter.com/bilgicm

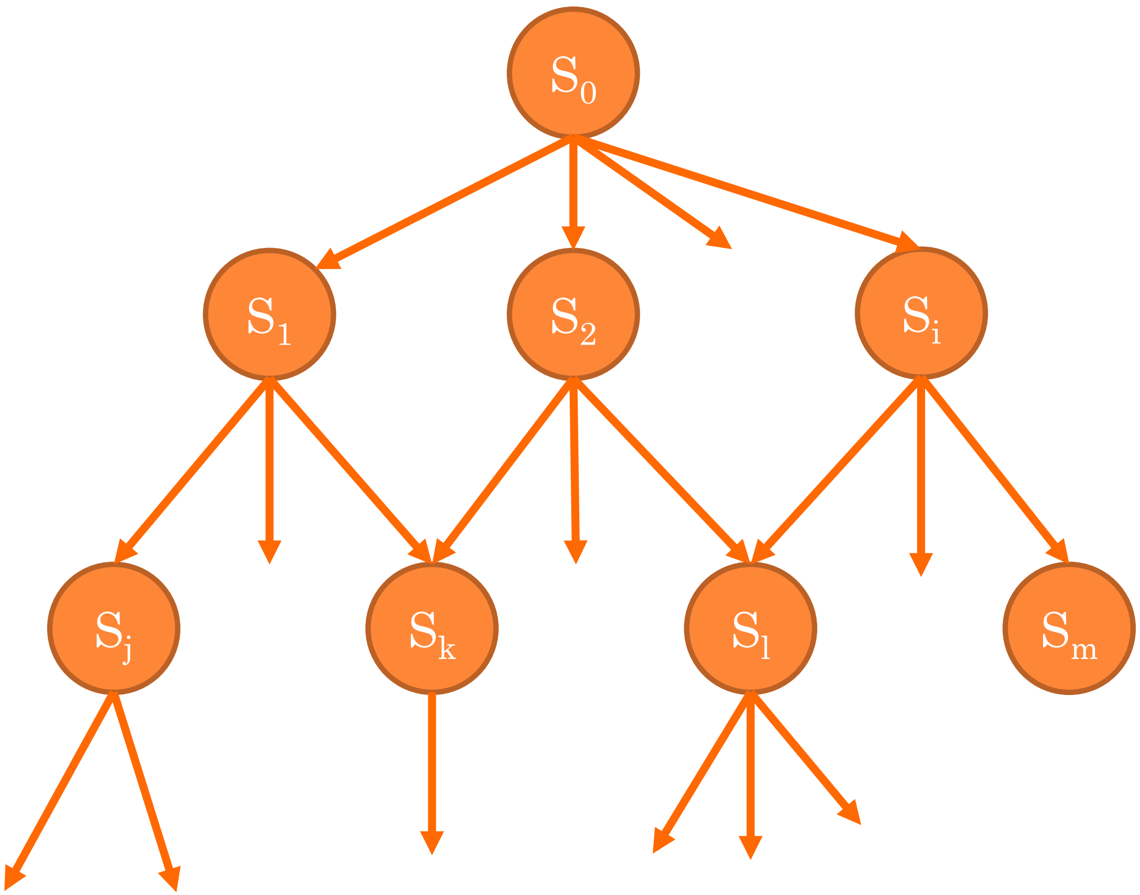
MOTIVATION

- AI is powered by search
- Consider multiple possibilities and find the “optimal” one
- Just a few examples
  - Travel from one location to another
    - Find the “optimal” route
  - Game playing
    - Find the “optimal” move
  - Machine learning
    - Find the “optimal” set of parameters
  - Machine translation
    - Find the “optimal” sequence of words
  - Constraint satisfaction
    - Find the “optimal” assignment
  - …

We will Study

- Hill climbing, A*, genetic algorithms, simulated annealing, …
- Maximum likelihood estimation, Bayesian estimation, expectation maximization, gradient optimization, Lagrange multipliers, policy search, …
- Alpha-beta search, Monte-Carlo tree search, …

Abstract Representation



Given

- An initial state
- A goal state
- Available actions at each state
- A transition model
- The cost of each action

Find

- A sequence of actions that take you from the initial state to the goal state, where the total cost of the sequence is minimized

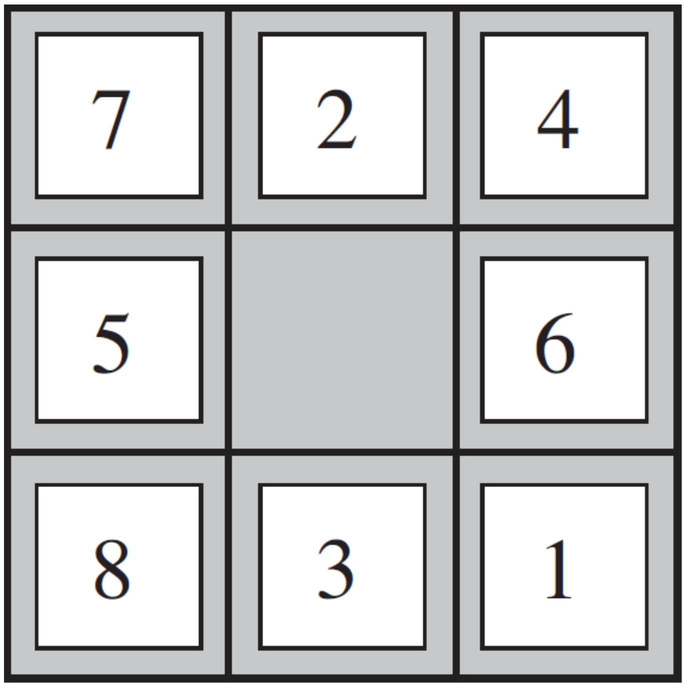
Travel Problem

- The world representation
  - In(City)
- Initial state
  - In(Madison, WI)
- Goal state
  - In(Detroit, MI)
- Actions
  - Travel to the neighboring city
- Transition model
  - If you are in city A and travel to city B, the state changes from In(A) to In(B)
- Cost
  - Distance traveled

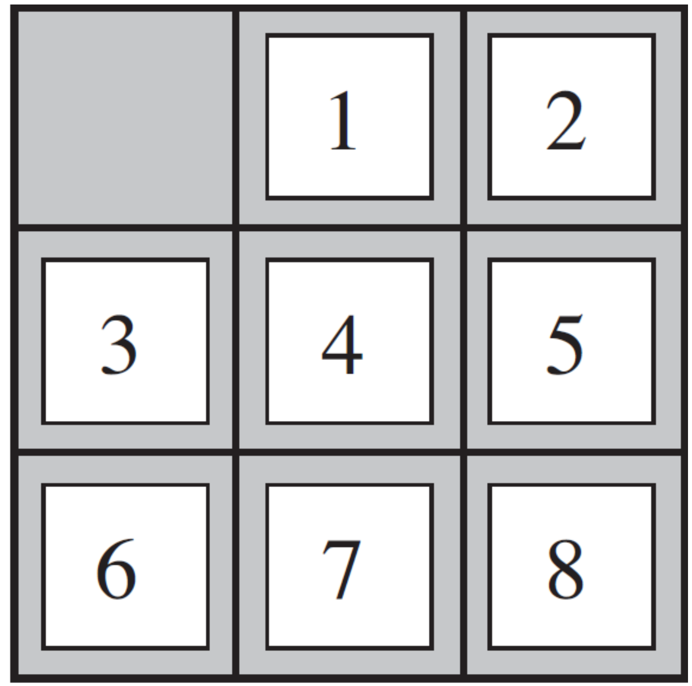
Traveling Salesman Problem

- Given
  - A list of N cities
  - Distances between each pair of cities
- Find
  - A minimum-cost travel plan where the agent visits each city exactly once and returns to the city of origin

8 Puzzle

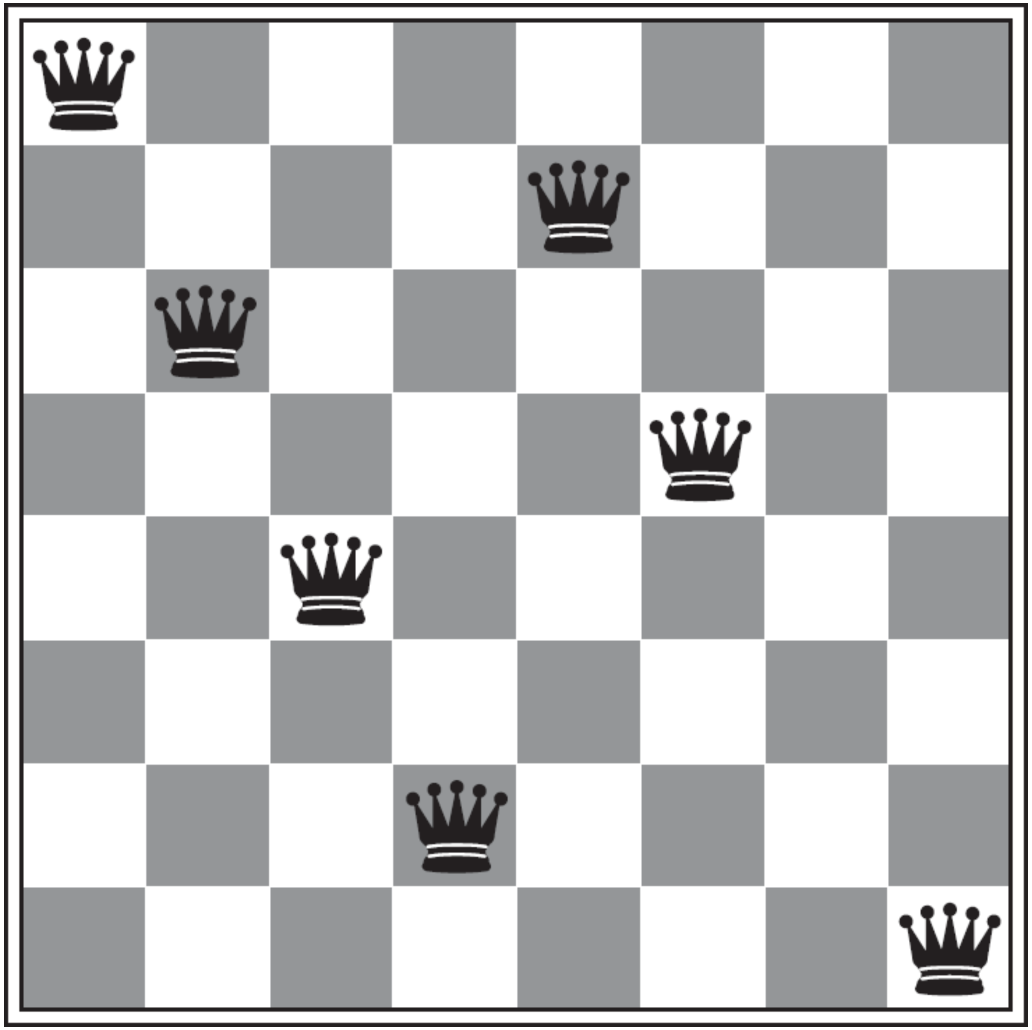


Start State



Goal State

8 Queens



OUTLINE

- Uniform cost search, greedy heuristic search, and A* search
  - E.g., travel problem, 3-puzzle problem
- Simulated annealing
  - E.g., traveling salesman problem
- Genetic algorithms
  - E.g., 8-queens problem

Exercise

- You are at an initial city D
- You are trying to reach to the city M
- The cities are not fully connected; you can travel from one city to only its neighbor cities
- You don't have access to the full map; you know only the parts you explored or discovered
- Each time you pick a city that you know and ask where you can go from there
- You want to find the shortest path from D to M, and you want to do it as fast as possible
- Notation:
  - $c(n, n)$: the cost of traveling from n to n'
  - $h(n)$: the estimate of the distance from n to the goal
- Three settings:
  - Setting 1: You are given the *c* function but not *h*
  - Setting 2: You are given the *h* function but not *c*
  - Setting 3: You are given both *c* and *h* functions

Best-first Search Pseudocode

- Given
  - The initial state, goal state, available actions, action costs, and maybe a heuristic function
  - `frontier` – a priority queue; `explored`: a set
- `Initialize frontier with the initial state`
- `While frontier is not empty`
  - `Pop the top node n in the frontier`
  - `If n is the goal state, return the solution`
  - `For each child n' of n:`
    - `If n' is not in explored`
      - `If n' is not in frontier`
        - `Add it to frontier`
      - `Else if the new path to n' is better than the old path, replace the old n' with the new n' in the frontier`
- `Return failure // frontier is empty; goal was not found`

BEST-FIRST SEARCH ALGORITHMS

- Define $f(n) = h(n) + g(n)$
  - $h(n)$: Cost estimate from n to the goal
  - $g(n)$: Cost from the initial state to n

1. Uniform-cost search
   - `frontier` is sorted using $g(n)$
2. Greedy heuristic search
   - `frontier` is sorted using $h(n)$
3. A* search
   - `frontier` is sorted using $f(n)$

A Travel Problem Example

- See OneNote

OPTIMAL

- Assume the optimal path cost is p^*
- Uniform cost search is guaranteed to return the optimal path
  - Expands all nodes where $g(n) \leq p^*$
- Greedy heuristic search is not optimal; ignores path costs
- A* is optimal only if h is admissible and consistent
  - Expands all nodes where $g(n) + h(n) \leq p^*$

Admissible? Consistent?

- **Admissible** if
  - *h(n)* never overestimates the optimal cost
  - That is *h(n)* is always optimistic
  - E.g., straight line distance between two cities
- **Consistent** if
  - If $h(n) \leq c(n,n') + h(n')$
  - n' is the successor of n
  - Triangle inequality
- If a heuristic is consistent, it is also admissible (the other way around is not guaranteed)

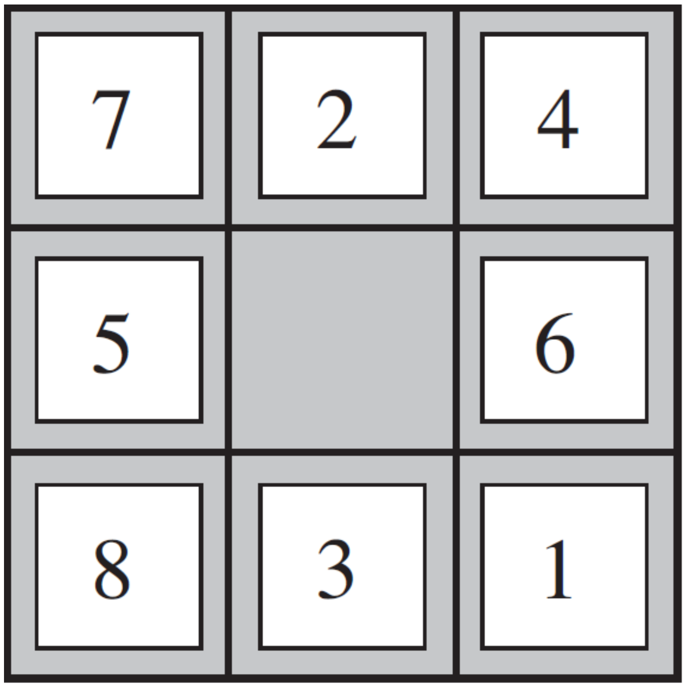
A* Optimality – Proof Sketch

1. If $h(n)$ is consistent, then $f(n)$ along any path is non-decreasing
2. When n is expanded, the optimal path to it has been found

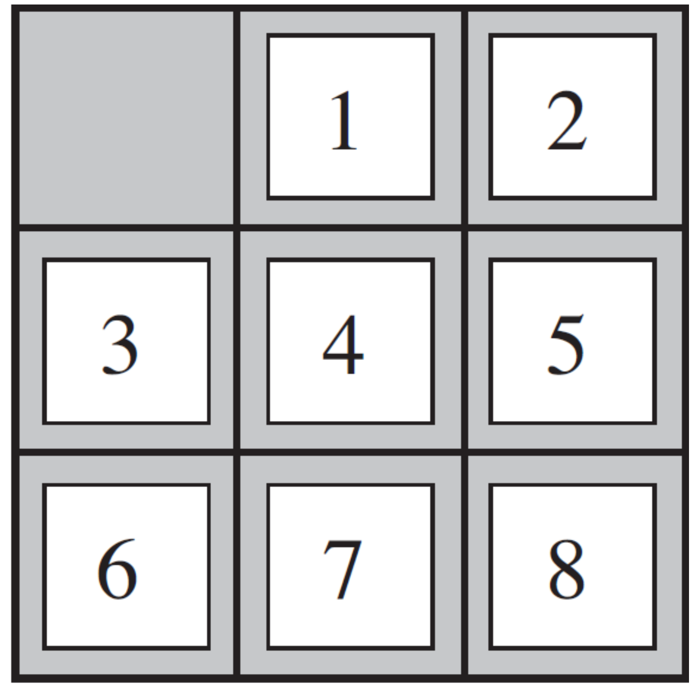
UCS vs A*

- A* = UCS if h(n) = 0 for all n
- UCS searches in circles whereas A* searches in ellipses
- Illustration in OneNote

8 Puzzle



Start State



Goal State

Heuristics

- How to design h(n)?
  - The answer is "depends on the domain" but some general principles exist
  - For example, relax the problem constraints a bit
- For travel problems
  - Flight distance
- For 8-Puzzle
  - Number of misplaced tiles
  - Manhattan distance

A 3-Puzzle Problem Example

- See OneNote

Other Search Algorithms

- Depth-first search
- Breadth-first search
- Iterative deepening search
- Bidirectional search

Comparing Algorithms

- Time complexity
- Space complexity
- Completeness
  - If there is a solution, can it find it?
- Optimality
  - Is the found solution the optimal one?

Comparing Heuristics

- h_i **dominates** h_j iff
  - $h_i(n) \geq h_j(n) \; \forall n$
- For the 8-Puzzles problem
  - Manhattan distance dominates # of misplaced tiles
- If you have multiple admissible heuristics where none dominates the other
  - Let $h(n) = \max(h_1(n), h_2(n), \ldots, h_m(n))$
  - $h(n)$ is admissible and it dominates all $h_i(n)$
  - Of course, computing a heuristic is not free, and hence we need to consider the computational cost of $h(n)$ compared to each $h_i(n)$

Effective Branching Factor

- If the total number of nodes generated is N and the solution depth is d, then
  - b^* is the branching factor that a uniform tree of depth d would need to have in order to contain $N+1$ nodes
- $N+1 = 1 + b^* + (b^*)^2 + \ldots + (b^*)^d$
- If A* finds a solution at depth 4 using 40 nodes, what is b*?
  - ≈ 2.182
- A good heuristic function achieves $b^* \approx 1$

Empirical Comparisons

- Figure 3.26 in http://aima.cs.berkeley.edu/figures.pdf
- https://github.com/aimacode/aima-python/blob/master/search4e.ipynb