Problem solving by searching

Milos Hauskrecht
milos@cs.pitt.edu
5329 Sennott Square

Example

• Assume a problem of computing the roots of the quadratic equation

\[ ax^2 + bx + c = 0 \]

Do you consider it a challenging problem?
Example

• Assume a problem of computing the roots of the quadratic equation

\[ ax^2 + bx + c = 0 \]

Do you consider it a challenging problem?
Hardly, we just apply the standard formula:

\[ x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \]

Solving problems by searching

• Some problems have a straightforward solution
  – Just apply a known formula, or follow a standardized procedure
    Example: solution of the quadratic equation
  – Hardly a sign of intelligence

• More interesting problems require search:
  – more than one possible alternative needs to be explored before the problem is solved
  – the number of alternatives to search among can be very large, even infinite.
Search example: Traveler problem

• Find a route from one city (Arad) to the other (Bucharest)

Example. Traveler problem

• Another flavor of the traveler problem:
  – find the route with the minimum length between S and T
Example. Puzzle 8.

- Find the sequence of the empty tile moves from the initial game position to the designated target position

<table>
<thead>
<tr>
<th>Initial position</th>
<th>Goal position</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 5 6 1 8 7 3 2</td>
<td>1 2 3 4 5 6 7 8</td>
</tr>
</tbody>
</table>


Find a configuration of n queens not attacking each other

A goal configuration

A bad configuration
A search problem

is defined by:

• **A search space:**
  – The set of objects among which we search for the solution
    Example: objects = routes between cities, or N-queen configurations

• **A goal condition**
  – What are the characteristics of the object we want to find in the search space?
    – Examples:
      • Path between cities A and B
      • Path between A and B with the smallest number of links
      • Path between A and B with the shortest distance
      • Non-attacking n-queen configuration

Search

• **Search (process)**
  – The process of exploration of the search space

• **The efficiency of the search depends on:**
  – The search space and its size
  – Method used to explore (traverse) the search space
  – Condition to test the satisfaction of the search objective
    (what it takes to determine I found the desired goal object)

• **Important to remember !!!**
  – You can choose the search space and the exploration policy
  – These choices can have a profound effect on the efficiency of the solution
Graph search

- Many search problems can be naturally represented as graph search problems.
- Typical example: Route finding
  - Map corresponds to the graph, nodes to cities, links to available connections between cities.
  - Goal: find a route (path) in the graph from S to T.

Graph search

- Less obvious conversion: Puzzle 8. Find a sequence of moves from the initial configuration to the goal configuration.
  - Nodes correspond to states of the game.
  - Links to valid moves made by the player.
- Note: the graph for some problem can become very large.
Graph search problem

- **States** - game positions, or locations in the map that are represented by nodes in the graph
- **Operators** - connections between cities, valid moves
- **Initial state** – start position, start city
- **Goal state** – target position (positions), target city (cities)

![Graph search diagram](image)

Graph search

- **More complex versions of the graph search problems:**
  - Find a minimum length path
    (= a route with the smallest number of connections, the shortest sequence of moves that solves Puzzle 8)

![Graph search diagram](image)
Graph search

- More complex versions of the graph search problems:
  - Find a minimum cost path
    (= a route with the shortest distance)

Graph search

- How to find the path in between S and T?
- A strawman solution:
  - Generate systematically all sequences of 1, 2, 3, … edges
  - Check if the sequence yields a path between S and T.
Graph search

Can we do better?

• We are not interested in sequences that do not start in S and that are not valid paths

• Solution:
  – Look only on valid paths starting from S
Graph search

- Being smarter about the space we search for the solution pays off in terms of the search process efficiency.

N-queens

Some problems can be converted to the graph search problems

- But some problems are harder and less intuitive
  - Take e.g. N-queens problem.

  ![Goal configuration]

- Problem:
  - We look for a configuration, not a sequence of moves
  - No distinguished initial state, no operators (moves)
N-queens

How to choose the search space for N-queens?

• Ideas?

Search space:

• Can we convert it to a graph search problem?
• We need states, operators, initial state and goal condition.

N-queens: solution 1

Search space:

– all configurations of N queens on the board
• Graph search:
  – States: configurations N queens
  – Operators: change a positions of one of the queens
  – Initial state: an arbitrary configuration
N-queens: solution 2

- **Search space:** configurations of 0, 1, 2, … N queens
- Graph search:
  - States configurations of 0, 1, 2, … N queens
  - Operators: additions of a queen to the board
  - Initial state: 0 queens on the board

Graph search

A trick: generate a configuration step by step (one queen per step)
- States (nodes) correspond to configurations of 0, 1, 2, 3, 4 queens
- Links (operators) correspond to the addition of a queen
- Initial state: no queens placed on the board
Graph search

N-queens problems
• This is a different graph search problem when compared to Puzzle 8 or Route planning:
  We want to find only the target configuration, not a path

Two types of graph search problems

• Path search
  – Find a path between states S and T
  – Example: traveler problem, Puzzle 8
  – Additional goal criterion: minimum length (cost) path

• Configuration search (constraint satisfaction search)
  – Find a state (configuration) satisfying the goal condition
  – Example: n-queens problem, design of a device with a predefined functionality
  – Additional goal criterion: “soft” preferences for configurations, e.g. minimum cost design
Search problem

Search problems that can be represented or converted into a graph search problems can be defined in terms of:

- **Initial state**
  - State (configuration) we start to search from (e.g. start city, initial game position)

- **Operators**:
  - Transform one state to another (e.g. valid connections between cities, valid moves in Puzzle 8)

- **Goal condition**:
  - Defines the target state (destination, winning position)

- **Search space** (the set of objects we search for the solution):
  - is now defined indirectly through:
    - the initial state + operators

---

Traveler problem.

**Traveler problem formulation:**

- **States**: different cities
- **Initial state**: city Arad
- **Operators**: moves to cities in the neighborhood
- **Goal condition**: city Bucharest
- **Type of the problem**: path search
- **Possible solution cost**: path length
Puzzle 8 example

Search problem formulation:
- **States:** tile configurations
- **Initial state:** initial configuration
- **Operators:** moves of the empty tile
- **Goal:** reach the winning configuration
- **Type of the problem:** path search
- **Possible solution cost:** a number of moves

N-queens problem

Problem formulation:
- **States:** configurations of 0 to 4 queens on the board
- **Initial state:** no-queen configuration
- **Operators:** add a queen to the leftmost unoccupied column
- **Goal:** a configuration with 4 non-attacking queens
- **Type of the problem:** configuration search
N-queens problem

Alternative formulation of N-queens problem

Problem formulation:
- **States**: different configurations of 4 queens on the board
- **Initial state**: an arbitrary configuration of 4 queens
- **Operators**: move one queen to a different unoccupied position
- **Goal**: a configuration with non-attacking queens
- **Type of the problem**: configuration search

Comparison of two problem formulations

**Solution 1:**
- **Operators**: switch one of the queens
  \[
  \binom{16}{4} \quad - \text{ all configurations}
  \]

**Solution 2:**
- **Operators**: add a queen to the leftmost unoccupied column
  \[
  1 + 4 + 4^2 + 4^3 + 4^4 < 4^5 \quad - \text{ configurations altogether}
  \]
Even better solution to the N-queens

Solution 2:

Operators: add a queen to the leftmost unoccupied column

< 4^5 - configurations altogether

**Improved solution** with a smaller search space

Operators: add a queen to the leftmost unoccupied column
such that it does not row-attack already placed queens

\[ \leq 1 + 4 + 4 \times 3 + 4 \times 3 \times 2 + 4 \times 3 \times 2 \times 1 = 65 \]

- configurations altogether

Formulating a search problem

- **Search (process)**
  - The process of exploration of the search space

- **The efficiency of the search depends on:**
  - The search space and its size
  - Method used to explore (traverse) the search space
  - Condition to test the satisfaction of the search objective
    (what it takes to determine I found the desired goal object)

- **Think twice before solving the problem by search:**
  - Choose the **search space** and the **exploration policy**