Problem solving by searching

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

Course administrivia

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

TA: Swapna Somasundaran
5422 Sennott Square
swapna@cs.pitt.edu

Course web page:
http://www.cs.pitt.edu/~milos/courses/cs1571/
Solving problems by searching

- Some problems have a straightforward solution
  - Just apply the 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. 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</td>
<td>1 2 3</td>
</tr>
<tr>
<td>6 1 8</td>
<td>4 5 6</td>
</tr>
<tr>
<td>7 3 2</td>
<td>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:
  – What is it?
  – A set of objects among which we search for the solution.
  – What is the search space for the puzzle 8 problem?

• A goal condition
A search problem

is defined by:

• **A search space:**
  – What is it?
  – A set of objects among which we search for the solution.
  – What is the search space for the puzzle 8 problem?
  – A sequence of moves.

• **A goal condition**
  – Give an example of a goal condition for the map problem.

• **A goal condition**
  – Give an example of a goal condition for the map problem.
  – A path from A to B.
  – The shortest path from A to B.
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 !!!
  - Conveniently chosen search space and the exploration policy 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 corresponds to states of the game,
  - links to valid moves made by the player

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**
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)

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](image)

  - **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:
– all configurations of N queens on the board

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

Search space:
– all configurations of N queens on the board

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

States are: N-queen configurations
Initial state: ?
Operators (moves)?

N-queens

Search space:
– all configurations of N queens on the board

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

Initial state: an arbitrary N-queen configuration
Operators (moves): change a position of one queen
N-queens

Is there an alternative way to formulate the N-queens problem as a search problem?

• Ideas?

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 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

Initial configuration

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

Valid goal configuration   Bad goal configuration
## Comparison of two problem formulations

**Solution 1:**

Operators: switch one of the queens

\[ \binom{16}{4} \] - all configurations

**Solution 2:**

Operators: add a queen to the leftmost unoccupied column

\[ 1 + 4 + 4^2 + 4^3 + 4^4 < 4^5 \] - 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 attack already placed queens

\[ \leq 1 + 4 + 4 \cdot 3 + 4 \cdot 3 \cdot 2 + 4 \cdot 3 \cdot 2 \cdot 1 = 65 \] - 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**

Search process

- Exploration of the state space through successive application of operators from the initial state
- A **search tree** = a kind of (search) exploration trace, with nodes corresponding to explored states
A search tree = a (search) exploration trace
- It is different from the graph defining the problem
- States can repeat in the search tree
General search algorithm

**General-search** \((\text{problem, strategy})\)
- **initialize** the search tree with the initial state of \text{problem}
- **loop**
  - if there are no candidate states to explore \textbf{return} failure
  - \textbf{choose} a leaf node of the tree to expand next according to \textit{strategy}
  - if the node satisfies the goal condition \textbf{return} the solution
  - \textbf{expand} the node and add all of its successors to the tree
- **end loop**
General search algorithm

**General-search** *(problem, strategy)*
*initialize* the search tree with the initial state of *problem*
*loop*
  *if* there are no candidate states to explore *return* failure
  *choose* a leaf node of the tree to expand next according to *strategy*
  *if* the node satisfies the goal condition *return* the solution
  *expand* the node and add all of its successors to the tree
*end loop*
General search algorithm

**General-search** (*problem, strategy*)
- **initialize** the search tree with the initial state of *problem*
- **loop**
  - if there are no candidate states to explore **return** failure
  - **choose** a leaf node of the tree to expand next according to *strategy*
    - if the node satisfies the goal condition **return** the solution
    - **expand** the node and add all of its successors to the tree
  - **end loop**
General search algorithm

General-search \((\text{problem, strategy})\)
initialize the search tree with the initial state of \text{problem}
loop
  if there are no candidate states to explore return failure
  choose a leaf node of the tree to expand next according to strategy
  if the node satisfies the goal condition return the solution
  expand the node and add all of its successors to the tree
end loop

• Search methods differ in how they explore the space, that is how they choose the node to expand next !!!!