Problem solving by searching

Milos Hauskrecht
milos@pitt.edu
5329 Sennott Square
Example

• Assume a problem of solving a linear equation

\[ 3x + 2 = 11 \]

Do you consider it a challenging problem?
Example

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

\[ 3x + 2 = 11 \]

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

\[ ax + b = c \]
\[ x = (c - b) / a \]
\[ x = 3 \]
Solving problems by searching

• Some problems have a straightforward solution
  – Just apply a known formula, or implement and follow a standardized procedure
  – Hardly a sign of intelligence

  **Example:** solution of linear or quadratic equations

• More interesting problems do not have a straightforward solution, and they 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: Path finding

- Find a path from one city to another city
Search example: Path finding

- Find a path from one city to another city
Example. Traveler problem

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

- Find a sequence of moves of tiles from the initial game position to the designated goal position.

Initial position

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

Goal position

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

Find a configuration of n queens on an $n \times n$ board such that queens do not attack each other.

A goal configuration

A bad configuration
A search problem

is defined by:

- **A search space:**
  - A set of objects among which we search for the solution
  - Examples: paths connecting two cities, or the different 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)
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)
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)
Search

• **Search (process)**
  – The process of exploration of the search space
• **Important**
  – We can often influence the efficiency of the search !!!!
  – We can be smart about choosing the search space, the exploration policy, and the design of the goal test
Graph representation of a search problem

- Search problems can be often represented using graphs
- **Example:** Finding a path on a map
  - Map corresponds to the graph, **nodes to cities, links to valid moves via available connections**
  - **Goal:** find a path (sequence of moves) in the graph from the start to the target city
Graph search problem

• A **graph search problem** is defined by:
  – A **state space** (all game positions, or all cities in the map)
  – **Operators** (= valid moves, actions transforming the states)
  – A **start state and a goal state**

• **State space graph:** a graph where states = nodes, operators = links

• A **solution** of the **path finding problem** is a sequence of operators that transforms the start state to a goal state (this sequence is also called a **plan**).
Puzzle 8 as a graph search problem

Puzzle 8. Find a sequence of moves from the initial configuration to the goal configuration.

Conversion to the state space graph

- nodes corresponds to states of the game,
- links to valid moves made by the player
Graph search problem

- More complex versions of the graph search problem:
  - Find the minimal length path
    (= a path with the smallest number of connections, or the shortest sequence of moves that solves Puzzle 8)
Graph search problem

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

Some problems are easy to formalize as graph search problems

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

  ![Goal configuration](image)

  **Goal configuration**

  • Problem:
    – We look for a configuration, not a sequence of moves
    – No distinguished initial state, no operators (moves)
N-queens as a graph search problem

Can we convert N-queens to a graph search problem?

- We need states, operators, initial state and goal condition.

  States: ?
  Initial state: ?
  Operators (moves): ?
  Goal state: ?
N-queens as a graph search problem

Can we convert N-queens to a graph search problem?

• We need states, operators, initial state and goal condition.

  **States:** ?
  Initial state: ?
  Operators (moves): ?
  Goal state: ?

How to choose the state space for N-queens?

Assume the **state space** = all configurations of N queens on the board
N-queens as a graph search problem

Can we convert N-queens to a graph search problem?

- We need states, operators, initial state and goal states.

**States:** all N-queen configurations

**Initial state:** ?

**Operators (moves):** ?

**Goal state:** ?
Can we convert N-queens to a graph search problem?

- We need states, operators, initial state and goal states.

**States:** all N-queen configurations

**Initial state:** ?

**Operators (moves):** ?

**Goal state:** ?

Goal
N-queens as a graph search problem

The state space:

Can we convert N-queens to a graph search problem?

- We need **states**, **operators**, **initial state** and **goal state**.

**States:** all N-queen configurations

**Initial state:** ?

**Operators (moves):** ?

**Goal state:** states satisfying the goal condition
N-queens as a graph search problem

The state space: 

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

States: all N-queen configurations
Initial state: ?
Operators (moves): ?
Goal state: states satisfying the goal condition
N-queens as a graph search problem

The state space:

Can we convert N-queens to a graph search problem?

- We need **states, operators, initial state and goal state**.

  **States**: all N-queen configurations
  
  **Initial state**: an arbitrary N-queen configuration
  
  **Operators (moves)**: change a position of one queen
  
  **Goal state**: states satisfying the goal condition
N-queens as a graph search problem

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

• Can we choose a different state space? Operators? Initial state?
N-queens as a graph search problem

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

• A new graph search problem:
  – **States:** configurations of 0, 1, 2, … N queens
  – **Operators:** additions of a queen to the board
  – **Initial state:** no queens on the board
  – **Goal state:** states satisfying the goal condition
N-queens as a graph search problem

N-queens problems

- This is a different graph search problem when compared to Puzzle 8 or Path planning:
  We want to find only the target configuration, not a path
Two types of graph search problems

Depending on the solution we seek we can distinguish:

- **Path search (planning) problems**
  - Solution is a path between states S and T
  - **Example**: traveler problem, Puzzle 8
  - **Additional goal criterion**: minimum length (cost) path

- **Configuration search problems**
  - Solution is a state (configuration) satisfying the goal condition.
  - **Example**: n-queens problem
  - **Additional goal criterion**: “soft” preferences on configurations, e.g. minimum cost design
Graph search problem

Search problems can be converted into a graph search problems:

• **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)

• **State space:**
  – Defined indirectly through: the initial state + operators

• **Solution:** Either a sequence of operators from S to T, or a goal (target) state
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

Formulation 1:

- **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

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
Real-world path-search problems

Route finding/navigation:

- **States**: roads, exits, intersections
- **Initial state**: current address
- **Operators**: moves to roads, exits, intersections
- **Goal condition**: target address
- **Type of the problem**: path search
- **Possible solution cost**: time, distance, tolls
Real-world configuration-search problems

Classroom scheduling:

- **States**: assignment of times, rooms, classes, teachers
- **Initial state**: arbitrary and possibly conflicting assignment
- **Operators**: changes of assignments
- **Goal condition**: non-conflicting schedule assignment
- **Type of the problem**: configurations search
- **Possible solution cost**: minimize class and teachers’ gaps,
Real-world configuration-search problems

VLSI design:

• **States:** circuit layout and connections
• **Initial state:** initial layout
• **Operators:** changes of layout of elements, connections
• **Goal condition:** configuration satisfying the design constraints
• **Type of the problem:** configuration search
• **Possible solution cost:** minimize connection distances, stray capacities, energy consumption
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)
N-queens problem: formulation 1

Formulation 1:

- **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

**Problem formulation:**

Bad goal configuration  Valid goal configuration
N-queens problem: formulation 2

Formulation 2:

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
Comparison of two problem formulations

Solution 1:

Operators: switch one of the queens

State space size: \( \binom{16}{4} \)

Solution 2:

Operators: add a queen to the leftmost unoccupied column

State space size: \( 1 + 4 + 4^2 + 4^3 + 4^4 < 4^5 \)
Even better solution to the N-queens

Solution 2:

Operators: add a queen to the leftmost unoccupied column

State space size: $< 4^5$

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

State space size:

$\leq 1 + 4 + 4 \times 3 + 4 \times 3 \times 2 + 4 \times 3 \times 2 \times 1 = 65$
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)

Think twice before solving the problem: Choose the search space wisely
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)
Search

• **Search (process)**
  – The process of exploration of the search space
• **The efficiency of the search depends on:**
  – We can often influence the efficiency of the search !!!!
  – We can be smart about choosing the search space, the exploration policy, and the design of the goal test
Search process

Exploration of the state space through successive application of operators from the initial state

- Search tree = structure representing the exploration trace
  - Is built on-line during the search process
  - Branches correspond to explored paths, and leaf nodes to the exploration fringe
Search tree

- A search tree = (search) exploration trace
  - different from the graph representation of the problem
  - states can repeat in the search tree

State space graph

Search tree built by exploring paths starting from city Arad
A branch in the search tree = a path in the graph
A branch in the search tree = a path in the graph

Search tree
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 next 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 next 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

Arad ← Node chosen to be expanded next
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 next return failure
  choose a leaf node of the tree to expand next according to \(\text{strategy}\)
  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 next 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

CS 2710 Foundations of AI
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 next *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 next 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 next **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**

Expanded nodes

Generated (active, open) nodes

CS 2710 Foundations of AI
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 next 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 **problem**
- **loop**
  - **if** there are no candidate states to explore next **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 next return \text{failure}
  choose a leaf node of the tree to expand next according to a \text{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 !!!!!
Implementation of search

- Search methods can be implemented using the **queue** structure and a **queuing function** $f$

**General search** $(\text{problem}, \text{Queuing-fn})$

$$\text{nodes} \leftarrow \text{Make-queue}(\text{Make-node}(\text{Initial-state}(\text{problem})))$$

**loop**

  - if nodes is empty then return failure
  - $\text{node} \leftarrow \text{Remove-node}(\text{nodes})$
  - if Goal-test$(\text{problem})$ applied to State$(\text{node})$ is satisfied then return node
  - $\text{nodes} \leftarrow \text{Queuing-fn}(\text{nodes}, \text{Expand}(\text{node}, \text{Operators}(\text{node})))$

end loop

- Candidates (**search tree nodes**) are added to the queue structure

- **Queuing function** $f$ determines what node will be selected next
Implementation of search

- A *search tree node* is a data-structure that is a part of the search tree.

![Diagram of a search tree node with attributes](image)

**Attributes:**
- state value $f$ (cost)
- depth
- path cost

- **Expand function** – applies Operators to the state represented by the search tree *node*. Together with *queuing-function* $f$ it fills the attributes.
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)