Problem Solving and Search in Artificial Intelligence

Uninformed Search Strategies

Nysret Musliu
Database and Artificial Intelligence Group
Institut für Informationssysteme, TU-Wien
Introduction

- Many classic algorithms are designed to search spaces for an optimum solution
- Broadly, classic methods fall into two classes:
  - Algorithms that only evaluate complete solutions
    - Exhaustive search, local search, …
  - Algorithms that require the evaluation of partially constructed or approximate solutions
    - Branch and bound strategies, …
Exhaustive Search

- Checks every solution in the search space until the best global solution has been found.
- Can be used only for small instances of problems.
- Exhaustive (enumerative) algorithms are simple.
- Search space can be reduced by backtracking search.
- Some optimization methods, e.g., branch and bound and A* are based on an exhaustive search.
Exhaustive Search

- How can we generate a sequence of every possible solution to the problem?
  - The order in which the solution are generated is irrelevant
  - Depends from selected representation
Enumerating the SAT

- We have to generate every possible binary string of length $n$.
- All solutions correspond to whole numbers in a one-to-one mapping.
- Generate all non-negative integers from 0 to $2^n - 1$ and convert each of these integers into the matching binary string of length $n$.

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>0</td>
<td>0001</td>
<td>1</td>
<td>0100</td>
<td>4</td>
</tr>
<tr>
<td>0010</td>
<td>2</td>
<td>0101</td>
<td>5</td>
<td>1000</td>
<td>8</td>
</tr>
<tr>
<td>0011</td>
<td>3</td>
<td>0110</td>
<td>6</td>
<td>1001</td>
<td>9</td>
</tr>
<tr>
<td>1010</td>
<td>10</td>
<td>1101</td>
<td>13</td>
<td>1011</td>
<td>11</td>
</tr>
<tr>
<td>1110</td>
<td>14</td>
<td>1111</td>
<td>15</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Enumerating the SAT

- Tree representations of search space for SAT

\[ x_1 = F \quad x_1 = T \]
\[ x_2 = F \]
\[ x_2 = T \quad x_2 = F \quad x_2 = T \]
\[ x_3 = F \quad x_3 = T \]
Enumerating TSP

- How to generate all possible permutations?
- If some cities are not connected, some permutation might not be feasible
Procedure gen1-permutation(i)
Begin
  k=k+1
  P[i]=k
  if k=n then
    for q=1 to n do
      print P[q]
    for q=1 to n do
      if P[q]=0 then gen1-permutation(q)
  k=k-1
  P[i]=0
end
Uninformed search strategies

- Breadth first search
- Depth first search
- Depth limited search
- Iterative deepening search
Breadth First Search

- The shallowest unexpanded node is first expanded
Breadth First Search

- The shallowest unexpanded node is first expanded

```
x1=T
x2=T
x2=F
x2=F
x3=F  x3=T
```

```
x1=F
x1=T
x2=F
x2=T
x2=F
x2=T
x3=F  x3=T
```
Breadth First Search

- The shallowest unexpanded node is first expanded
Breadth First Search

- The shallowest unexpanded node is first expanded
Breadth First Search
Breadth First Search

\[ x_1 = T \]
\[ x_2 = T \]
\[ x_2 = F \]
\[ x_3 = F \]
\[ x_3 = T \]
**Depth First Search**

**Procedure** depth-first(v)

**Begin**

visit v

for each child w of v do

depth-first(w)

end
Depth First Search

Procedure depth-first(v)
Begin
    visit v
    for each child w of v do
        depth-first(w)
    end
end
Procedure depth-first(v)

Begin

visit v

for each child w of v do

depth-first(w)

end
Depth First Search

Procedure depth-first(v)
Begin
    visit v
    for each child w of v do
        depth-first(w)
    end
end
Backtracking

- Suppose the SAT formula contains a clause: \((x_1 \lor x_2)\)...
Backtracking

- Suppose the SAT formula contains a clause: \((x_1 \lor x_2)\)...
Suppose the SAT formula contains a clause: \( \ldots (x_1 \lor x_2) \ldots \)

Remaining branches below this node can lead to nothing but a dead end.
Backtracking

- Suppose the SAT formula contains a clause: \((x_1 \lor x_2)\)...

Remaining branches below this node can lead to nothing but a dead end.
Search algorithms performance

- Completeness
  - Does the algorithm finds a solution if that exists

- Optimality
  - Does the algorithm finds an optimal solution

- Time Complexity
  - Number of expanded nodes

- Space Complexity
  - Memory needed to perform the search
Search algorithms performance

b : branching factor (maximum number of successors)

d : the depth of the shallowest goal node in the search tree

m : maximum depth of the tree
Breadth first search performance

- Complete if \( b \) is finite
- Optimal if the cost of each step is 1
- Time:
  \[
  1 + b + b^2 + \ldots + b^d = \frac{(b^{d+1}-1)}{(b-1)} = O(b^d)
  \]
- Space:
  \( O(b^d) \)
Depth first search performance

- Complete if the search tree is finite
- Not optimal
- Time:
  \[ 1 + b + b^2 + \ldots + b^m = O(b^m) \]
- Space:
  \[ O(bm) \]
Depth-limited search

- Depth first search with depth limit L
  - Nodes at depth L are not expanded
- Eliminates problem with infinite path
- How to select L?
- Possible failures:
  - No solution
  - Cutoff – no solution within the depth limit
Iterative deepening depth first search

- Repeat Depth-limited search with \( L=1,2,3,\ldots \)
Iterative deepening depth first search

Limit = 2
Iterative deepening depth first search

Limit = 3
Properties of Iterative deepening search

- Complete
- Time
  - $O(b^d)$: $(d+1)(1) + db + (d-1)b^2 + \ldots + (1) b^d$
- Space:
  - $O(bd)$
- Optimal if step cost is 1
Construct Search Tree for 8-Queens Problem?
Construct search tree for 8-Puzzle?
(Artificial Intelligence, Russell and Norvig): The missionaries and cannibals problem is usually stated as follows. Three missionaries and three cannibals are on one side of the river, along with a boat that can hold one or two people. Find a way to get everyone to the other side, without ever leaving a group of missionaries in one place outnumbered by the cannibals in that place. This problem is famous in AI because it was the subject of the first paper that approached problem formulation from an analytical viewpoint (Amarel, 1968). (18 Points)

- Formulate the problem precisely, making only those distinctions necessary to ensure a valid solution. Draw a diagram of the complete state space.
- Solve this problem by using depth first search and breadth first search. (Check for the repeated states).
Rotating Workforce Scheduling

<table>
<thead>
<tr>
<th></th>
<th>Mo</th>
<th>Tu</th>
<th>We</th>
<th>Th</th>
<th>Fr</th>
<th>Sa</th>
<th>Su</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>D</td>
<td>D</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>A</td>
<td>A</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>H</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>I</td>
<td></td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>A</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>J</td>
<td></td>
<td>D</td>
<td>D</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>K</td>
<td></td>
<td>N</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>L</td>
<td>N</td>
<td>N</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td></td>
</tr>
</tbody>
</table>
Literature

- Artificial Intelligence: A Modern Approach (Stuart Russell and Peter Norvig)
  - Chapter 3

- How to Solve It: Modern Heuristics (Z. Michalewicz and D. B. Fogel.)
  - Chapter 3