Chapter 6

Tree Searching Strategies

Satisfiability problem

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>

Tree representation of 8 assignments.

If there are $n$ variables $x_1, x_2, \ldots, x_n$, then there are $2^n$ possible assignments.

Hamiltonian circuit problem

A graph containing a Hamiltonian circuit.

- An instance:
  - $x_1 \ldots \ldots (1)$
  - $x_1 \ldots \ldots (2)$
  - $x_1 \lor x_2 \ldots \ldots (3)$
  - $x_3 \ldots \ldots (4)$
  - $-x_2 \ldots \ldots (5)$

A partial tree to determine the satisfiability problem.

- We may not need to examine all possible assignments.
The tree representation of whether there exists a Hamiltonian circuit.

**Breadth-First Search (BFS)**
- 8-puzzle

  The breadth-first search uses a queue to holds all expanded nodes.

**Depth-First Search (DFS)**
- e.g. sum of subset problem
  \[ S = \{7, 5, 1, 2, 10\} \]
  \[ \exists S' \subseteq S \exists \text{ sum of } S' = 9 ? \]

- A stack can be used to guide the depth-first search.

**Hill Climbing**
- A variant of depth-first search

  The method selects the locally optimal node to expand.

- e.g. 8-puzzle problem
  Evaluation function \[ f(n) = d(n) + w(n) \]
  where \[ d(n) \] is the depth of node \( n \)
  \[ w(n) \] is the number of misplaced tiles in node \( n \).
Best-First Search
(Beam Search)

- Combing depth-first search and breadth-first search.
- Selecting the node with the best estimated cost among all active nodes.
- This method has a global view.

Best-First Search Scheme

Step 1: Form a one-element list consisting of the root node.
Step 2: Remove the first element from the list. Expand the selected element. If one of the descendants of the first element is a goal node, then stop; otherwise, add the descendants into the list.
Step 3: Sort the entire list by the values of some estimation function.
Step 4: If the list is empty, then failure. Otherwise, go to Step 2.
Branch-and-bound strategy

- This strategy can be used to efficiently solve optimization problems.
- e.g.

![Diagram of a multi-stage graph searching problem]

A multi-stage graph searching problem.

Personnel assignment problem

- A linearly ordered set of persons $P = \{P_1, P_2, \ldots, P_n\}$, where $P_1 < P_2 < \ldots < P_n$.
- A partially ordered set of jobs $J = \{J_1, J_2, \ldots, J_n\}$.

Suppose that $P_i$ and $P_j$ are assigned to jobs $f(P_i)$ and $f(P_j)$ respectively. If $f(P_i) \leq f(P_j)$, then $P_i \leq P_j$. Cost $C_{ij}$ is the cost of assigning $P_i$ to $J_j$. We want to find a feasible assignment with the minimum cost. i.e.

$$X_{ij} = 1 \text{ if } P_i \text{ is assigned to } J_j$$

$$X_{ij} = 0 \text{ otherwise}.$$

- Minimize $\sum_{i,j} C_{ij} X_{ij}$

Solved by branch-and-bound

- e.g. A partial ordering of jobs

$J_1 \bullet J_2 \bullet J_3 \bullet J_4 \bullet J_5 \bullet J_6$

- After topological sorting, one of the following topologically sorted sequences will be generated:

$J_1, J_2, J_3, J_4, J_5, J_6$

$J_1, J_2, J_4, J_3, J_5, J_6$

$J_2, J_1, J_3, J_4, J_5, J_6$

$J_2, J_1, J_4, J_3, J_5, J_6$

- One of feasible assignments:

$P_1 \bigtriangleup J_1, P_2 \bigtriangleup J_2, P_3 \bigtriangleup J_3, P_4 \bigtriangleup J_4$
All possible solutions can be represented by a solution tree.

**Solution Tree**

A reduced cost matrix can be obtained:
subtract a constant from each row and each column respectively such that each row and each column contains at least one zero.

**Reduced cost matrix**

A reduced cost matrix can be obtained:
subtract a constant from each row and each column respectively such that each row and each column contains at least one zero.

**Cost matrix**

Best-first search scheme:

\[
\begin{array}{cccc}
| \text{Jobs} | \text{Persons} | \text{Cost} | \\
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>29 19 17 12</td>
<td>17</td>
</tr>
<tr>
<td>2</td>
<td>32 30 26 28</td>
<td>12</td>
</tr>
<tr>
<td>3</td>
<td>0 15 4 6</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>8 0 0 5</td>
<td>(-3)</td>
</tr>
</tbody>
</table>
\end{array}
\]

Total cost subtracted: 12+26+3+10+3 = 54

This is a lower bound of our solution.
Branch-and-bound for the personnel assignment problem

Bounding of sub-solutions:

Traveling salesperson problem

Cost matrix

Reduced cost matrix

Further reduced

Total cost reduced: 84+7+1+4 = 96 (lower bound)
- The highest level of a decision tree:

```
    All solutions  Lower bound = 96
    All solutions with arc 4-6
    All solutions without arc 4-6
```

```
Lower bound = 96  Lower bound = 96 + 32 = 128
```

- A reduced cost matrix if arc (4,6) is included in the solution.

```
<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>0</td>
<td>83</td>
<td>9</td>
<td>30</td>
<td>50</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>66</td>
<td>37</td>
<td>17</td>
<td>26</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>29</td>
<td>1</td>
<td>19</td>
<td>0</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>21</td>
<td>56</td>
<td>7</td>
<td>28</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>85</td>
<td>8</td>
<td>89</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>18</td>
<td>0</td>
<td>0</td>
<td>58</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>
```

Arc (6,4) is changed to be infinity since it cannot be included in the solution.

- The reduced cost matrix for all solutions with arc 4-6

```
<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>83</td>
<td>9</td>
<td>30</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>66</td>
<td>37</td>
<td>17</td>
<td>26</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>29</td>
<td>1</td>
<td>19</td>
<td>0</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>21</td>
<td>56</td>
<td>7</td>
<td>28</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>85</td>
<td>8</td>
<td>89</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>18</td>
<td>0</td>
<td>0</td>
<td>58</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>
```

- Total cost reduced: 96 + 3 = 99 (new lower bound)

- A branch-and-bound solution of a traveling salesperson problem.
The 0/1 knapsack problem

- Positive integer \( P_1, P_2, \ldots, P_n \) (profit)
- \( W_1, W_2, \ldots, W_n \) (weight)
- \( M \) (capacity)

maximize \( \sum_{i=1}^{n} P_i X_i \)
subject to \( \sum_{i=1}^{n} W_i X_i \leq M \) \( X_i = 0 \) or 1, \( i = 1, \ldots, n \).

The problem is modified:
minimize \(- \sum_{i=1}^{n} P_i X_i \)

- e.g. \( n = 6, M = 34 \)

<table>
<thead>
<tr>
<th>( i )</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>( P_i )</td>
<td>6</td>
<td>10</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>( W_i )</td>
<td>10</td>
<td>19</td>
<td>8</td>
<td>10</td>
<td>12</td>
<td>8</td>
</tr>
</tbody>
</table>

\((P_i/W_i \geq P_{i+1}/W_{i+1})\)

- A feasible solution: \( X_1 = 1, X_2 = 1, X_3 = 0, X_4 = 0, X_5 = 0, X_6 = 0 \)
- \(- (P_1 + P_2) = -16 \) (upper bound)
Any solution higher than -16 cannot be optimal.

Relax the restriction

- Relax our restriction from \( X_i = 0 \) or 1 to \( 0 \leq X_i \leq 1 \) (knapsack problem)

Let \(- \sum_{i=1}^{n} P_i X_i \) be an optimal solution for 0/1 knapsack problem and \(- \sum_{i=1}^{n} P_i X'_i \) be an optimal solution for knapsack problem. Let \( Y = - \sum_{i=1}^{n} P_i X_i \), \( Y' = - \sum_{i=1}^{n} P_i X'_i \).

\( \Rightarrow Y' \leq Y \)

Upper bound and lower bound

- We can use the greedy method to find an optimal solution for knapsack problem:

\( X_1 = 1, X_2 = 1, X_3 = 5/8, X_4 = 0, X_5 = 0, X_6 = 0 \)

\(- (P_1 + P_2 + 5/8 P_3) = -18.5 \) (lower bound)

-18 is our lower bound. (only consider integers)

\( \Rightarrow -18 \leq \text{optimal solution} \leq -16 \)
optimal solution: \( X_1 = 1, X_2 = 0, X_3 = 0, X_4 = 1, X_5 = 1, X_6 = 0 \)

\(- (P_1 + P_4 + P_6) = -17 \)
The A* algorithm

- Used to solve optimization problems.
- Using the best-first strategy.
- If a feasible solution (goal node) is obtained, then it is optimal and we can stop.
- Cost function of node $n$ : $f(n)$
  
  $f(n) = g(n) + h(n)$
  
  $g(n)$: cost from root to node $n$.
  
  $h(n)$: estimated cost from node $n$ to a goal node.
  
  $h^*(n)$: “real” cost from node $n$ to a goal node.

- If we guarantee $h(n) \leq h^*(n)$, then
  
  $f(n) = g(n) + h(n) \leq g(n) + h^*(n) = f^*(n)$

Example for A* algorithm

- A Graph to Illustrate A* Algorithm.

  Stops iff the selected node is also a goal node.

Step 1:

<table>
<thead>
<tr>
<th>$g(A)$</th>
<th>$h(A)$</th>
<th>$f(A)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>min{2,3}</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>min{2}</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>min{2,2}</td>
<td>5</td>
</tr>
</tbody>
</table>
Step 2: Expand node A.

- $g(D) = 2 + 2 = 4$
- $h(D) = \min\{3, 1\} = 1$
- $f(D) = 4 + 1 = 5$
- $g(E) = 2 + 3 = 5$
- $h(E) = \min\{2, 2\} = 2$
- $f(E) = 5 + 2 = 7$

Step 3: Expand node C.

- $g(F) = 3 + 2 = 5$
- $h(F) = \min\{3, 1\} = 1$
- $f(F) = 5 + 1 = 6$
- $g(G) = 3 + 2 = 5$
- $h(G) = \min\{5\} = 5$
- $f(G) = 5 + 5 = 10$

Step 4: Expand node D.

- $g(H) = 2 + 2 + 1 = 5$
- $h(H) = \min\{5\} = 5$
- $f(H) = 5 + 5 = 10$
- $g(I) = 2 + 2 + 3 = 7$
- $h(I) = 0$
- $f(I) = 7 + 0 = 7$

Step 5: Expand node B.

- $g(J) = 4 + 2 = 6$
- $h(J) = \min\{5\} = 5$
- $f(J) = 6 + 5 = 11$
Step 6: Expand node $F$.

$$f(n) \leq f^*(n)$$

$g(K) = 3 + 2 + 1 = 6$ \quad $h(K) = \min\{5\} = 5$ \quad $f(K) = 6 + 5 = 11$

$g(L) = 3 + 2 + 3 = 8$ \quad $h(L) = 0$ \quad $f(L) = 8 + 0 = 8$

Node $I$ is a goal node. Thus, the final solution has been obtained.

The channel routing problem

- A Channel Specification

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>0</td>
<td>7</td>
<td>0</td>
<td>3</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>0</td>
</tr>
</tbody>
</table>


A feasible routing

- 7 tracks are needed.
An optimal routing

- 4 tracks are needed.
- This problem is NP-complete.

A* algorithm for the channel routing problem

- Vertical constraint graph:
  - e.g. net 3 must be wired before net 4
  - Maximum cliques in HCG: \{1,8\}, \{1,3,7\}, \{5,7\}. Each maximum clique can be assigned to a track.

- \( f(n) = g(n) + h(n) \),
  - \( g(n) \): the level of the tree
  - \( h(n) \): maximal local density

A partial solution tree for the channel routing problem by using A* algorithm.