Artificial Intelligence Lecture Notes by Dr. Belal Al-Khateeb

Week No.: 3

Lecture Title: Systematic Search: Basic Graph Concepts; State Space Representation of Problems.

*** <u>Search Techniques</u>**

There are two types of searches, these are:

1- Blind search
   - Depth First Search.
   - Breadth First Search.
   - Hybrid Search.

2- Heuristic search
   - Hill Climbing.
   - Best First Search.
   - A algorithm.
   - A* algorithm.

* **<u>Structures and Strategies for State Space Search</u>**

To successfully design and implement search algorithms, a programmer must be able to analyze and predict their behavior. Questions that need to be answered include:

- Is the problem solver guaranteed to find a solution?
- Will the problem solver always terminate, or can it become caught in an infinite loop?
- When a solution is found, it is guaranteed to be optimal?
- What is the complexity of the search process in terms of time usage? Space usage?
- How can the interpreter most effectively reduce search complexity?

- How can an interpreter be designed to most effectively utilize a representation language?

The theory of **state space search** is the primary tool for answering these questions. By representing a problem as a **state space** graph, the graph theory can be used to analyze the structure and complexity of both the problem and the procedures used to solve it.

If a graph is used, the problem of cycles will occur so the best structure to represent a problem is a **tree structure** (there is a unique path between every two nodes)**,** which can be defined as a tree with no cycles. It is important to distinguish between problems whose state space is a tree and those that may contain loops (graph). General graph search algorithms must detect and eliminate loops from potential solution paths, while tree searches may gain efficiency by eliminating this test and its overhead. So convert graph representation to a tree representation. To convert a graph to a tree, remove any nodes that may cause a cycle.

<u>Example: convert a graph to a tree</u>