|  |  |  |
| --- | --- | --- |
| Logo_MEC  **LECTURE HANDOUTS**  **III/V**  **IT** | **MUTHAYAMMAL ENGINEERING COLLEGE**  **(An Autonomous Institution)**  **(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)**  **Rasipuram - 637 408, Namakkal Dist., Tamil Nadu** | **L1** |

**Course Name with Code : 19ITC17 Artificial Intelligence**

**Course Faculty : P.Bhuvaneshwari**

**Unit : I - introduction to al and production systems**

**Date of Lecture:**

|  |
| --- |
| **Topic of Lecture:** Constraint Satisfaction problem- Related algorithms- Measure of performance and analysis of algorithms |
| **Introduction:**   * Constraint satisfaction is a technique where a problem is solved when its values satisfy certain constraints or rules of the problem. * Performance metrics to analyze the optimum problem solving strategy |
| **Prerequisite knowledge for Complete understanding and learning of Topic:**   * Search Strategies * Basic Mathematics * Time Complexity * Space Complexity |
| **Detailed content of the Lecture:**  **Artificial Intelligence (AI) is a branch of Science which deals with helping machines to find solutions to complex problems in a more human-like fashion.\**  **Branches of Artificial intelligence:**   * **Games playing**: programming computers to play games such as chess and checkers. * **Expert systems**: programming computers to make decisions in real-life situations (forexample, some expert systems help doctors diagnose diseases based on symptoms). * **Natural Language Processing**: programming computers to understand natural humanlanguages. * **Neural Networks**: Systems that simulate intelligence by attempting to reproduce thetypes of physical connections that occur in human brains. * **Robotics**: programming computers to see and hear and react to other sensory stimulicurrently, no computers exhibit full artificial intelligence (that is, are able to simulate human behavior).   **Applications of AI:**   * **Game playing**   There is some AI in them, but they play well against people mainly through brute force computation--looking at hundreds of thousands of positions. To beat a world champion by brute force and known reliable heuristics requires being able to look at 200 million positions per second.   * **Speech recognition**   In the 1990s, computer speech recognition reached a practical level for limited purposes. Thus United Airlines has replaced its keyboard tree for flight information by a system using speech recognition of flight numbers and city names. It is quite convenient.   * **Understanding natural language**   Just getting a sequence of words into a computer is not enough. Parsing sentences is not enough either. The computer has to be provided with an understanding of the domain the text is about, and this is presently possible only for very limited domains.   * **Computer vision**   The world is composed of three-dimensional objects, but the inputs to the human eye and computers' TV cameras are two dimensional. Some useful programs can work solely in two dimensions, but full computer vision requires partial three-dimensional information that is not just a set of two-dimensional views. At present there are only limited ways of representing three-dimensional information directly, and they are not as good as what humans evidently use.   * **Expert systems**   A ``knowledge engineer'' interviews experts in a certain domain and tries to embody their knowledge in a computer program for carrying out some task. How well this works depends on whether the intellectual mechanisms required for the task are within the present state of AI. One of the first expert systems was MYCIN in 1974, which diagnosed bacterial infections of the blood and suggested treatments. It did better than medical students or practicing doctors, provided its limitations were observed.   * **Intelligent Robots** − Robots are able to perform the tasks given by a human. They have sensorsto detect physical data from the real world such as light, heat, temperature, movement, sound, bump, and pressure. They have efficient processors, multiple sensors and huge memory, to exhibit intelligence. In addition, they are capable of learning from their mistakes and they can adapt to the new environment. * **Heuristic classification**   One of the most feasible kinds of expert system given the present knowledge of AI is to put some information in one of a fixed set of categories using several sources of information. An example is advising whether to accept a proposed credit card purchase. Information is available about the owner of the credit card, his record of payment and also about the item he is buying and about the establishment from which he is buying it (e.g., about whether there have been previous credit card frauds at this establishment).  **HISTORY OF ARTIFICIAL INTELLIGENCE:**   * **The gestation of artificial intelligence (1943-1955)**   There were a number of early examples of work that can be characterized as AI, but it was Alan Turing who first articulated a complete vision of AI in his 1950 article "Computing Machinery and Intelligence." Therein, he introduced the Turing test, machine learning, genetic algorithms, and reinforcement learning.   * **The birth of artificial intelligence (1956)**   McCarthy convinced Minsky, Claude Shannon, and Nathaniel Rochester to help him bring together U.S. researchers interested in automata theory, neural nets, and the study of  intelligence. They organized a two-month workshop at Dartmouth in the summer of 1956.  Perhaps the longest-lasting thing to come out of the workshop was an agreement to adopt  McCarthy's new name for the field: artificial intelligence.   * **Early enthusiasm, great expectations (1952-1969)**   The early years of Artificial Intelligence were full of successes-in a limited way. General  Problem Solver (GPS) was a computer program created in 1957 by Herbert Simon and Allen  Newell to build a universal problem solver machine. The order in which the program considered sub goals and possible actions was similar to that in which humans approached the same problems. Thus, GPS was probably the first program to embody the "thinking humanly" approach.   * IBM, Nathaniel Rochester and his colleagues produced some of the first AI pro-grams. * Herbert Gelernter (1959) constructed the Geometry Theorem Prover, which was able to prove   theorems that many students of mathematics would find quite tricky.   * Lisp was invented by John McCarthy in 1958 while he was at the Massachusetts Institute   of Technology (MIT). In 1963, McCarthy started the AI lab at Stanford.   * Knowledge-based systems: The key to power? (1969-1979)   Dendral was an influential pioneer project in artificial intelligence (AI) of the 1960s, and the computer software expert system that it produced. Its primary aim was to help organic chemists in identifying unknown organic molecules, by analyzing their mass spectra and using  knowledge of chemistry. It was done at Stanford University by Edward Feigenbaum, Bruce  Buchanan, Joshua Lederberg, and Carl Djerassi.   * AI becomes an industry (1980-present)   In 1981, the Japanese announced the "Fifth Generation" project, a 10-year plan to build  intelligent computers running Prolog. Overall, the AI industry boomed from a few million  dollars in 1980 to billions of dollars in 1988.   * AI becomes a science (1987-present)   In recent years, approaches based on hidden Markov models (HMMs) have come to dominate the area.   * Speech technology and the related field of handwritten character recognition are already making the transition to widespread industrial and consumer applications. * The Bayesian network formalism was invented to allow efficient representation of, and rigorous reasoning with, uncertain knowledge. * The emergence of intelligent agents (1995-present)   One of the most important environments for intelligent agents is the Internet.  **Components of Artificial Intelligence:**   * **Reasoning, problem-solving:** Researchers had developed machines with algorithms that enable machines to solve puzzles or quiz similar to humans. AI can also deal with uncertain or incomplete information through advanced algorithms. * **Knowledge Representation:**It is the representation of all the knowledge which is stored by an agent to make an expert system. Knowledge can be a set of objects, relations, concepts, or properties. * **Planning:**Intelligent agents should be able to set goals and make plans to achieve those goals. They should be able to visualize the future and make predictions about their actions taken for achieving the goal. * **Learning:** It is the study of the computer algorithms which improve automatically through experiences. This concept is known as Machine Learning. * **Natural Language Processing:** This processing enables a machine to read and understand human language by processing the human language into machine language. * **Perception:** An ability of the machine to use input from sensors, microphones, wireless signals, etc. for understanding different aspects of the world.   **artificial-intelligence-notes-unit-1-22-638** |
| **Video Content / Details of website for further learning (if any):**   1. [www.artint.info/html/ArtInt.htm](http://www.artint.info/html/ArtInt.htm)l 2. [www.aima.cs.berkeley.ed](http://www.aima.cs.berkeley.ed/)u 3. [www-formal.stanford.edu/jmc/whatisai](http://www-formal.stanford.edu/jmc/whatisai/)/ 4. [www.nptel.ac.in/courses/10610612](http://www.nptel.ac.in/courses/10610612)6 |
| **Important Books/Journals for further learning including the page nos.:**  1. Kevin Night and Elaine Rich, Nair B., “Artificial Intelligence (SIE)”, McGraw Hill- 2008.  Page No. : 1- 29 |

**Course Faculty**

**Verified by HOD**

|  |  |  |
| --- | --- | --- |
| Logo_MEC  **LECTURE HANDOUTS**  **III/ V**  **IT** | **MUTHAYAMMAL ENGINEERING COLLEGE**  **(An Autonomous Institution)**  **(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)**  **Rasipuram - 637 408, Namakkal Dist., Tamil Nadu** | **L2**  **2** |

**Course Name with Code : 19ITC17 Artificial Intelligence**

**Course Faculty : P.Bhuvaneshwari**

**Unit : I - introduction to al and production systems**

**Date of Lecture:**

|  |
| --- |
| **Topic of Lecture:** Agents, Environment**,** Problem formulation, Problem Definition and Characteristics |
| **Introduction:**  **Problem Solving Approach:**   * Problem Formulation * Defining the Problem * Alternatives to solve the Problem * Find a optimized solution |
| **Prerequisite knowledge for Complete understanding and learning of Topic:**   * Understand the Problem * Define the Problem * Alternatives to solve the problem by various method * Find a optimized solution |
| **Detailed content of the Lecture:**  **Agents and Environment:**  Diagrammatic-Representation-of-an-Agent  An agent can be viewed as anything that perceives its environment through sensors and acts upon that environment through actuators.  For example, human being perceives their surroundings through their sensory organs known as sensors and take actions using their hands, legs, etc., known as actuators. Intelligent Agent An intelligent agent is a goal-directed agent. It perceives its environment through its sensors using the observations and built-in knowledge, acts upon the environment through its actuators. Rational Agent A rational agent is an agent which takes the right action for every perception. By doing so, it maximizes the performance measure, which makes an agent be the most successful.  **Omniscient Agent**  An omniscient agent is an agent which knows the actual outcome of its action in advance. However, such agents are impossible in the real world.  Task environment:  A task environment is a problem to which a rational agent is designed as a solution. Rationality of an agent It is expected from an intelligent agent to act in a way that maximizes its performance measure. Therefore, the rationality of an agent depends on four things:   * The performance measure which defines the criterion of success. * The agent’s built-in knowledge about the environment. * The actions that the agent can perform. * The agent’s percept sequence until now.  Properties/Classification of Task EnvironmentFully Observable vs. Partially Observable: When an agent’s sensors allow access to complete state of the environment at each point of time, then the task environment is fully observable, whereas, if the agent does not have complete and relevant information of the environment, then the task environment is partially observable.  **Example:** In the Checker Game, the agent observes the environment completely while in Poker Game, the agent partially observes the environment because it cannot see the cards of the other agent.  **Note:** Fully Observable task environments are convenient as there is no need to maintain the internal state to keep track of the world. Single-agent vs. Multiagent When a single agent works to achieve a goal, it is known as Single-agent, whereas when two or more agents work together to achieve a goal, they are known as Multiagents.  **Example:** Playing a crossword puzzle – single agent  Playing chess –multiagent (requires two agents) Deterministic vs. Stochastic If the agent’s current state and action completely determine the next state of the environment, then the environment is deterministic whereas if the next state cannot be determined from the current state and action, then the environment is Stochastic.  **Example:** Image analysis – Deterministic  Taxi driving – Stochastic (cannot determine the traffic behavior)  **Note:** If the environment is partially observable, it may appear as Stochastic Episodic vs. Sequential If the agent’s episodes are divided into atomic episodes and the next episode does not depend on the previous state actions, then the environment is episodic, whereas, if current actions may affect the future decision, such environment is sequential.  **Example:**  Part-picking robot – Episodic  Chess playing – Sequential Static vs. Dynamic If the environment changes with time, such an environment is dynamic; otherwise, the environment is static.  **Example:**Crosswords Puzzles have a static environment while the Physical world has a dynamic environment. Discrete vs. Continuous If an agent has the finite number of actions and states, then the environment is discrete otherwise continuous.  **Example:** In Checkers game, there is a finite number of moves – Discrete  A truck can have infinite moves while reaching its destination –           Continuous. Known vs. Unknown In a known environment, the agents know the outcomes of its actions, but in an unknown environment, the agent needs to learn from the environment in order to make good decisions.  **Example:** A tennis player knows the rules and outcomes of its actions while a player needs to learn the rules of a new video game.  **Note:**A known environment is partially observable, but an unknown environment is fully observable. Types of Agent Programs:  * **Simple reflex agents:** It is the simplest agent which acts according to the current percept only, pays no attention to the rest of the percept history. The agent function of this type relies on the **condition-action rule** – “If condition, then action.” It makes correct decisions only if the environment is fully observable. These agents cannot ignore infinite loop when the environment is partially observable but can escape from infinite loops if the agents randomize its actions.   **Example:** iDraw, a drawing robot which converts the typed characters into  writing without storing the past data.  **Note:** Simple reflex agents do not maintain the internal state and do not depend on the percept theory.   * **Model-based agent:** These type of agents can handle partially observable environments by maintaining some internal states. The internal state depends on the percept history, which reflects at least some of the unobserved aspects of the current state. Therefore, as time passes, the internal state needs to be updated which requires two types of knowledge or information to be encoded in an agent program i.e., the evolution of the world on its own and the effects of the agent’s actions.   **Example:** When a person walks in a lane, he maps the pathway in his mind.   * **Goal-based agents:** It is not sufficient to have the current state information unless the goal is not decided. Therefore, a goal-based agent selects a way among multiple possibilities that helps it to reach its goal.   **Note:** With the help of searching and planning (subfields of AI), it becomes easy for the Goal-based agent to reach its destination.   * **Utility-based agents:** These types of agents are concerned about the performance measure. The agent selects those actions which maximize the performance measure and devote towards the goal.   **Example:** The main goal of chess playing is to ‘check-and-mate’ the king, but the player completes several small goals previously.  **Problem Solving:**  There are four things that are to be followed in order to solve a problem:   1. *Define the problem*. A precise definition that must include precise specifications of whatthe initial situation(s) will be as well as what final situations constitute acceptable solutions to the problem. 2. *Problem must be analyzed*. Some of the important features land up having an immenseimpact on the appropriateness of various possible techniques for solving the problem. 3. *Isolate and represent the task knowledge* that is necessary to solve the problem. 4. Amongst the available ones *choose the best problem-solving technique*(s) and apply the same to the particular problem.   **State Space Representation of a problem:**  In this section we examine the concept of a state space and the different searches that can be used to explore the search space in order to find a solution. Before an AI problem can be solved it must be represented as a state space. The state space is then searched to find a solution to the problem. A state space essentially consists of a set of nodes representing each state of the problem, arcs between nodes representing the legal moves from one state to another, an initial state and a goal state. Each state space takes the form of a tree or a graph. Factors that determine which search algorithm or technique will be used include the type of the problem and the how the problem can be represented. |

|  |
| --- |
| **Video Content / Details of website for further learning (if any):**   1. [www.artint.info/html/ArtInt.htm](http://www.artint.info/html/ArtInt.htm)l 2. [www.aima.cs.berkeley.ed](http://www.aima.cs.berkeley.ed/)u 3. [www-formal.stanford.edu/jmc/whatisai](http://www-formal.stanford.edu/jmc/whatisai/)/ 4. [www.nptel.ac.in/courses/10610612](http://www.nptel.ac.in/courses/10610612)6 |
| **Important Books/Journals for further learning including the page nos.:**  1. Kevin Night and Elaine Rich, Nair B., “Artificial Intelligence (SIE)”, McGraw Hill- 2008.  Page No. : 29-35 |

**Course Faculty**

**Verified by HOD**

|  |  |  |
| --- | --- | --- |
| Logo_MEC  **LECTURE HANDOUTS**  **III/V**  **IT** | **MUTHAYAMMAL ENGINEERING COLLEGE**  **(An Autonomous Institution)**  **(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)**  **Rasipuram - 637 408, Namakkal Dist., Tamil Nadu** | **L3** |

**Course Name with Code : 19ITC17 Artificial Intelligence**

**Course Faculty : P.Bhuvaneshwari**

**Unit : I - introduction to al and production systems**

**Date of Lecture:**

|  |
| --- |
| **Topic of Lecture:** Production Systems  **Introduction:**   * Production systems provide search structures that form the core of many intelligent processes. * it is useful to structure AI programs in a way that facilitates describing and performing the search process |
| **Prerequisite knowledge for Complete understanding and learning of Topic:**   * Problem Formulation * Problem Representation * Problem Solving Techniques |
| **Detailed content of the Lecture:**  Production systems provide search structures that form the core of many intelligent processes. Hence it is useful to structure AI programs in a way that facilitates describing and performing the search process. Do not be confused by other uses of the word *production,* such as to describe what is done in factories. A *production system* consists of:   1. **A set of rules**, each consisting of a left side and a right hand side. Left hand side or patterndetermines the applicability of the rule and a right side describes the operation to be performed if the rule is applied. 2. **One or more knowledge/databases** that contain whatever information is appropriate for theparticular task. Some parts of the database may be permanent, while other parts of it may pertain only to the solution of the current problem. The information in these databases may be structured in any appropriate way. 3. **A control strategy** that specifies the order in which the rules will be compared to thedatabase and a way of resolving the conflicts that arise when several rules match at once. 4. **A rule applier**.   Image result for production system in ai  Image result for production system in ai  The production rules are also known as condition – action, antecedent – consequent, pattern – action, situation – response, feedback – result pairs.   For example,   If (you have an exam tomorrow)   THEN (study the whole night)   The production system can be classified as monotonic, non-monotonic, partially commutative and commutative.  **Features of Production System:**  Some of the main features of production system are:  **Expressiveness and intuitiveness**: In real world, many times situation comes like “i f this happen-you will do that”, “if this is so-then this should happ en” and many more. The production rules essentially tell us what to do in a given situation.   1.     **Simplicity:**The structure of each sentence in a production system is unique and uniform as they use“IF-THEN” structure. This structure provides simpli city in knowledge representation. This feature of production system improves the readability of production rules.   2.     **Modularity:**This means production rule code the knowledge available in discrete pieces.Information can be treated as a collection of independent facts which may be added or deleted from the system with essentially no deletetious side effects.   3.     **Modifiability:**This means the facility of modifying rules. It allows the development of productionrules in a skeletal form first and then it is accurate to suit a specific application.  4.     **Knowledge intensive:**The knowledge base of production system stores pure knowledge. This partdoes not contain any type of control or programming information. Each production rule is normally written as an English sentence; the problem of semantics is solved by the very structure of the representation.    **Disadvantages of production system:**   1.     **Opacity:**This problem is generated by the combination ofproduction rules. The opacity is generatedbecause of less prioritization of rules. More priority to a rule has the less opacity.  2.     **Inefficiency:**During execution of a program several rules may active. A well devised control strategyreduces this problem. As the rules of the production system are large in number and they are hardly written in hierarchical manner, it requires some forms of complex search through all the production rules for each cycle of control program.   3.     **Absence of learning:**Rule based production systems do not store the result of the problem for futureuse. Hence, it does not exhibit any type of learning capabilities. So for each time for a particular problem, some new solutions may come.   4.     **Conflict resolution:**The rules in a production system should not have any type of conflictoperations. When a new rule is added to a database, it should ensure that it does not have any conflicts with the existing rules. |
| **Video Content / Details of website for further learning (if any):**   1. [www.artint.info/html/ArtInt.htm](http://www.artint.info/html/ArtInt.htm)l 2. [www.aima.cs.berkeley.ed](http://www.aima.cs.berkeley.ed/)u 3. [www-formal.stanford.edu/jmc/whatisai](http://www-formal.stanford.edu/jmc/whatisai/)/ 4. [www.nptel.ac.in/courses/10610612](http://www.nptel.ac.in/courses/10610612)6 |
| **Important Books/Journals for further learning including the page nos.:**  1. Kevin Night and Elaine Rich, Nair B., “Artificial Intelligence (SIE)”, McGraw Hill- 2008.  Page No. : 36- 44 |
|  |

**Course Faculty**

**Verified by HOD**

|  |  |  |
| --- | --- | --- |
| Logo_MEC  **LECTURE HANDOUTS**  **III/ V**  **IT** | **MUTHAYAMMAL ENGINEERING COLLEGE**  **(An Autonomous Institution)**  **(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)**  **Rasipuram - 637 408, Namakkal Dist., Tamil Nadu** | **L4** |

**Course Name with Code : 19ITC17 Artificial Intelligence**

**Course Faculty : P.Bhuvaneshwari**

**Unit : I - introduction to al and production systems**

**Date of Lecture:**

|  |
| --- |
| **Topic of Lecture:** Control and Search strategies |
| **Introduction:**  Control strategy that determines the order in which the rules are applied to the database, and provides a way of resolving any conflicts that can arise when several rules match at once. |
| **Prerequisite knowledge for Complete understanding and learning of Topic:**  Components of AI  Production System- Basics |
| **Detailed content of the Lecture:**   1. **Problem Characteristics** 2. Is the problem decomposable? 3. Can solution steps be ignored or undone? 4. Is the Universal Predictable? 5. Is good solution absolute or relative ? 6. The knowledge base consistent ? 7. What is the role of Knowledge? 8. Does the task requires interaction with the person.   **b) Control Strategies**  One of the searches that are commonly used to search a state space is the depth first search. The depth first search searches to the lowest depth or ply of the tree.  **Depth First Search Algorithm**  def dfs (in Start, out State)  open = [Start];  closed = [];  State = failure;  while (open <> []) AND (State <> success)  begin  remove the leftmost state from  open  , call it X;  if X is the goal, then  State = success  else begin  generate children of X;  put X on closed  eliminate the children of X on  open  or  closed  put remaining children on left end of  open  end else  endwhile  return State;  enddef  Breadth First Search  The breadth first search visits nodes in a left to right manner at each level of the tree. Each node is not visited more than once.  **Breadth First Search Algorithm**  def bfs (in Start, out State)  open = [Start];  closed = [];  State = failure;  while (open <> []) AND (State <> success)  begin  remove the leftmost state from  open  , call it X;  if X is the goal, then  State = success  else begin  generate children of X;  put X on closed  eliminate the children of X on  open  or  closed  put remaining children on right end of  open  end else  endwhile  return State;  enddef  **Differences Between Depth First and Breadth First**  Breadth first is guaranteed to find the shortestpath from the start to the goal. DFS may get “lost” in search and hence a depth-bound may have to be imposed on a depth first search.  BFS has a bad branching factor which can lead to combinatorialexplosion. The solution path found by the DFS may be long and not optimal. DFS  more efficient for search spaces with many branches, i.e. the branching factor is high.  The following factors need to be taken into consideration when deciding which search to use:   * The importance of finding the shortest path to the goal. * The branching of the state space. * The available time and resources. * The average length of paths to a goal node. * All solutions or only the first solution.   **b) Heuristic Search**  DFS and BFS may require too much memory to generate an entire state space - in these cases heuristic search is used.  Heuristics help us to reduce the size of thesearch space. An evaluation function is applied to each goal to assess howpromising it is in leading to the goal  Examples of heuristic searches:   * Best first search * A\* algorithm * Hill climbing   Heuristic searches incorporate the use of domain-specific knowledge in the process of choosing which node to visit next in the search process. Search methods that include the use of domain knowledge in the form of heuristics are described as “weak” search methods. The knowledge used is “weak” in that it usually helps but does not always help to find a solution.  **Calculating heuristics**  Heuristics are rules of thumb that may find a solution but are not guaranteed to. Heuristic functions have also been defined as evaluation functions that estimate the cost from a node to the goal node. The incorporation of domain knowledge into this arch process by means of heuristics ismeant to speed up the search process. |
| **Video Content / Details of website for further learning (if any):**   1. [www.artint.info/html/ArtInt.htm](http://www.artint.info/html/ArtInt.htm)l 2. [www.aima.cs.berkeley.ed](http://www.aima.cs.berkeley.ed/)u 3. [www-formal.stanford.edu/jmc/whatisai](http://www-formal.stanford.edu/jmc/whatisai/)/ 4. [www.nptel.ac.in/courses/10610612](http://www.nptel.ac.in/courses/10610612)6 |
| **Important Books/Journals for further learning including the page nos.:**  1. Kevin Night and Elaine Rich, Nair B., “Artificial Intelligence (SIE)”, McGraw Hill- 2008.  Page No. : 55- 77 |

**Course Faculty**

**Verified by HOD**

|  |  |  |
| --- | --- | --- |
| Logo_MEC  **LECTURE HANDOUTS**  **III/ V**  **IT** | **MUTHAYAMMAL ENGINEERING COLLEGE**  **(An Autonomous Institution)**  **(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)**  **Rasipuram - 637 408, Namakkal Dist., Tamil Nadu** | **L5** |

**Course Name with Code : 19ITC17 Artificial Intelligence**

**Course Faculty : P.Bhuvaneshwari**

**Unit : I - introduction to al and production systems**

**Date of Lecture:**

|  |
| --- |
| **Topic of Lecture:** Production system characteristics- Specialized production systems |
| **Introduction:**   * The production system is a model of computation that can be applied to implement search algorithms and model human problem solving. * Problem solving knowledge can be packed up in the form of little quanta called productions. A production is a rule consisting of a situation recognition part and an action part. * A production is a situation-action pair in which the left side is a list of things to watch for and the right side is a list of things to do so. |
| **Prerequisite knowledge for Complete understanding and learning of Topic:**   * Knowledge about Production system and Rule Generation.   **Production System and its Components:** |
| A production system consists of following components.   1. *A set of production rules*, which are of the form A V B, left hand side constituent that represent the current problem state and a right hand side that represent an output state. A rule is applicable if its left hand side matches with the current problem state. 2. *A database*, which contains all the appropriate information for the particular task. Somepart of the database may be permanent while some part of this may pertain only to the solution of the current problem. 3. *A control strategy* that specifies order in which the rules will be compared to the databaseof rules and a way of resolving the conflicts that arise when several rules match simultaneously. 4. *A rule applier*, which checks the capability of rule by matching the content state with theleft hand   side of the rule and finds the appropriate rule from database of rules.  **Production System Characteristics:**  Production systems are important in building intelligent matches which can provide us a good set of production rules, for solving problems.  There are four types of production system characteristics, namely   1. Monotonic production system 2. Non-monotonic production system 3. Commutative law based production system, and lastly, 4. Partially commutative law based production system.  * Monotonic Production System (MPS): The Monotonic production system (MPS) is a system in which the application of a rule never prevents later application of another rule that could also have been applied at the time that the first rule was selected. * Non-monotonic Production (NMPS): The non-monotonic production system is a system in which the application of a rule prevents the later application of the another rule which may not have been applied at the time that the first rule was selected, i.e. it is a system in which the above rule is not true, i.e. the monotonic production system rule not true. * Commutative Law Based Production System (CBPS): Commutative law based production systems is a system in which it satisfies both monotonic & partially commutative. * Partially Commutative Production System (PCPS): The partially commutative production system is a system with the property that if the application of those rules that is allowable & also transforms from state x to state ‘y’. |
| **Video Content / Details of website for further learning (if any):**   1. [www.artint.info/html/ArtInt.htm](http://www.artint.info/html/ArtInt.htm)l 2. [www.aima.cs.berkeley.ed](http://www.aima.cs.berkeley.ed/)u 3. [www-formal.stanford.edu/jmc/whatisai](http://www-formal.stanford.edu/jmc/whatisai/)/ 4. [www.nptel.ac.in/courses/10610612](http://www.nptel.ac.in/courses/10610612)6 |
| **Important Books/Journals for further learning including the page nos.:**  1. Kevin Night and Elaine Rich, Nair B., “Artificial Intelligence (SIE)”, McGraw Hill- 2008.  Page No. : 65- 73 |

**Course Faculty**

**Verified by HOD**

|  |  |  |
| --- | --- | --- |
| Logo_MEC  **LECTURE HANDOUTS**  **III/ V**  **IT** | **MUTHAYAMMAL ENGINEERING COLLEGE**  **(An Autonomous Institution)**  **(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)**  **Rasipuram - 637 408, Namakkal Dist., Tamil Nadu** | **L6** |

**Course Name with Code : 19ITC17 Artificial Intelligence**

**Course Faculty : P.Bhuvaneshwari**

**Unit : I - introduction to al and production systems**

**Date of Lecture:**

|  |
| --- |
| **Topic of Lecture:** Problem graphs with an example (Constraint Satisfaction Problem- Missionaries and Cannibals) |
| **Introduction:**   * Various strategies to solve the problem * Most strategies may lead to solution * Choose strategy which produces results at low cost and time. |
| **Prerequisite knowledge for Complete understanding and learning of Topic:**    Graph Theory  Discrete Mathematics  Problem formulation and representation Techniques.  **Detailed content of the Lecture:**  Problem solving methods- Problem graphs  Problem Representation: |
| * States, operators and searching * representing state spaces using graphs * finding a path from A to B in a graph * Example: missionaries & cannibals: representing states & operators * methods of search: depth-first, breadth-first   **Missionaries and Cannibals**   * There are three missionaries and three cannibals on the left bank of a river. * They wish to cross over to the right bank using a boat that can only carry two at a time. * The number of cannibals on either bank must never exceed the number of missionaries on the same bank, otherwise the missionaries will become the cannibals' dinner! * Plan a sequence of crossings that will take everyone safely accross.   This kind of problem is often solved by a graph search method. We represent the problem as a set of  states  which are snapshots of the world and  operators  which transform one state into another  States can be mapped to nodes of a graph and operators are the edges of the graph. Before studying the missionaries and cannibals problem, we look at a simple graph search algorithm in Prolog. the missionaries and cannibals program will have the same basic structure.  **Graph Representation**  mandc1  A graph may be represented by a set of edge predicates and a list of vertices.  edge(1, 5). edge(1, 7).  edge(2, 1). edge(2, 7).  edge(3, 1). edge(3, 6).  edge(4, 3). edge(4, 5).  edge(5, 8).  edge(6, 4). edge(6, 5).  edge(7, 5).  edge(8, 6). edge(8, 7).  vertices([1, 2, 3, 4, 5, 6, 7, 8]).  **Finding a path**   * Write a program to find path from one node to another. * Must avoid cycles (i.e. going around in circle). * A template for the clause is:   path(Start, Finish, Visited, Path).  Start  is the name of the starting node  Finish  is the name of the finishing node  Visited  is the list of nodes already visited.  Path  is the list of nodes on the path, including **Start** and **Finish**.  **The Path Program**   * The search for a path terminates when we have nowhere to go.   path(Node, Node, \_, [Node]).   * A path from Start to Finish starts with a node, X, connected to Start followed by a path from X to Finish.   path(Start, Finish, Visited, [Start | Path]) :-  edge(Start, X),  not(member(X, Visited)),  path(X, Finish, [X | Visited], Path). Representing the state Now we return to the problem of representing the missionaries anc cannibals problem:   * A state is one "snapshot" in time. * For this problem, the only information we need to fully characterise the state is:   + the number of missionaries on the left bank,   + the number of cannibals on the left bank,   + the side the boat is on.   All other information can be deduced from these three items.   * In Prolog, the state can be represented by a 3-arity term,   state(Missionaries, Cannibals, Side) Representing the Solution  * The solution consists of a list of moves, e.g.   [move(1, 1, right), move(2, 0, left)]   * We take this to mean that 1 missionary and 1 cannibal moved to the right bank, then 2 missinaries moved to the left bank. * Like the graph search problem, we must avoid returning to a state we have visited before. * The visited list will have the form:   [MostRecent\_State | ListOfPreviousStates] Overview of Solution  * We follow a simple graph search procedure:   + Start from an initial state   + Find a neighbouring state   + Check that the new state has not been visited before   + Find a path from the neighbour to the goal.   The search terminates when we have found the state:  state(0, 0, right). Possible Moves  * A move is characterised by the number of missionaries and the number of cannibals taken in the boat at one time. * Since the boat can carry no more than two people at once, the only possible combinations are:   carry(2, 0).  carry(1, 0).  carry(1, 1).  carry(0, 1).  carry(0, 2).   * where carry(M, C) means the boat will carry M missionaries and C cannibals on one trip.  Feasible Moves  * Once we have found a possible move, we have to confirm that it is feasible. * I.e. it is not feasible to move more missionaries or more cannibals than are present on one bank. * When the state is state(M1, C1, left) and we try carry(M, C) then   M <= M1 and C <= C1   * must be true. * When the state is state(M1, C1, right) and we try carry(M, C) then   M + M1 <= 3 and C + C1 <= 3   * must be true.  Legal Moves  * Once we have found a feasible move, we must check that is legal. * I.e. no missionaries must be eaten.   legal(X, X).  legal(3, X).  legal(0, X).   * The only safe combinations are when there are equal numbers of missionaries and cannibals or all the missionaries are on one side. |
| **Video Content / Details of website for further learning (if any):**   1. [www.artint.info/html/ArtInt.htm](http://www.artint.info/html/ArtInt.htm)l 2. [www.aima.cs.berkeley.ed](http://www.aima.cs.berkeley.ed/)u 3. [www-formal.stanford.edu/jmc/whatisai](http://www-formal.stanford.edu/jmc/whatisai/)/ 4. [www.nptel.ac.in/courses/10610612](http://www.nptel.ac.in/courses/10610612)6 |
| **Important Books/Journals for further learning including the page nos.:**  1. Kevin Night and Elaine Rich, Nair B., “Artificial Intelligence (SIE)”, McGraw Hill- 2008.  Page No. : 82- 88 |

**Course Faculty**

**Verified by HOD**

|  |  |  |
| --- | --- | --- |
| Logo_MEC  **LECTURE HANDOUTS**  **III/ V**  **IT** | **MUTHAYAMMAL ENGINEERING COLLEGE**  **(An Autonomous Institution)**  **(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)**  **Rasipuram - 637 408, Namakkal Dist., Tamil Nadu** | **L7** |

**Course Name with Code : 19ITC17 Artificial Intelligence**

**Course Faculty : P.Bhuvaneshwari**

**Unit : I - introduction to al and production systems**

**Date of Lecture:**

|  |
| --- |
| **Topic of Lecture:** Problem solving methods |
| **Introduction:**   * Various search strategies used to find the solution for a given problem. * Strategies to analyze while choosing the Search method. |
| **Prerequisite knowledge for Complete understanding and learning of Topic:**   * Knowledge about Production system and Rule Generation. * Discrete Mathematics * Graph Theory- Basics   **Detailed Content:**   * **Search Strategies**   There are two types of strategies that describe a solution for a given problem:   * **Uninformed Search (Blind Search)**   This type of search strategy does not have any additional information about the states except the information provided in the problem definition. They can only generate the successors and distinguish a goal state from a non-goal state. These type of search does not maintain any internal state, that’s why it is also known as **Blind search.**  **There are following types of uninformed searches:**   * Breadth-first search * Uniform cost search * Depth-first search * Depth-limited search * Iterative deepening search * **Informed Search (Heuristic Search)**   This type of search strategy contains some additional information about the states beyond the problem definition. This search uses problem-specific knowledge to find more efficient solutions. This search maintains some sort of internal states via heuristic functions (which provides hints), so it is also called **heuristic search.**  **There are following types of informed searches:**   * Best first search (Greedy search) * A\* search  Breadth-first search (BFS): It is a simple search strategy where the root node is expanded first, then covering all other successors of the root node, further move to expand the next level nodes and the search continues until the goal node is not found.  **BFS**expands the shallowest (i.e., not deep) node first using FIFO (First in first out) order. Thus, new nodes (i.e., children of a parent node) remain in the queue and old unexpanded node which are shallower than the new nodes, get expanded first.  In BFS, goal test **(a test to check whether the current state is a goal state or not)** is applied to each node at the time of its generation rather when it is selected for expansion.  Breadth-first-search-tree  **Breadth-first search**  In the above figure, it is seen that the nodes are expanded level by level starting from the root node **A** till the last node**I** in the tree. Therefore, the BFS sequence followed is: **A->B->C->D->E->F->G->I**.  **BFS Algorithm**   * Set a variable **NODE** to the initial state, i.e., the root node. * Set a variable **GOAL** which contains the value of the goal state. * Loop each node by traversing level by level until the goal state is not found. * While performing the looping, start removing the elements from the queue in **FIFO** order. * If the goal state is found, **return goal state** otherwise continue the search.   **The performance measure of BFS is as follows:**   * **Completeness:** It is a complete strategy as it definitely finds the goal state. * **Optimality:** It gives an optimal solution if the cost of each node is same. * **Space Complexity:** The space complexity of BFS is **O(bd)**, i.e., it requires a huge amount of memory. Here, b is the **branching factor** and d denotes the **depth/level** of the tree * **Time Complexity:** BFS consumes much time to reach the goal node for large instances.   **Disadvantages of BFS**   * The biggest disadvantage of BFS is that it requires a lot of memory space, therefore it is a memory bounded strategy. * BFS is time taking search strategy because it expands the nodes **breadthwise.**   **Note:** BFS expands the nodes level by level, i.e., breadthwise, therefore it is also known as a **Level search technique.** Uniform-cost search: Unlike BFS, this uninformed search explores nodes based on their path cost from the root node. It expands a node n having the lowest path cost g(n), where g(n) is the total cost from a root node to node n. Uniform-cost search is significantly different from the breadth-first search because of the following two reasons:   * First, the goal test is applied to a node only when it is selected for expansion **not when it is first generated** because the first goal node which is generated may be on a suboptimal path. * Secondly, a goal test is added to a node, only when a better/optimal path is found.   Thus, uniform-cost search expands nodes in a sequence of their **optimal path cost** because before exploring any node, it searches the optimal path. Also, the step cost is positive so, paths never get shorter when a new node is added in the search.  Uniform-cost-search-on-a-binary-tree  **Uniform-cost search on a binary tree**  In the above figure, it is seen that the goal-state is **F** and start/initial state is **A**. There are three paths available to reach the goal node. We need to select an optimal path which may give the lowest total cost g(n). Therefore, **A->B->E->F** gives the optimal path cost i.e., **0+1+3+4=8.**  **Uniform-cost search Algorithm**   * Set a variable **NODE** to the initial state, i.e., the root node and expand it. * After expanding the root node, select one node having the lowest path cost and expand it further. Remember,**the selection of the node should give an optimal path cost.** * If the goal node is searched with optimal value, return **goal state**, else carry on the search.   **The performance measure of Uniform-cost search**   * **Completeness:** It guarantees to reach the goal state. * **Optimality:**  It gives optimal path cost solution for the search. * **Space and time complexity:** The worst space and time complexity of the uniform-cost search is **O(b1+LC\*/ᵋ˩).**   **Note:** When the path cost is same for all the nodes, it behaves similar to **BFS.**  **Disadvantages of Uniform-cost search**   * It does not care about the number of steps a path has taken to reach the goal state. * It may stick to an infinite loop if there is a path with infinite zero cost sequence. * It works hard as it examines each node in search of lowest cost path.  Depth-first search This search strategy explores the deepest node first, then backtracks to explore other nodes. It uses **LIFO (Last in First Out)** order, which is based on the **stack,**in orderto expand the unexpanded nodes in the search tree. The search proceeds to the deepest level of the tree where it has no successors. This search expands nodes till infinity, i.e.,  the depth of the tree.  Depth-first-search  **DFS search tree**  In the above figure, DFS works starting from the initial node **A** (root node) and traversing in one direction deeply till node **I** and then backtrack to B and so on. Therefore,  the sequence will be **A->B->D->I->E->C->F->G.**  **DFS Algorithm**   * Set a variable **NODE** to the initial state, i.e., the root node. * Set a variable **GOAL** which contains the value of the goal state. * Loop each node by traversing deeply in one direction/path in search of the goal node. * While performing the looping, start removing the elements from the stack in **LIFO** order. * If the goal state is found, **return goal state** otherwise backtrack to expand nodes in other direction.   **The performance measure of DFS**   * **Completeness:** DFS does not guarantee to reach the goal state. * **Optimality:** It does not give an optimal solution as it expands nodes in one direction deeply. * **Space complexity**: It needs to store only a single path from the root node to the leaf node. Therefore, DFS has **O(bm)** space complexity where b is the **branching factor(i.e., total no. of child nodes, a parent node have)** and m is the **maximum length of any path.** * **Time complexity**: DFS has **O(bm)** time complexity.   **Disadvantages of DFS**   * It may get trapped in an infinite loop. * It is also possible that it may not reach the goal state. * DFS does not give an optimal solution.   **Note:** DFS uses the concept of**backtracking**to explore each node in a search tree. Depth-limited search This search strategy is similar to DFS with a little difference. The difference is that in depth-limited search, we limit the search by imposing a **depth limit l** to the depth of the search tree. It does not need to explore till infinity. As a result, the **depth-first search is a special case of depth-limited search.** when the limit l is infinite.  Depth-limited-search-on-a-binary-tree  **Depth-limited search on a binary tree**  In the above figure, the **depth-limit is 1.** So, only level 0 and 1 get expanded in **A->B->C** DFS sequence, starting from the root node **A** till node **B**. It is not giving satisfactory result because we could not reach the goal node**I**.  **Depth-limited search Algorithm**   * Set a variable **NODE** to the initial state, i.e., the root node. * Set a variable **GOAL** which contains the value of the goal state. * Set a variable**LIMIT** which carries a depth-limit value. * Loop each node by traversing in **DFS**manner till the depth-limit value. * While performing the looping, start removing the elements from the stack in **LIFO** order. * If the goal state is found, **return goal state**. Else **terminate the search.**   **The performance measure of Depth-limited search**     * **Completeness:** Depth-limited search does not guarantee to reach the goal node. * **Optimality:**It does not give an optimal solution as it expands the nodes till the depth-limit. * **Space Complexity:** The space complexity of the depth-limited search is **O(bl).** * **Time Complexity:** The time complexity of the depth-limited search is **O(bl).**   **Disadvantages of Depth-limited search**   * This search strategy is not complete. * It does not provide an optimal solution.   **Note:** Depth-limit search terminates with two kinds of failures: the **standard failure value** indicates “no solution,” and **cut-off value**, which indicates “no solution within the depth-limit.” Iterative deepening depth-first search/Iterative deepening search: This search is a combination of BFS and DFS, as BFS guarantees to reach the goal node and DFS occupies less memory space. Therefore, iterative deepening search combines these two advantages of BFS and DFS to reach the goal node. It gradually increases the depth-limit from **0,1,2** and so on and reach the goal node.  In the above figure, the goal node is**H** and initial **depth-limit =[0-1]**. So, it will expand level 0 and 1 and will terminate with **A->B->C** sequence. Further, change the **depth-limit =[0-3]**, it will again expand the nodes from level 0 till level 3 and the search terminate with **A->B->D->F->E->H** sequence where **H**is the desired goal node.  Iterative-deepening-depth-first-search-Iterative-deepening-search  **Iterative deepening search Algorithm**   * Explore the nodes in DFS order. * Set a LIMIT variable with a limit value. * Loop each node up to the limit value and further increase the limit value accordingly. * Terminate the search when the goal state is found.   **The performance measure of Iterative deepening search**   * **Completeness:**Iterative deepening search may or may not reach the goal state. * **Optimality:**It does not give an optimal solution always. * **Space Complexity:**It has the same space complexity as BFS, i.e., **O(bd).** * **Time Complexity:**It has **O(d)** time complexity.   **Disadvantages of Iterative deepening search**   * The drawback of iterative deepening search is that it seems wasteful because it generates states multiple times.   **Note:** Generally, iterative deepening search is required when the search space is large, and the depth of the solution is unknown. |
| **Video Content / Details of website for further learning (if any):**   1. [www.artint.info/html/ArtInt.htm](http://www.artint.info/html/ArtInt.htm)l 2. [www.aima.cs.berkeley.ed](http://www.aima.cs.berkeley.ed/)u 3. [www-formal.stanford.edu/jmc/whatisai](http://www-formal.stanford.edu/jmc/whatisai/)/ 4. [www.nptel.ac.in/courses/10610612](http://www.nptel.ac.in/courses/10610612)6 | |
| **Important Books/Journals for further learning including the page nos.:**  1. Kevin Night and Elaine Rich, Nair B., “Artificial Intelligence (SIE)”, McGraw Hill- 2008.  Page No. : 44- 55 | |

**Course Faculty**

**Verified by HOD**

|  |  |  |
| --- | --- | --- |
| Logo_MEC  **LECTURE HANDOUTS**  **III/ V**  **IT** | **MUTHAYAMMAL ENGINEERING COLLEGE**  **(An Autonomous Institution)**  **(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)**  **Rasipuram - 637 408, Namakkal Dist., Tamil Nadu** | **L8** |

**Course Name with Code : 19ITC17 Artificial Intelligence**

**Course Faculty : P.Bhuvaneshwari**

**Unit : I - introduction to al and production systems**

**Date of Lecture:**

|  |
| --- |
| **Topic of Lecture:** Heuristic functions- Hill climbing |
| **Introduction:**    **Heuristic Functions is** a **function** that ranks alternatives in search algorithms at each branching step based on available information to decide which branch to follow. For example, it may approximate the exact solution. |
| **Prerequisite knowledge for Complete understanding and learning of Topic:**   * Basics of Mathematics * Problem solving methods |
| **Detailed content of the Lecture:** Properties of a Heuristic search AlgorithmUse of heuristic function in a heuristic search algorithm leads to following properties of a heuristic search algorithm:  * **Admissible Condition:** An algorithm is said to be admissible, if it returns an optimal solution. * **Completeness:** An algorithm is said to be complete, if it terminates with a solution (if the solution exists). * **Dominance Property:** If there are two admissible heuristic algorithms **A1**and**A2** having **h1** and**h2** heuristic functions, then**A1** is said to dominate **A2** if **h1** is better than **h2** for all the values of node **n.** * **Optimality Property:** If an algorithm is **complete, admissible**, and **dominating** other algorithms, it will be the best one and will definitely give an optimal solution.   **Example: 8-puzzle problem**  Consider the following 8-puzzle problem where we have a start state and a goal state. Our task is to slide the tiles of the current/start state and place it in an order followed in the goal state. There can be four moves either **left, right, up, or down**. There can be several ways to convert the current/start state to the goal state, but, we can use a heuristic function h(n) to solve the problem more efficiently. A heuristic function for the 8-puzzle problem is defined below: heuristic-function-for-the-8-puzzle-problem  **h(n)=Number of tiles out of position.**  So, there is total of three tiles out of position i.e., 6,5 and 4. Do not count the empty tile present in the goal state). i.e. h(n)=3. Now, we require to minimize the value of **h(n) =0.**  We can construct a state-space tree to minimize the h(n) value to 0, as shown below:  heuristic-function-for-the-8-puzzle-problem-2  It is seen from the above state space tree that the goal state is minimized from h(n)=3 to h(n)=0. However, we can create and use several heuristic functions as per the reqirement. It is also clear from the above example that a heuristic function h(n) can be defined as the information required to solve a given problem more efficiently. The information can be related to the **nature of the state, cost of transforming from one state to another, goal node characterstics,** etc., which is expressed as a heuristic function.   * **Hill Climbing Algorithm**   Hill climbing search is a local search problem. The purpose of the hill climbing search is to climb a hill and reach the topmost peak/point of that hill. It is based on the **heuristic search technique** where the person who is climbing up on the hill estimates the direction which will lead him to the highest peak.                                          State-space Landscape of Hill climbing algorithm To understand the concept of hill climbing algorithm, consider the below landscape representing the **goal state/peak** and the **current state** of the climber. The topographical regions shown in the figure can be defined as:   * **Global Maximum:** It is the highest point on the hill, which is the goal state. * **Local Maximum:** It is the peak higher than all other peaks but lower than the global maximum. * **Flat local maximum:** It is the flat area over the hill where it has no uphill or downhill. It is a saturated point of the hill. * **Shoulder:** It is also a flat area where the summit is possible. * **Current state:** It is the current position of the person.   **Types of Hill climbing search algorithm**  There are following types of hill-climbing search:   * Simple hill climbing * Steepest-ascent hill climbing * Stochastic hill climbing * Random-restart hill climbing  Simple hill climbing search Simple hill climbing is the simplest technique to climb a hill. The task is to reach the highest peak of the mountain. Here, the movement of the climber depends on his move/steps. If he finds his next step better than the previous one, he continues to move else remain in the same state. This search focus only on his previous and next step.   * **Simple hill climbing Algorithm**  1. Create a **CURRENT** node, **NEIGHBOUR**node, and a **GOAL**node. 2. If the **CURRENT node=GOAL node**, return **GOAL** and terminate the search. 3. Else  **CURRENT node<= NEIGHBOUR node,** move ahead. 4. Loop until the goal is not reached or a point is not found.   **Steepest-ascent hill climbing**  Steepest-ascent hill climbing is different from simple hill climbing search. Unlike simple hill climbing search, It considers all the successive nodes, compares them, and chooses the node which is closest to the solution. Steepest hill climbing search is similar to **best-first search** because it focuses on each node instead of one.  Note: Both simple, as well as steepest-ascent hill climbing search, fails when there is no closer node. Steepest-ascent hill climbing algorithm  1. Create a**CURRENT** node and a **GOAL**node. 2. If the **CURRENT node=GOAL** node, return **GOAL** and terminate the search. 3. Loop until a better node is not found to reach the solution. 4. If there is any better successor node present, expand it. 5. When the **GOAL**is attained, return **GOAL** and terminate.  * **Stochastic hill climbing**   Stochastic hill climbing does not focus on all the nodes. It selects one node at random and decides whether it should be expanded or search for a better one.   * **Random-restart hill climbing**   Random-restart algorithm is based on **try and try strategy**. It iteratively searches the node and selects the best one at each step until the goal is not found. The success depends most commonly on the shape of the hill. If there are few plateaus, local maxima, and ridges, it becomes easy to reach the destination. Limitations of Hill climbing algorithm Hill climbing algorithm is a fast and furious approach. It finds the solution state rapidly because it is quite easy to improve a bad state. But, there are following limitations of this search:   * **Local Maxima:** It is that peak of the mountain which is highest than all its neighboring states but lower than the global maxima. It is not the goal peak because there is another peak higher than it. * **Plateau:** It is a flat surface area where no uphill exists. It becomes difficult for the climber to decide that in which direction he should move to reach the goal point. Sometimes, the person gets lost in the flat area. * **Ridges:** It is a challenging problem where the person finds two or more local maxima of the same height commonly. It becomes difficult for the person to navigate the right point and stuck to that point itself.  Simulated Annealing  * Simulated annealing is similar to the hill climbing algorithm. * It works on the current situation. * It picks **a random move** instead of picking **the best move**. * If the move leads to the improvement of the current situation, it is always accepted as a step towards the solution state, else it accepts the move having **a probability less than 1**. * This search technique was first used in **1980** to solve **VLSI layout** problems. * It is also applied for factory scheduling and other large optimization tasks. |
| **Video Content / Details of website for further learning (if any):**   1. [www.artint.info/html/ArtInt.htm](http://www.artint.info/html/ArtInt.htm)l 2. [www.aima.cs.berkeley.ed](http://www.aima.cs.berkeley.ed/)u 3. [www-formal.stanford.edu/jmc/whatisai](http://www-formal.stanford.edu/jmc/whatisai/)/ 4. [www.nptel.ac.in/courses/10610612](http://www.nptel.ac.in/courses/10610612)6 |
| **Important Books/Journals for further learning including the page nos.:**  1. Kevin Night and Elaine Rich, Nair B., “Artificial Intelligence (SIE)”, McGraw Hill- 2008.  Page No. : 65- 73 |

**Course Faculty**

**Verified by HOD**

|  |  |  |
| --- | --- | --- |
| Logo_MEC  **LECTURE HANDOUTS**  **III/ V**  **IT** | **MUTHAYAMMAL ENGINEERING COLLEGE**  **(An Autonomous Institution)**  **(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)**  **Rasipuram - 637 408, Namakkal Dist., Tamil Nadu** | **L9** |

**Course Name with Code : 19ITC17 Artificial Intelligence**

**Course Faculty : P.Bhuvaneshwari**

**Unit : I - introduction to al and production systems**

**Date of Lecture:**

|  |
| --- |
| **Topic of Lecture:** Constraint Satisfaction problem- Related algorithms- Measure of performance and analysis of algorithms |
| **Introduction:**   * Constraint satisfaction is a technique where a problem is solved when its values satisfy certain constraints or rules of the problem. * Performance metrics to analyze the optimum problem solving strategy |
| **Prerequisite knowledge for Complete understanding and learning of Topic:**   * Search Strategies * Basic Mathematics * Time Complexity * Space Complexity |
| **Detailed content of the Lecture:**  Constraint satisfaction is a technique where a problem is solved when its values satisfy certain constraints or rules of the problem. Such type of technique leads to a deeper understanding of the problem structure as well as its complexity.  Constraint satisfaction depends on three components, namely:   * **X:** It is a set of variables. * **D:** It is a set of domains where the variables reside. There is a specific domain for each variable. * **C:** It is a set of constraints which are followed by the set of variables.   In constraint satisfaction, domains are the spaces where the variables reside, following the problem specific constraints. These are the three main elements of a constraint satisfaction technique. The constraint value consists of a pair of **{scope, rel}**. The **scope** is a tuple of variables which participate in  the constraint and **rel** is a relation which includes a list of values which the variables can take to satisfy the constraints of the problem.  **Solving Constraint Satisfaction Problems**  The requirements to solve a constraint satisfaction problem (CSP) is:   * A state-space * The notion of the solution.   A state in state-space is defined by assigning values to some or all variables such as  **{X1=v1, X2=v2, and so on…}.**  **An assignment of values to a variable can be done in three ways:**   * **Consistent or Legal Assignment:** An assignment which does not violate any constraint or rule is called Consistent or legal assignment. * **Complete Assignment:** An assignment where every variable is assigned with a value, and the solution to the CSP remains consistent. Such assignment is known as Complete assignment. * **Partial Assignment:**An assignment which assigns values to some of the variables only. Such type of assignments are called Partial assignments.   **Types of Domains in CSP**  **There are following two types of domains which are used by the variables :**   * **Discrete Domain:** It is an infinite domain which can have one state for multiple variables. **For example,** a start state can be allocated infinite times for each variable. * **Finite Domain:** It is a finite domain which can have continuous states describing one domain for one specific variable. It is also called a continuous domain.   **Constraint Types in CSP**  With respect to the variables, basically there are following types of constraints:   * **Unary Constraints:** It is the simplest type of constraints that restricts the value of a single variable. * **Binary Constraints:** It is the constraint type which relates two variables. A value **x2** will contain a value which lies between **x1**and **x3**. * **Global Constraints:** It is the constraint type which involves an arbitrary number of variables.   **Some special types of solution algorithms are used to solve the following types of constraints:**   * **Linear Constraints:** These type of constraints are commonly used in linear programming where each variable containing an integer value exists in linear form only. * **Non-linear Constraints:** These type of constraints are used in non-linear programming where each variable (an integer value) exists in a non-linear form.   **Note:** A special constraint which works in real-world is known as**Preference constraint.**  **Constraint Propagation**  In local state-spaces, the choice is only one, i.e., to search for a solution. But in CSP, we have two choices either:   * We can search for a solution or * We can perform a special type of inference called **constraint propagation**.   Constraint propagation is a special type of inference which helps in reducing the legal number of values for the variables.The idea behind constraint propagation is **local consistency.**  In local consistency, variables are treated as **nodes**, and each binary constraint is treated as an **arc** in the given problem. **There are following local consistencies which are discussed below:**   * **Node Consistency:** A single variable is said to be node consistent if all the values in the variable’s domain satisfy the unary constraints on the variables. * **Arc Consistency:** A variable is arc consistent if every value in its domain satisfies the binary constraints of the variables. * **Path Consistency:** When the evaluation of a set of two variable with respect to a third variable can be extended over another variable, satisfying all the binary constraints. It is similar to arc consistency. * **k-consistency:** This type of consistency is used to define the notion of stronger forms of propagation. Here, we examine the k-consistency of the variables.   **Example:**   * **Cryptarithmetic Problem:**This problem has one most important constraint that is, we cannot assign a different digit to the same character. All digits should contain a unique alphabet. * Cryptarithmetic Problem is a type of constraint satisfaction problem where the game is about digits and its unique replacement either with alphabets or other symbols. In cryptarithmetic problem, the digits  (0-9) get substituted by some possible alphabets or symbols. The task in cryptarithmetic problem is to substitute each digit with an alphabet to get the result arithmetically correct.   We can perform all the arithmetic operations on a given cryptarithmetic problem.  **The rules or constraints on a cryptarithmetic problem are as follows:**   * There should be a unique digit to be replaced with a unique alphabet. * The result should satisfy the predefined arithmetic rules, i.e., 2+2 =4, nothing else. * Digits should be from **0-9** only. * There should be only one carry forward, while performing the addition operation on a problem. * The problem can be solved from both sides, i.e., **lefthand side (L.H.S), or righthand side (R.H.S)**   Let’s understand the cryptarithmetic problem as well its constraints better with the help of an example:   * Given a cryptarithmetic problem, i.e., **S E N D + M O R E = M O N E Y**   In this example, add both terms **S E N D** and **M O R E**to bring **M O N E Y** as a result.  **Follow the below steps to understand the given problem by breaking it into its subparts:**   * Starting from the left hand side (L.H.S) , the terms are **S** and **M**. Assign a digit which could give a satisfactory result. Let’s assign**S->9** and **M->1**.     Hence, we get a satisfactory result by adding up the terms and got an assignment for **O** as **O->0**as well.   * Now, move ahead to the next terms **E** and **O** to get**N**as its output.     **Adding E and O, which means 5+0=0, which is not possible because** according to cryptarithmetic constraints, we cannot assign the same digit to two letters. So, we need to think more and assign some other value.    **Note: When we will solve further, we will get one carry, so after applying it, the answer will be satisfied.**     * Further, adding the next two terms **N** and **R** we get,     But, we have already assigned **E->5**. Thus, the above result does not satisfy the values  because we are getting a different value for **E.** So, we need to think more.  **Again, after solving the whole problem, we will get a carryover on this term, so our answer will be satisfied.**    **where 1 will be carry forward to the above term**  Let’s move ahead.   * Again, on adding the last two terms, i.e., the rightmost terms **D**and **E**, we get **Y** as its result.     **where 1 will be  carry forward to the above term**   * Keeping all the constraints in mind, the final resultant is as follows:      * Below is the representation of the assignment of the digits to the alphabets. |
| **Video Content / Details of website for further learning (if any):**   1. [www.artint.info/html/ArtInt.htm](http://www.artint.info/html/ArtInt.htm)l 2. [www.aima.cs.berkeley.ed](http://www.aima.cs.berkeley.ed/)u 3. [www-formal.stanford.edu/jmc/whatisai](http://www-formal.stanford.edu/jmc/whatisai/)/ 4. [www.nptel.ac.in/courses/10610612](http://www.nptel.ac.in/courses/10610612)6 |
| **Important Books/Journals for further learning including the page nos.:**  1. Kevin Night and Elaine Rich, Nair B., “Artificial Intelligence (SIE)”, McGraw Hill- 2008.  Page No. : 74- 80 |

**Course Faculty**

**Verified by HOD**