Lecture 11:
This lecture explores advanced search techniques for game-playing AI, moving beyond basic minimax and alpha-beta pruning.

Key topics include:

- **Move Ordering:** Emphasizes how ordering moves effectively (e.g., trying captures first, using "killer moves," or iterative deepening) significantly improves the efficiency of alpha-beta pruning, effectively doubling the search depth.
- **Transposition Tables:** Discusses using hash tables to store and reuse evaluations of previously encountered positions, further enhancing search depth.
- **Imperfect Real-time Decisions:** Introduces heuristic minimax, where a cutoff test and an evaluation function (EVAL) replace terminal tests and utility functions, making the algorithms practical for real-time games with time limits.
- **Evaluation Functions:** Details how evaluation functions estimate the utility of non-terminal game states. Good evaluation functions are strongly correlated with winning chances, computationally efficient, and can be structured as weighted linear functions of various game features (e.g., material values in chess).
- **Cutting Off Search:** Explains that simple fixed-depth cutoffs can lead to errors. More sophisticated methods include:
    - **Quiescence Search:** Expanding non-quiescent positions (those likely to have wild value swings) further until quiescent positions are reached.
    - **Horizon Effect Mitigation:** Using singular extensions to consider "clearly better" moves beyond the normal depth limit to avoid temporary delays of unavoidable damage.
- **Forward Pruning:** Explores techniques like beam search and PROBCUT to prune moves immediately without full consideration, acknowledging the risk of pruning the best move. PROBCUT uses statistical probabilities to make more informed pruning decisions.
- **Search vs. Lookup:** Highlights that many game programs use table lookups for opening and ending game phases, leveraging human expertise for openings and computer-solved endgames.
- **Stochastic Games:** Introduces games with random elements (like dice rolls), such as backgammon.
    - **Chance Nodes:** Explains the need for chance nodes in game trees to represent probabilistic outcomes.
    - **Expectiminimax:** Generalizes minimax to handle chance nodes by calculating the expected value of a position (weighted average of outcomes).
    - **Evaluation in Stochastic Games:** Notes the sensitivity of move choices to the scale of evaluation values and suggests using a positive linear transformation of the probability of winning.
    - **Computational Challenges:** Highlights that stochastic games significantly increase the search complexity, making deep searches impractical.
    - **Pruning and Simulation:** Mentions that alpha-beta pruning can be adapted for chance nodes, and Monte Carlo simulations (rollouts) can be used to evaluate positions by playing thousands of games against itself with random dice rolls.

The document concludes by noting that combining these techniques allows for expert-level play in games like chess, with grandmaster status requiring extensively tuned evaluation functions and large databases. It also briefly mentions other areas of game-playing research like partially observable games.

Lecture 12:
This lecture on "Local Search" covers the following topics:

- **Local Search Algorithms (LSAs)**: These algorithms are used when the path to the goal doesn't matter. They operate with a single current node, moving to its neighbors, and use very little memory. LSAs are useful for optimization problems, aiming to find the best state based on an objective function.
- **Search Space Landscape**: This concept describes the state space with location (state) and elevation (heuristic cost or objective function). The goal is to find either a global minimum (for cost) or a global maximum (for objective function).
- **Hill Climbing Search**: This greedy local search algorithm continuously moves in the direction of increasing value until it reaches a peak where no neighbor has a higher value. It's called "greedy" because it doesn't look beyond the immediate neighbors.
    - **Limitations**: Hill climbing can get stuck due to local maxima (peaks lower than the global maximum), ridges (sequences of local maxima), and plateaus (flat areas).
    - **Variations**:
        - **Stochastic hill climbing**: Randomly chooses from uphill moves, with probability varying by steepness.
        - **First-choice hill climbing**: Generates successors randomly until one better than the current state is found, useful when there are many successors.
        - **Random-restart hill climbing**: Conducts multiple hill climbing searches from randomly generated initial states, effective even with many local maxima.
- **Genetic Algorithms (GAs)**: These are optimization algorithms based on genetic recombination and "survival of the fittest," with some mutation.
    - **Key features**: They work with an encoding of parameters (not the parameters themselves) and use probabilistic transition rules.
    - **Benefits**: Easy to understand, modular, supports multi-objective optimization, good for noisy environments, always provides an answer that improves over time, and is inherently parallel. They are also flexible for hybrid applications and have a substantial history of use.
    - **When to use GAs**: When alternate solutions are too slow or complicated, as an exploratory tool, for problems similar to those successfully solved by GAs, for hybridization, or when GA benefits meet problem requirements.
    - **Algorithm**: Involves generating an initial population, evaluating fitness, and then iteratively generating new populations through selection, crossover, and mutation until a stopping condition is met.

Lecture 13:

The file "lecture13.pdf" is a presentation on stochastic search techniques, specifically focusing on Genetic Algorithms (GAs), Tabu Search, and Simulated Annealing.

Here's an in-depth review of the content:

Overview of Stochastic Search Techniques

The lecture introduces three stochastic search techniques used for optimization: Genetic Algorithms, Tabu Search, and Simulated Annealing. It delves into the mechanics and applications of each.

Topics Covered:

- **Genetic Algorithms (GAs):**
  - **Selection:** This section explains how individuals with higher fitness are more likely to be chosen for the next generation. It details several selection schemes, including:
    - Deterministic sampling
    - Remainder stochastic sampling (with and without replacement)
    - Stochastic sampling (with and without replacement, including roulette wheel selection)
    - Stochastic tournament (Wetzel ranking)
  - **Crossover:** Discusses how genetic material is exchanged between two chosen strings (parents) to create offspring. Types of crossover explained include:
    - 1-point crossover
    - Uniform crossover
    - n-point crossover
  - **Mutation:** Describes the process of randomly changing a bit (0 to 1 or 1 to 0) in a string according to a mutation probability.
  - **Fitness:** Defined as a measure of the "goodness" of an organism, determining its probability of survival and mating.
  - **Example:** A detailed step-by-step example is provided to illustrate the application of a GA to maximize x^2 over a range of integers, showing the process of initial population, fitness calculation, mating pool selection, crossover, and mutation over one generational cycle.
  - **Representations:** Briefly touches upon different representations beyond binary, such as Gray coding, integers, and floating-point variables.
  - **Population Models:** Compares Generational Models (entire population replaced) and Steady-State Models (one offspring replaces one member), and introduces the concept of Elitism.
  - **Application Types:** Provides a table outlining various domains where GAs are applied, including control, design, scheduling, robotics, machine learning, signal processing, game playing, and combinatorial optimization.
- **Tabu Search:**
  - Described as an optimization technique that works with a candidate solution and a number of neighborhood points, using a "tabu list" of prohibited moves to avoid getting trapped in local optima.
  - Explains the "aspiration criteria" which can override tabu restrictions if a move leads to a better solution than the best known so far.
  - Outlines the procedure and algorithm for Tabu Search.
- **Simulated Annealing (SA):**

    - A probabilistic technique for approximating the global optimum of a function, inspired by the metallurgical process of annealing.
    - Emphasizes the idea of "slow cooling" which corresponds to a slow decrease in the probability of accepting worse solutions, allowing for a more extensive search and avoiding local minima.
    - Details the calculation of the probability of accepting a worse solution based on the change in energy (ΔE) and temperature (T).
    - Presents the algorithm for Simulated Annealing.

This lecture provides a comprehensive introduction to these stochastic search techniques, explaining their core concepts, operators, and providing an illustrative example for Genetic Algorithms.

Lecture 14:
This document, "Lecture 14: Genetic Programming (GP)," by V. Susheela Devi, defines and describes Genetic Programming as a method for evolving computer programs through successive applications of genetic operators.

Key aspects of GP covered in the document include:

- **Definition:** GP adapts hierarchically organized segments of computer programs, maintaining a population of candidate programs whose fitness is measured by their ability to solve tasks. Programs are modified using crossover and mutation.
- **Description:** GP starts with a random population of programs made from arithmetic, logical, and domain-specific functions, as well as data items. New programs are generated using customized genetic operators (crossover, mutation), and their fitness is evaluated based on their performance in a problem environment.
- **Components:** GP has six components, similar to Genetic Algorithms: a set of structures for transformation, initial structures, a domain-dependent fitness measure, genetic operators, termination conditions, and parameters/state descriptions for each generation. LISP s-expressions are a primary representation.
- **Tree Structure:** Genetic operators manipulate tree structures of s-expressions, mapping them into new program segments.
- **Initialization:** Initial structures require defining a set of functions (F) and terminal values (T). A population of initial programs is generated by randomly selecting elements from F and T, forming tree structures.
- **Fitness Measure:** This is problem-domain dependent and evaluates how well programs perform on a set of tasks. Raw fitness often sums errors, while normalized fitness ranges from 0 to 1. Fitness can also reward smaller, more parsimonious programs.
- **Genetic Operators:** Primary operators are reproduction (copying programs) and crossover (exchanging subtrees between programs). Secondary operators include mutation (introducing random changes) and permutation (exchanging terminal symbols or subtrees).
- **Example (Kepler's Third Law):** The document illustrates GP with Kepler's Third Law ($P^2 = cA^3$). It describes setting up terminal symbols (A), functions ({+, -, *, /, sq, sqrt}), creating an initial random population of programs, and using planetary data to define a fitness measure (number of outputs within 20% of correct values).

Lecture 15:
This lecture on "Agents that think logically" by V. Susheela Devi (NPTEL) introduces knowledge-based (KB) agents, knowledge representation (KR), and different types of logic.

Key takeaways:

- **KB Agents:** These agents use a knowledge base (KB) to store facts about the world. They use "TELL" to add facts and "ASK" to query the KB, which then uses an inference mechanism and logical reasoning to provide answers.
- **Knowledge Representation:** KR uses a language defined by syntax (how sentences are represented) and semantics (their actual meaning). The goal is to create new facts (entailment) from existing ones. KR languages should handle qualitative knowledge, allow new knowledge to be inferred, represent general principles and specific situations, capture complex semantic meaning, and allow meta-level reasoning.
- **Types of Knowledge:** The lecture categorizes knowledge into facts about objects, events, performance (how to do things), and meta-knowledge (knowledge about what we know).
- **KR Characteristics:** A good KR language should be expressive, concise, unambiguous, independent of context, and effective. Natural language is expressive but ambiguous.
- **Logic:** Logic consists of a formal system (syntax and semantics) and a proof theory (rules for deducing entailment).
- **Kinds of Logic:**
    - **Propositional Logic:** Symbols represent whole propositions (facts) combined with boolean connectives.
    - **First Order Logic (FOL):** Represents objects, predicates, connectives, and quantifiers to make sentences about anything in the universe.
    - **Temporal Logic:** Reasons about time, assuming the world is ordered by time points or intervals.
    - **Logic using Probability Theory:** Allows agents to have a degree of belief (0 to 1).
    - **Fuzzy Logic:** Deals with degrees of truth.
- **Ontological and Epistemological Commitment:** The lecture briefly touches on how different logics commit to what exists in the world (ontological) and what an agent believes (epistemological). For example, propositional logic and FOL deal with facts, while probability and fuzzy logic deal with degrees of belief.