Session 5

I. Announcements [5 minutes]

- Solutions to Assignment 0 are posted on the class web-page.
- Assignment 2 is due October 6-th; that’s Next Thursday.
  - You are allowed to work in pairs; but not required. Those who work alone will not receive special treatment or preferential treatment.
  - Who does not have a partner that would like to work in a group.

II. Adversarial Search [15 minutes]

- Adversarial search is a mixture of game theory and classic search; a special case of both.
  - We’ve already talked about search—now we use to find optimal strategies!
  - Game Theory – the formal process of decision making in competitive environments. This area of study seeks to identify the optimal strategy for players by assuming opponents will play optimally.
    - **Prisoner’s Dilemma** (Name building exercise) – John and Sue are arrested for theft (1 year in prison). The police only have sufficient evidence to convict them with a minor crime of trespassing (1 week in jail). Separately, the police offer John and Sue a deal to get no prison time if they confess and implicate the other prisoner of conspiracy (2 years in jail). What should John and Sue do optimally?
    - **Paper Rock Scissors** potential strategies (which is best?):
      - Always play Rock.
      - Play Rock ½ the time and Paper ½ the time.
- The games we consider are zero-sum, turn-taking, deterministic, 2-player games of **perfect information**.
  - **game tree** – a representation that represents all legal sequences of decisions.
    - **root** – the initial state of the game.
    - **(internal) nodes** – represents decision made by current players.
    - **edges** – legal choices for a given decision in the tree.
    - **terminal node** – an ending of the game giving a utility to each player.
- **optimal strategy** – a contingent strategy that leads to an outcome at least as good as any other strategy by assuming the opponent is infallible.
• **Stopping search prematurely** – time limits prevent full exploration of the tree.
  o **evaluation function** – a “heuristic” for accessing the utility of a nonterminal game state; an estimate of the expected value of a state.
  o **features** – elements of the state that indicate its strength.
  o **quiescent state** - unlikely to have major changes in the near future.
  o **horizon effect** – an unavoidable damaging move looms on the horizon.
  o **singular extensions** – a move that is “clearly better” than others.

**Games of Chance with imperfect information**

• **averaging over clairvoyancy** – the strategy of computing optimal moves by averaging over possibilities for the unseen variables.
  o This strategy is flawed as it assumes all future uncertainty will have disappeared by the time the future is reached.
  o Thus, the strategy never makes moves that seek to reveal information.

• **belief states** – games states are replaced by possible states along with their corresponding probabilities.

• In games of imperfect information, it’s best to reveal as little as possible, often by acting unpredictably.
III. AIMA [30 minutes]

- The first real project is due soon and you need to be able to use AIMA in order to effectively use your time.
  - Hopefully, everybody has already started on their projects and you have questions prepared. For everybody else, you need to start on your project immediately.

- Track considerations
  - Weakly connected components and multiple components using 1 grid.
  - State-space of edges – MxNx4 matrix of connections.
  - etc.

- Large-scale LISP
  - Top-Down and Bottom-Up Programming
    - In LISP we don’t just do top-down programming, we also do bottom-up –building the compiler up to our program.
    - While we won’t be writing huge extensions to the compiler in this class, we can
  - Rapid Prototyping
    - Write a specification for a function
    - Write the function
      - Implement dependent functions with stubs to be done upon completion of this program.
    - Test the functions individually – do not proceed until each function works independently; debugging an entire project at once in LISP is a painstaking.
    - After building and testing your functions, integrate them by implementing stubs in the same manner. Continue until entire program is implemented and correct.

- Questions
- Group Work