Artificial Intelligence

Search III

Lecture 5

(September 8, 1999)

Tralvex (Rex) Yeap MAAAI MSCS

University of Leeds

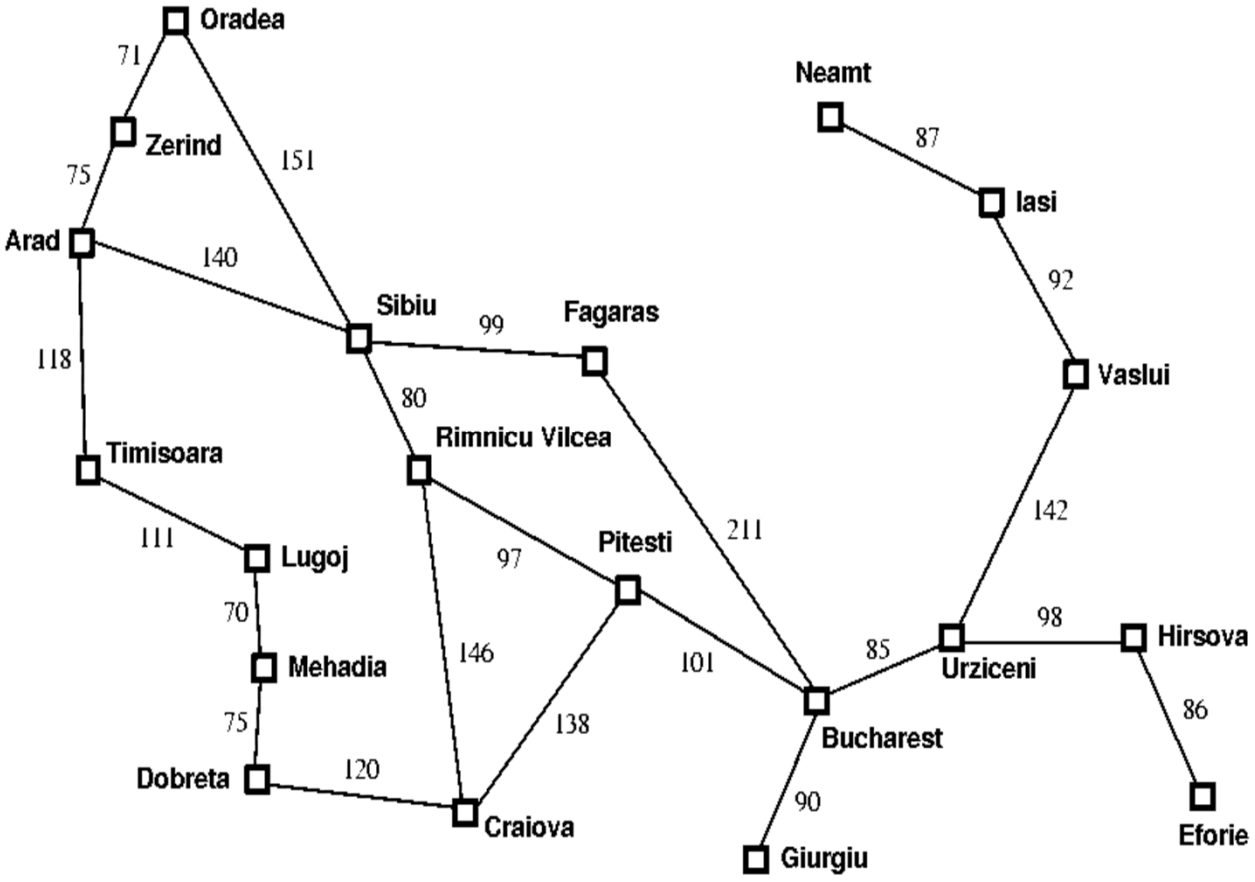
Content: Search III

- Quick Review on Lecture 4
- Class Activity 1: A* Search Workout for Romania Map
- Why Study Games?
- Game Playing as Search
- Special Characteristics of Game Playing Search
- Ingredients of 2-Person Games
- Normal and Game Search Problem
- A Game Tree for Naughts and Crosses
- A Perfect Decision Strategy Based on Minimaxing
- The Need for Imperfect Decision
- Assigning Utilities: Evaluation Functions
- When to Cut Search
- Chess Positions
- Alpha - Beta Pruning
- State of the Art in Checkers, Backgammon, Go & Chess
- Students' Mini Research Presentation by Group B
- Class Activity 2: Real-world Paper Reading & Practical - "The end of an era, the begin-ning of another? HAL, Deep Blue and Kasparov"
- Class Activity 3: Video on Deep Blue vs Kasparov (May, 1997) Game #6.
- Class Activity 4: Real-world Paper Reading & Practical - "Smart Moves - Intelligent Pathfinding"
- What's in Store for Lecture 6

Quick Review on Lecture 5

- Constrained Satisfaction Search
- Heuristic Search Methods
- Definition of a Heuristic Function
- Best First Search
  - An example - Sliding Tiles
  - Greedy Search
  - A* Search
- Heuristics for an 8-puzzle problem
- Iterative Improvement Algorithms
  - Hill-climbing
  - Simulated Annealing
- Students' Mini Research Presentation by Group B
- Class Activity: Real-world Paper Reading - "Smart Moves - Intelligent Pathfinding"

Class Activity 1:
A* Search Workout for Romania Map



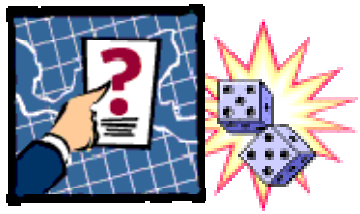
Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Figure 6.1 Map of Romania with road distances in km, and straight-line distances to Bucharest.
h_{SLD}(n) = straight-line distance between n and the goal location.

Class Activity 1:
A* Search Workout for Romania Map (cont)

Search Method	Path	Depth	No. of Nodes involved
Optimal Solution	A S R F B	-	-
Depth First Search (DFS)	A Z O S R P B	6	7
Breadth First Search (BFS)	A Z S T O F R L B	3	9
Bi-directional Search (BDS)	A S F B	2	12
Uniform Cost Search (UCS)	A S F B	3	8
Greedy Search (GS)	A S F B	3	9
A* Search (AS)			

Why Study Games?



Games offer:

- Intellectual Engagement
- Abstraction
- Representability
- Performance Measure

- Not all games are suitable for AI research. We will restrict ourselves to **2 person perfect information** board games.

Game Playing as Search

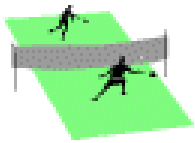
- Playing a game involves searching for the **best move**.
- Board games clearly involve notions like **start state**, **goal state**, **operators**, etc. We can this usefully import problem solving techniques that we have already met.
- There are nevertheless important **differences** from standard search problems.

Special Characteristics of Game Playing Search

Main differences are **uncertainties** introduced by

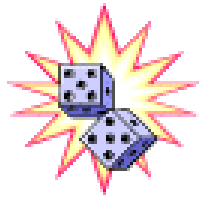
- Presence of an **opponent**. One do not know what the opponent will do until he/she does it. Game playing programs must solve the contingency problem.
- **Complexity**. Most interesting games are simply **too complex to solve by exhaustive means**. Chess, for example, has an average branching factor of *35*. Uncertainty also arises from not having the resources to compute a move which is guaranteed to be the best.

Ingredients of 2-Person Games



- **Players**: We call them Max and Min.
- **Initial State**: Includes board position and whose turn it is.
- **Operators**: These correspond to legal moves.
- **Terminal Test**: A test applied to a board position which determines whether the game is over. In chess, for example, this would be a checkmate or stalemate situation.
- **Utility Function**: A function which assigns a **numeric value** to a **terminal state**. For example, in chess the outcome is win (+1), lose (-1) or draw (0). Note that by convention, we always measure utility relative to Max.

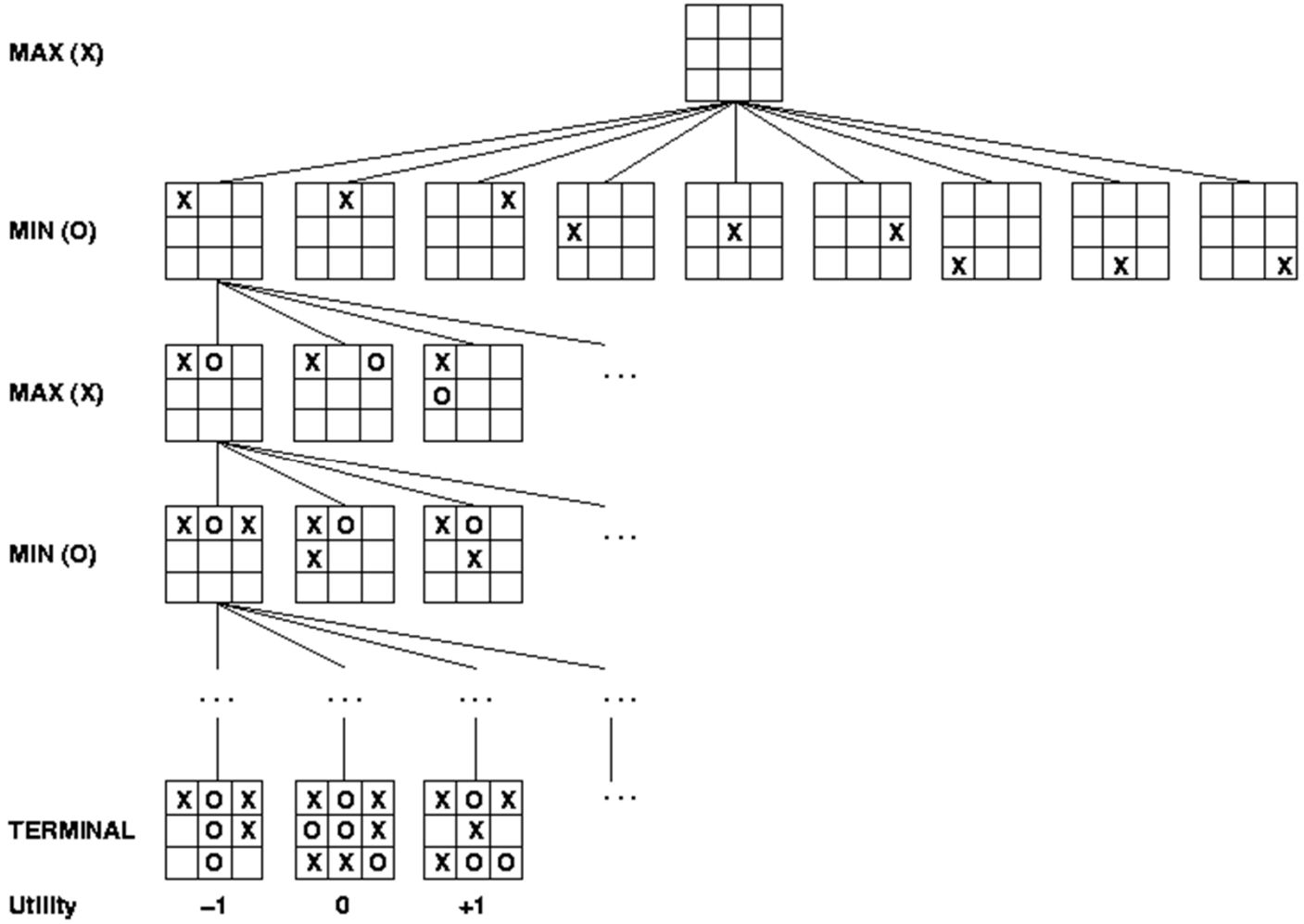
Normal and Game Search Problem



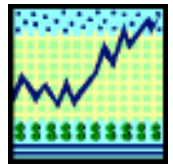
- **Normal search problem**: **Max** searches for a sequence of moves yielding a winning position and then makes the first move in the sequence.

- **Game search problem**: Clearly, this is not feasible in in a game situation where **Min**'s moves must be taken into consideration. **Max** must devise a strategy which leads to a winning position no matter what moves Min makes.

A Game Tree for Naughts and Crosses



A Perfect Decision Strategy Based on Minimaxing



1. Generate the whole game tree.
2. Apply the utility function to leaf nodes to get their values.
3. Use the utility of nodes at level n to derive the utility of nodes at level n-1.
4. Continue backing up values towards the root (one layer at a time).
5. Eventually the backed up values reach the top of the tree, at which point Max chooses the move that yields the highest value. This is called the minimax decision because it maximises the utility for Max on the assumption that Min will play perfectly to minimise it.

The Need for Imperfect Decision

- **Problem**: Minimax assumes the program has **time** to search to the terminal nodes

- **Solution**: **Cut off search earlier** and apply a **heuristic evaluation function** to the leaves.

Assigning Utilities: Evaluation Functions

In chess, evaluation function might be based on

- **Material**: pawn=1, bishop=3, knight=3, rook=5, queen=9
- **Position**: (1) pawns on final rank, (2) check next move (3) fork next move (4) etc.
- Weighted linear **evaluation function**

  $e = w_1 f_1 + w_2 f_2 + ... + w_n f_n$

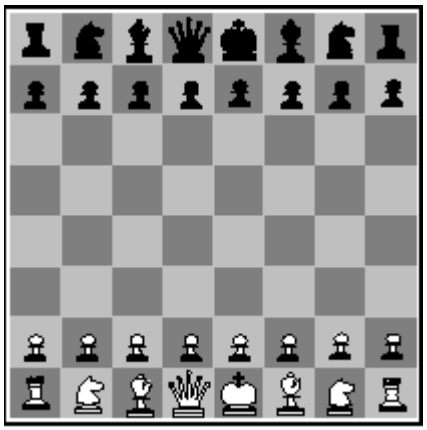
  where f is some feature and w is a weighting that biases the contribution of that feature to the final score.

When to Cut Search

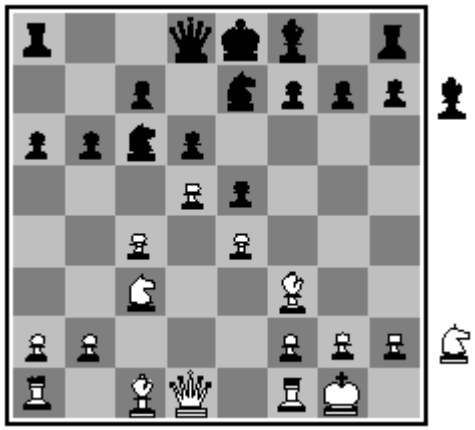
- **Depth Limit**: don't search beyond a given depth.

- **Iterative Deepening**: Stop search after a given time (i.e. time is up when resources run out).
  Assume:
  (1) 1000 positions/second
  (2) Branching Factor 35
  (3) 150 seconds/move (tournament play)
  $N = B^d$
  Max effective ply lookahead = 3-4

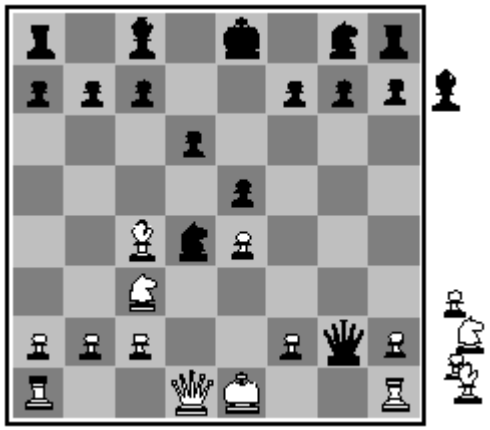
Chess Positions



(a) White to move
Fairly even

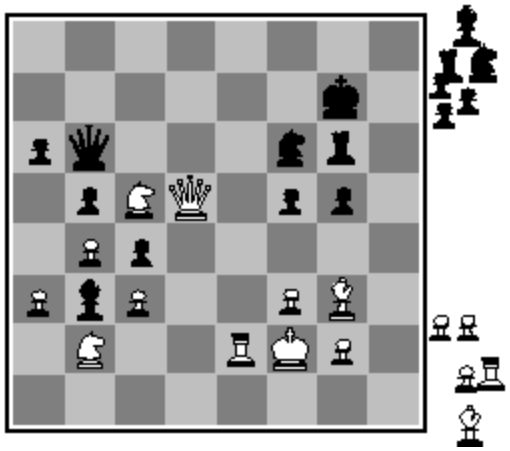


(b) Black to move
White slightly better

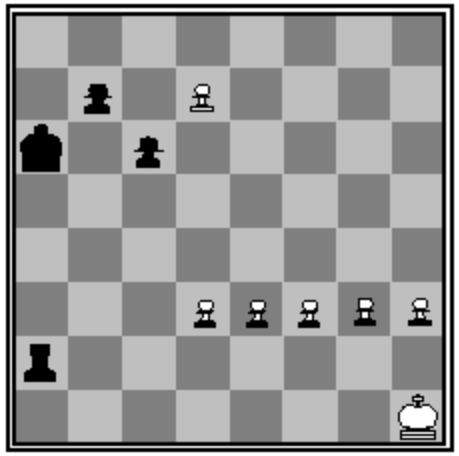


(c) White to move
Black winning

Chess Positions (cont)



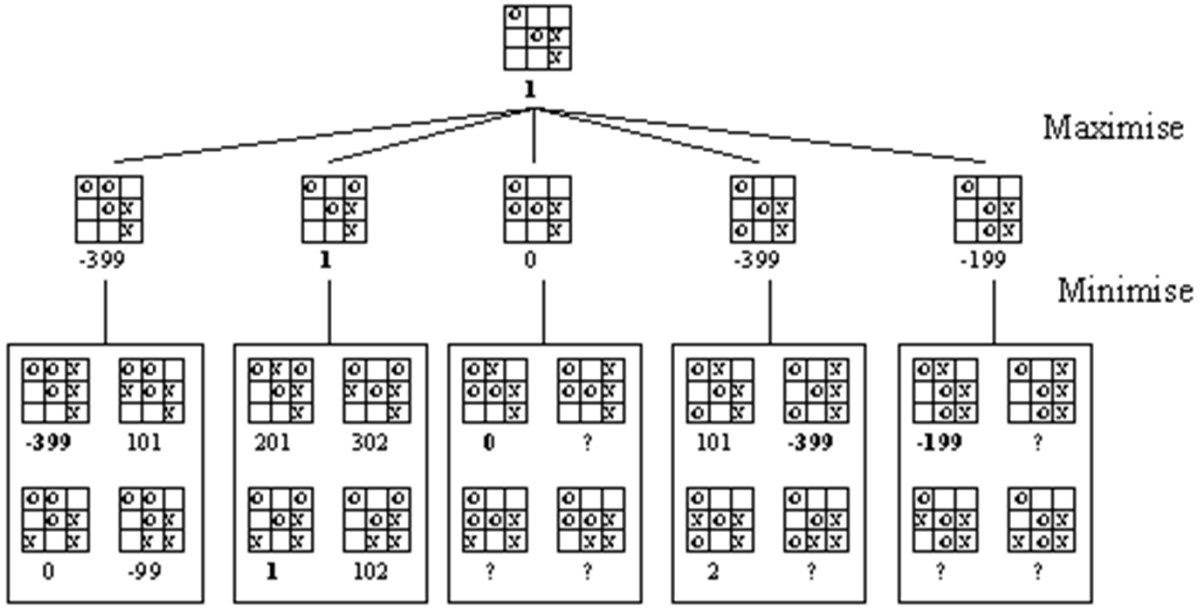
(d) Black to move
White about to lose



(d) Black to move

Alpha - Beta Pruning

- **Alpha-beta pruning** is a technique that enables the Minimax algorithm **to ignore branches** that will **not contribute further** to the outcome of the search.



State of the Art in Checkers

- 1952: Samuel developed a checkers program that **learn** its own evaluation function through self play.
- 1992: Chinook (J. Schaeffer) wins the U.S. Open. At the world championship, Marion Tinsley beat Chinook. **Alpha-beta search** were used.

State of the Art in Backgammon

- The inclusion of uncertainty from dice rolls makes search an expensive luxury.
- 1980: BKG using oneply search and **lots of luck** defeated the human world champion.
- 1992: Tesauro combines Samuel's **learning** method with neural networks to develop a new evaluation function, resulting in a program ranked among the top 3 players in the world.

State of the Art in Go

- The branching factor approaches *360*, **infeasible** for regular search methods.
- $2,000,000 prize available for first computer program to defeat a top level player.
- Go seems like an area likely to benefit from intensive research using sophisticated reasoning methods

State of the Art in Chess

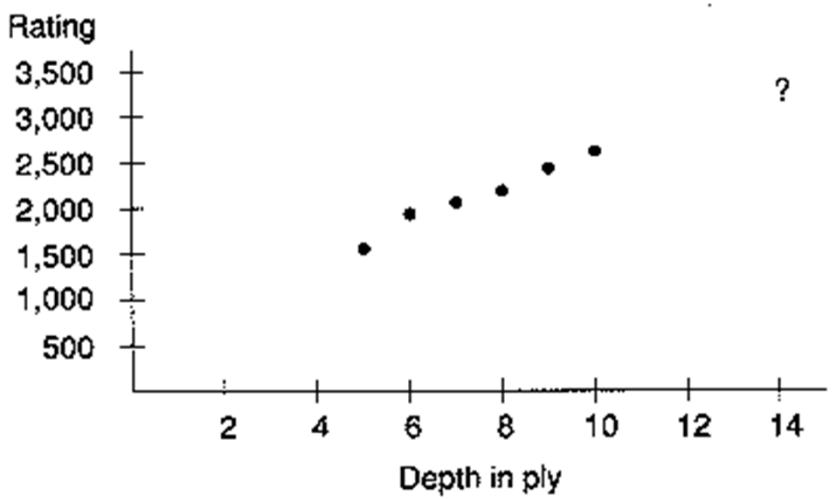
- Chess has received by far the **largest share of attention** in game playing.
- Progress beyond a mediocre level was initially very slow
- 1970: First program to win ACM North American Computer Chess Championship, it uses straightforward **alpha-beta search**, augmented with book **openings** and infallible **end game** algorithms.

State of the Art in Chess (cont)

- 1982: Belle, first **special-purpose chess computer**, can search *few million* of position per move.
- 1985: Hitech, ranked among the top 800 human players around the world, can search *10 million* of position per move.
- 1993: Deep Thought 2, ranked top 100 human players, can search *500 million* of position per move, reaching a depth of 11.

State of the Art in Chess (cont)

- 1997: Deep Blue, ranked top human player? Can search *1 billion* of position per move, reaching a depth of 14. Beat Chess master Gary Kasparov during the last match.



Program Strength Graph, as measured by the US Chess Federation's rating scale. Gary Kasparov is rated at 2600.

Class Activity 2: Real-world Paper Reading

"The end of an era, the beginning of another? HAL, Deep Blue and Kasparov "

- Public Perception of Artificial Intelligence (AI)
- The Chess Turing Test
- Should We Mimic Humans to Achieve AI?
- A New Era?
- Conclusion
- References

Source: http://www.chess.ibm.com/learn/html/e.8.1.html

Class Activity 3: Real-world Paper Reading

"Smart Moves: Intelligent Pathfinding"

- Pathfinding on the move
- Looking before you leap
- The star of the search algorithm (A* Search)
- How do I use A*
- The Limitation of A*
- Transforming the search space
- Storing it better
- Fine-tuning your search engine
- What if I'm in a smooth world

Source: http://www.gamasutra.com/features/programming/080197/pathfinding.htm

Students' Mini Research Presentation by Group B

What's in Store for Lecture 6

- Knowledge Representation
- Class Activity: One paper on Knowledge Representation - Reading

End of Lecture 5

Good Night.