Lecture #27: Games and Adversary Search
David Mix Barrington
31 March 2014
Games and Adversary Search

- Review: A* Search
- Modeling Two-Player Games
- When There is a Game Tree
- The Determinacy Theorem
- Searching a Game Tree
- Examples of Games
Review: A* Search

- The A* Search depends on a **heuristic function**, which is a **lower bound** on the distance to the goal.
- If $x$ is a node, and $g$ is the nearest goal node to $x$, the **admissibility condition** on $h$ is that $0 \leq h(x) \leq d(x, g)$.

![Diagram of A* Search](image)
**Review: A* Search**

- Suppose we have taken $y$ off of the open list. The best-path distance from the start $s$ to the goal $g$ through $y$ is $d(s, y) + d(y, g)$, and this cannot be less than $d(s, y) + h(y)$.

- Thus when we find a path of length $k$ from $s$ to $y$, we put $y$ onto the open list with priority $k + h(y)$. We still record the distance $d(s, y)$ when we take $y$ off of the open list.
Review: A* Search

- The advantage of A* over uniform-cost search is that we do not consider entries x in the closed list for which $d(s, x) + h(x)$ is greater than the actual best-path distance from s to g.

- This is because when we find the best path to g with length $d(s, g)$, we will put g on the open list with priority $d(s, g) + h(g) = d(s, g)$ and it will come off before any node with higher priority value.
The 15 Puzzle

- The 15-puzzle is a $4 \times 4$ grid of pieces with one missing, and the goal is to put them in a certain arrangement by repeatedly sliding a piece into the hole.

- We can imagine a graph where nodes are positions and edges represent legal moves.
The 15 Puzzle

• In order to move from a given position to the goal, each piece must move at least the Manhattan distance from its current position to its goal position.

• The sum of all these Manhattan distances gives us an admissible, consistent heuristic for the actual minimum number of moves to reach the goal. So an $A^*$ search will be faster than a uniform-cost search.
Clicker Question #1

• Suppose we do a uniform-cost search of the directed graph where nodes are 15-puzzle positions and there is a directed edge labeled “1” for every legal move. Which is not true?

• (a) This is the same as $A^*$ with $h(x) = 0$ always.
• (b) This is the same as DFS of the unlabeled graph.
• (c) This is the same as BFS of the unlabeled graph.
• (d) We’d do better with $A^*$ and a heuristic that counted the tiles that were out of place.
Answer #1

- Suppose we do a uniform-cost search of the directed graph where nodes are 15-puzzle positions and there is a directed edge labeled “1” for every legal move. Which is not true?
  - (a) This is the same as A* with $h(x) = 0$ always.
  - (b) This is the same as DFS of the unlabeled graph.
  - (c) This is the same as BFS of the unlabeled graph.
  - (d) We’d do better with A* and a heuristic that counted the tiles that were out of place.
Modeling Two-Player Games

- There are many kinds of games, and we are now going to look at a theory which will let us model and analyze some of them.

- You probably know that the game of **tic-tac-toe** is not very interesting to play, because if both players are familiar with the game the result is always a draw.

- There is a strategy for the first player, X, that allows her to always win or draw. There is also a strategy for O, the second player, letting him win or draw. If both players play these strategies, there is a draw.
Modeling Two-Player Games

• Any game that shares certain particular features of tic-tac-toe is **determined** in the same way.

• We must have **sequential moves, two players**, a **deterministic** game with no randomness, a **zero-sum** game, and **perfect information**.

• In these cases we can model the game by a **game tree**.
Game Trees

- A game tree has a node for every possible **state** or **position** of the game. The root node represents the **start position**.

- A node $y$ is a child of a node $x$ if it is possible, according to the rules of the game, to get to $y$ from $x$ in one move.
Game Trees

- Every node is labelled by whose turn it is.
- Usually the two players alternate moves, so we can call them the **first** and **second** player (White and Black), but our analysis will not change if one player can make several moves in a row.
Game Trees

- The **leaves** of the tree represent positions where the **result** of the game is known.

- We label leaves with a real number indicating how much White is paid by Black, typically 1 for a White win, 0 for a draw, and -1 for a Black win, but any real number values are possible.
Clicker Question #2

• Who wins the game represented to the right, if Black plays optimally?
  
• (a) White can win if and only if she first moves left.

• (b) White can win if and only if she first moves right.

• (c) White can win with either first move.

• (d) Black always wins.
Answer #2

• Who wins the game represented to the right, if Black plays optimally?

• (a) White can win if and only if she first moves left.

• (b) White can win if and only if she first moves right.

• (c) White can win with either first move.

• (d) Black always wins.
When We Have a Game Tree

- To be represented by such a tree the game must be discrete, deterministic, zero-sum, and have perfect information.

- The tree is finite if there are only finitely many sequences of moves that can ever occur. We could have a finite game graph where nodes can be reached in more than one way or even revisited, but we won't analyze these here.
The Determinacy Theorem

- Each leaf has a **game value**, the real number we defined above. We can inductively assign a game value to every node of the tree, by the following rules.

- The value val(s) of a final position is its label.

- If White is to move in position s, val(s) is the maximum value of any child of s.

- If Black is to move in position s, val(s) is the minimum value of any child of s.
The Determinacy Theorem

- The **Determinacy Theorem** says that:
  - (1) any game given by a finite tree has a game value $v$ (the value of the root given by the definition above),
  - (2) White has a strategy that guarantees her a result of *at least* $v$, and
  - (3) Black has a strategy that guarantees him that the result will be *at most* $v$. Thus $v$ is the result if both players play *optimally*. 
Proving Determinacy

• We prove that for each node $x$ in the tree, each player has a strategy that gets them either a result of $\text{val}(x)$ or a result that is even better for them.

• If $x$ is a leaf of the tree this is obvious.

• If it is White’s move she can move to the child with value $\text{val}(x)$, and by the IH get at least this result.

• It’s just the same if Black is to move.
Clicker Question #3

• What is the value of the game represented to the right?
  • (a) -2
  • (b) 0
  • (c) 3
  • (d) 7
Answer #3

- What is the value of the game represented to the right?
  - (a) -2
  - (b) 0
  - (c) 3
  - (d) 7
Winning Tic-Tac-Toe

- The chart to the right, if it were big enough to read, would tell you **complete strategies** for each player guaranteeing a result of 0 (a draw) or better.
Winning Tic-Tac-Toe

• The X strategy starts with moving to the top left, then has a reply to each of the eight O moves that could follow, then a reply to each of the six possible O responses to that move, and so on.

• The desired moves are in red.
Winning Tic-Tac-Toe

- The O strategy must have responses to all nine initial X moves, then to all seven X responses to each of those moves, and so on.

- The messiest parts of the chart is where the game goes for all nine moves, since each board is 1/9 the area of the last.
Searching a Game Tree

- The Determinacy Theorem only tells us that these optimal strategies exist, not that they are possible to implement.

- If it is possible to calculate the game value of any node, then choosing the right move is easy. And we have a recursive algorithm to compute the game value, so what is the problem?

- The tree could be really really big.
Adversary Search

- An exhaustive adversary search computes the exact value.
- If we can’t do that, we need an estimate of the game value.
- In Chess, for example, we can evaluate material and some positional facts to get a good idea whether one position is better than another.
Adversary Search

• We can then use **finite lookahead**, playing a game that ends in \( k \) moves, where the payoff is the estimated value of the position at the end of those \( k \) moves.

• **Alpha-beta pruning**, which we won’t do in this course, is a way to improve the search. But the required time is still usually exponential in the number of moves to go.
Examples of Games

• In 1965, my father’s M.S. thesis was to build a tic-tac-toe program that learned from its mistakes.

• By 1992, and probably earlier, students in CMPSCI 187 could build a winning program that exhaustively searched the game tree on every move.
Examples of Games

- There is either a winning strategy for White in Chess, or a drawing strategy for Black. But no one knows which is true.
- Current Chess programs succeed by doing a better job of searching and evaluating positions.
Examples of Games

- Computers don’t approach chess the way good human players do. We can use games as benchmarks for AI achievement.
- Checkers is easier than Chess, and Go is harder.
- Calvinball (from Calvin and Hobbes) allows rules to be changed at will.