R15/Ishank Juneja/16D070012

March 22, 2020

The Game of Dots-And-Boxes

Dots-and-boxes (DnB) is a classic combinatorial game played by people around the world. The setup for the two-player game consists of a rectangular grid of dots. Players take turns drawing horizontal or vertical lines between adjacent pairs of dots, forming boxes. When a player forms a new box they "capture" the box and add it to their box tally. At the end of the game, the player with the higher box tally wins.
The paper by Barker and Korf presents a strategy to efficiently solve the Dots-And-Boxes domain for an arbitrary (but small enough) grid. In the paper, *solving* a problem in the DnB domain entails computing the margin of boxes by which a player wins provided they follow an optimal strategy.
From the point of view of player moves, DnB is not very complex, however the size of its state-space is quite large. In particular the number of unique states - without considering player scores - is given by 2^p, where p is the number of possible edges. It is enough to solve DnB for these many unscored states since the scores of the players do not effect the optimal strategy at any point of time. What is worse is that a simple depth-first-search (DFS) starting from the empty state would generate $p!$ states on this problem, many of the generated states being duplicate states reached using an alternate sequence of steps.

Solution using retrograde analysis

Retrograde Analysis involves solving any given game state by working back from the unique terminal state (all edges filled) towards the given state. Using this technique the value of a given state can be computed using the pre-computed values of the states successors.
The solver starts by determining the value of every state with exactly one edge unfilled. The algorithm keeps proceeding in this manner until it reaches the start state with zero filled edges. Once the start state is reached, the value associated with every state is known. The merit of this procedure is that only 2^p states are parsed instead of the $p!$ states in a simple DFS. However, the 2^p dependence is still quite large. In this paper, the authors present a substantially more efficient solution based on Alpha-Beta minimax search.

Alpha-Beta minimax search with domain specific heuristics

Alpha-Beta minimax search is a DFS of the search space which maintains upper and lower bounds on the possible values of subtrees that could (potentially) affect the minimax value of the root node. The framework is useful since any subtree known to be lying outside this feasibility range can be eliminated without completely exploring it. Thus, Alpha-Beta avoids exploring the entire search space. There are however, other forms of redundancy created by both the problem structure and the nature of DFS. The method presented in the paper further exploits the following tools and results to boost performance-

- **Chains:** For certain states containing a chain like structure of edges, there are only two moves that can possibly be optimal. These are to either capture all available boxes or to spare a certain minimum number of boxes to avoid the opponent taking a lot of boxes in their very next move. This insight allows us to explore substantially fewer nodes at the cost of more expensive node evaluations.
- **Transposition Table (TT):** DFS by itself does not solve the problem of duplicate evaluations of identical nodes. Referring to a cached table that associates with each previously explored state its minimax value helps alleviate this problem.
- **Symmetries:** The authors also exploit all possible symmetries in the DnB problem. These are - horizontal, vertical, diagonal (square grids only) and corner symmetries.
- **Move ordering:** Some computational advantage can also be gained from choosing a good order for exploring child nodes. Based on the properties already exploited by the other heuristics, move ordering decides on exploring edges in an order radiating outwards from the centre of the board.

Based on extensive computational experiments presented in the paper, we can conclude that the solution to DnB presented in the paper is quite a significant advancement over its predecessors. In particular, its computational efficiency also makes the 4×5 DnB problem solvable within reasonable time. In my opinion, the paper is a great example of the application of domain knowledge in improving computational efficiency.

Some questions I have are -

- Why is the move-ordering heuristic of picking moves radiating outwards first good in practice?
- The corner symmetry property proof using graph isomorphisms is not very clear to me.