Vanessa Le

Professor Brooks

CS 386-02

20 September 2024

Assignment 2

QUESTION 1:

Algorithm	Time Complexity	Space Complexity	Complete?	Optimal?
BFS	O(bd+1) time, where b is the branching factor, and d is the depth of the solution.	O(bd+1) space	Yes. BFS is guaranteed to find a solution.	Yes, depending on edge costs. Assuming that all actions have uniform cost. BFS will find the shallowest solution in the search tree.
UCS	O(b(1+C/ε)) C* is the total cost of the optimal solution, ϵ is the smallest step cost (i.e., the minimum cost between any two connected nodes)	O(b(1+C/ε)) C* is the total cost of the optimal solution, ϵ is the smallest step cost (i.e., the minimum cost between any two connected nodes)	UCS = complete if the branching, b, is finite.	Yes
DFS	O(bn), where n is the depth of the search tree.	(O(bn)) space	On a finite graph, yes. If our search space is infinitely large, or we can't check for repeated states, no.	No. There is no guarantee that the first solution found will be the best.
DLS	O(b^d)	O(b^d)	Yes	Yes
IDS	O(bd)	O(bd)	Has the same optimality and completeness	Can extend the idea of depth-limited

			guarantees as BFS, but with better space requirements.	search to regain optimality, but keep DFS' linear memory needs.
A*	g(n) + h(n), where g is the cost so far and h is the estimate to the goal.	g(n) + h(n), where g is the cost so far and h is the estimate to the goal.	Under certain conditions, A* is complete	A* is optimal (always finds the best solution) in cases where our heuristic is admissible.

QUESTION 2:

a) What were the engineering advances that led to Deep Blue's success? Which of them can be transferred to other problems, and which are specific to chess?

- In the paper provided, there were many factors that led to Deep Blue's success: a single-chip chess search engine, a massively parallel system with multiple levels of parallelism, a strong emphasis on search extensions, a complex evaluation function, and effective use of a Grandmaster game database. Talking about the first three, firstly, those single-chips were designed to speedily search through potential chess moves, which was basically a chess search engine capable of evaluating chess positions extremely quickly. Secondly, Deep Blue distributed the workload across many processors, and was able to significantly reduce the time it took to search the game tree and make decisions. Thirdly, search extensions let Deep Blue to look into complex positions, thus improving its ability to find solutions in critical moments of the game.
- Massively parallel systems can be transferred to other problems since they are useful in a wide range of fields, such as machine learning. Search techniques are applicable in any decision-making problem with a state-space or tree structure, such as pathfinding and optimization. Heuristic evaluation functions are widely used in AI and optimization to guide decision-making.
- The chess search engine chips would not apply directly to other domains without significant modification. Additionally, the grandmaster game databases are purely

chess-specific, and while the concept of precomputed databases is transferable, the contents are unique to chess.

b) AlphaZero is compared to a number of modern game-playing programs, such as StockFish, which work similarly to Deep Blue. The paper shows that AlphaZero is able to defeat StockFish even when it is given only 1/100 of the computing time. Why is that? Please frame your answer in terms of search and the number of nodes evaluated.

- Stockfish involves a large amount of blind searching, searching and evaluating many possible moves (nodes) that are mostly less relevant and less optimal to play. On the other hand, AlphaZero uses neural networks and is trained by self-play to search and evaluate positions more intelligently, which allows it to focus on positions (nodes) that are more likely to lead to winning outcomes, avoiding the need for brute-force evaluation of many positions like how Stockfish does it.