Alpha-Beta Pruning

COMP110 - Research journal

1806868

December 2, 2018

We will be looking into the paper An Analysis of Alpha-Beta pruning by D.E. Knuth and R.W. Moore and how it has had a lasting influence in the field of computing. This paper will investigate how Alpha-Beta influenced other tree search algorithms which lead to the world champion of chess getting beaten by the IMB's deep blue AI project.

1 Introduction

The paper by D.E Knuth and R.W Moore[1] investigates a technique called alpha-beta pruning that is generally used to speed up the tree search process which can be used to play games like chess. D.E Knuth and R.W Moore felt it was necessary to present a new expository account of the method as papers that came before such as Experiments With a Multipurpose, Theorem-Proving Heuristic Program [2] are somewhat vague and did not expose deep cutoffs. In another article by Slagle and Dixon [3] it clearly indicates the possibility of deep cutsoffs but the paper had to resort to a rather complicated description to explain the alpha-beta technique, as it seems difficult to communicate verbally or in conventional mathematical language.

D.E. Knuth and R.W. Moore found that it was superior to express the alpha-beta method in an algorithmic language (or pseudocode) [4]although to their knowledge at the time there where two papers published that expressed the alpha-beta method in an algorithmic language. One of which Elements of Combinatorial Computing [5, Section 3.3] wasnt even the full alpha-beta procedure, it was incapable of making deep cutoffs and the other by Dahl and Belsnes [6, Section 8.1] was presented using label parameters, making it hard to prove. Not only can Knuth and Moores paper be proven by induction it clearly demonstrates the branch-and-bound strategy (F1) and the alpha-beta strategy (F2) procedures [1] which allows for deep cut-offs, using an algorithmic language.

2 Alpha-Beta Influences

An Analysis of Alpha-Beta Pruning by D.E. Knuth and R.W. Moore has had an influence in computing since it was published.

2.1 Negascout

In the 1980 a new method of pruning called Scout was proposed by Pearl [7] which offered advantages over alpha-beta when searching large game trees [8]. The improved efficiency of Scout is due the fact that it bypasses a majority of node without evaluating them precisely. This is achieved by a boolean function that determines whether or not each visited node exceeds that of a previous computed reference value [9]. This results in fewer subtrees to be revisited to compute their precise minmax value.

Negascout was developed on basis of the alpha-beta algorithm and was named Negascout because of its relation to Scout [8]. Unlike Scout, Negascout uses the negamax approach that resembles that of the alpha-beta algorithm which often results in searching fewer nodes than scout because it bypasses the tree with two bounds (like alpha-beta) rather than using a Boolean function (like Scout). It is suggested in Reinefeld paper [8] that Negascout become more efficient than alpha-beta when searching large game trees.

2.2 SSS*

In Stockmans paper [4] it suggests that the alpha-beta can beaten In efficiency if subtrees of the game tree are traversed in parallel since alpha-beta explores the game tree from left-to-right and may have to do more work than is required when the optimal path is more towards the right of the tree. SSS* develops multiple paths in all regions of the game tree and results in gaining a better global perspective of the tree and can preform cutoffs where alpha-beta cannot [4][10]. The SSS* algorithm performs the tree search via state sapce search hence its name SSS* and never explores nodes that are ignored by alpha-beta. However it is suggested that SSS* can be 5 [11] to 10[12] times slower than alpha-beta due its high storage requirements and management cost [11][10], although it examines significantly less nodes than that of alpha-beta and results in it being impractical for real game playing programs.

Further research was done to address the performance issues of SSS* [13][10] and lead to the introduction of AB-SSS*. AB-SSS* is a reformulated version of SSS* based on alpha-beta which resolved the memory requirements. The solution was to extend alpha-beta to include the well-known transposition table [14][13, Section 3.1]. This provides faster access and efficient storage over standard SSS*, as long as the transposition table is larger enough to store the min or max solution trees as a minimum. alpha-beta still out preforms AB-SSS* when the transposition table is relatively small, however once storage usages reaches a critical level it generally out performances alpha-beta. But this can only be achived if AB-SSS* is given a reasonable amount of storage and enough memory is available to store the max solution tree.

3 Conclusion

Conclusion here please

References

[1] D. E. Knuth and R. W. Moor, "An analysis of alpha-beta pruning," *Artificial Intelligence*, vol. 6, no. 4, pp. 293 – 326, 1975. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0004370275900193

[2] J. R. Slagle and P. Bursky, "Experiments with a multipurpose, theorem-proving heuristic program," *J. ACM*, vol. 15, no. 1, pp. 85–99, Jan. 1968. [Online]. Available: http://doi.acm.org/10.1145/321439.321444

[3] J. R. Slagle and J. E. Dixon, "Experiments with some programs that search game trees," *J. ACM*, vol. 16, no. 2, pp. 189–207, Apr. 1969. [Online]. Available: http://doi.acm.org/10.1145/321510.321511

[4] G. C. Stockman, "A minimax algorithm better than alpha-beta?" *Artificial Intelligence*, vol. 12, no. 2, pp. 179–196, 1979.

[5] M. B. WELLS, "Chapter 4 - search and enumerationbacktrack programming," in *Elements of Combinatorial Computing*, M. B. WELLS, Ed. Pergamon, 1971, pp. 93 – 126. [Online]. Available: http://www.sciencedirect.com/science/article/pii/B9780080160917500089

[6] O.-J. Dahl and D. Belsnes, *Algoritmer og datastrukturer*. Lund: Studentlitteratur, 1973.

[7] J. Pearl, "Asymptotic properties of minimax trees and game-searching procedures," *Artificial Intelligence*, vol. 14, no. 2, pp. 113–138, 1980.

[8] A. Reinefeld, "An improvement of the scout tree-search algorithm," *ICCA Journal*, vol. 6, no. 4, pp. 4–14, 1983.

[9] J. Pearl, "Scout: A simple game-searching algorithm with proven optimal properties." in *AAAI*, 1980, pp. 143–145.

[10] A. Reinefeld, *A minimax algorithm faster than alpha-beta.* Citeseer, 1993.

[11] A. Muszycka and R. Shinghal, "An empirical comparison of pruning strategies in game trees," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-15, no. 3, pp. 389–399, May 1985.

[12] T. A. Marsland, A. Reinefeld, and J. Schaeffer, "Low overhead alternatives to sss," *Artificial Intelligence*, vol. 31, no. 2, pp. 185–199, 1987.

[13] A. Plaat, J. Schaeffer, W. Pijls, and A. de Bruin, "A minimax algorithm better than sss*," *Artificial Intelligence*, vol. 87, no. 1-2, pp. 255–293, 1996.

[14] T. A. Marsland and M. Campbell, "Parallel search of strongly ordered game trees," *ACM Computing Surveys (CSUR)*, vol. 14, no. 4, pp. 533–551, 1982.