Iterative Quantum Tree Search

Luís Tarrataca and Andreas Wichert

GAIPS/INESC-ID
Department of Computer Science, Instituto Superior Técnico
Technical University of Lisbon
Avenida Professor Cavaco Silva
2780-990 Porto Salvo, Portugal
{luis.tarrataca,andreas.wichert}@ist.utl.pt

Abstract. Tree search algorithms are employed in order to solve a recurrent class of problems in the field of artificial intelligence. In this type of problems, a solution is represented as a sequence of symbolic actions whose length is also referred to as depth. When the depth of the shallowest solution is unknown some form of iteration needs to be applied in order to determine an appropriate limit. In this work we extend an existing $O(\sqrt{b^d})$ method for quantum tree search in order to accommodate for the requirements of iterative quantum tree search. We prove that the computational complexity of our iterative proposal remains $O(\sqrt{b^d})$.

Key words: iterative, quantum search; tree search.

1 Introduction

When no efficient polynomial-time algorithm to a problem is known, the solution typically resides in analyzing the state space. In artificial intelligence problems, such a procedure usually consists of examining a graph with a tree-like structure. Such problems can be described by a tuple (S_i, S_g, A) representing finite sets of, respectively, initial states, goal states and actions whose application leads states in S_i to S_g. The solution to a problem is represented as a sequence of actions, or path, leading the initial state to a goal state.

As an example lets consider one of the most frequent problems approached in artificial intelligence, namely the missionaries and cannibals problem. In this problem three missionaries and three cannibals need to cross a river using a boat that holds only two people. In addition, the boat cannot cross the river empty. The missionaries will be eaten if they are outnumbered by the cannibals on either bank. Amarel proposed to solve the problem by depicting the current state of the system as a vector of dimension 3 representing, respectively, the numbers of missionaries, cannibals and boats on the initial bank of the river [1]. The initial state is represented by vector $< 3, 3, 1 >$ and the goal state by $< 0, 0, 1 >$. Each action is also represented through the same vector notation. Accordingly, the set of possible actions A is $\{< 1, 0, 1 >, < 2, 0, 1 >, < 0, 1, 1 >, <$

$0, 2, 1 >, < 1, 1, 1 >\}$. Manipulations to the overall system state are performed through addition and subtraction of the respective action vectors.

Solving the problem is simply a matter of performing a tree search with the initial state as the root, and alternating between addition and subtraction operations, as depicted in Figure 1. Each depth layer d is responsible for adding at most b^d nodes to the tree, where b is the branching factor resulting from the number of actions utilized. Each one of the b^d leaf nodes requires a unique path leading to them. Eventually, the goal state will be achieved through the sequence of actions $\{< 0, 2, 1 >, < 0, 1, 1 >, < 0, 2, 1 >, < 0, 1, 1 >, < 2, 0, 1 >, < 1, 1, 1 >, < 2, 0, 1 >, < 0, 1, 1 >, < 0, 2, 1 >, < 0, 1, 1 >, < 0, 2, 1 >\}$. Although this specific instance was restricted to three missionaries and three cannibals, the problem can be generalized to any number of missionaries and cannibals.

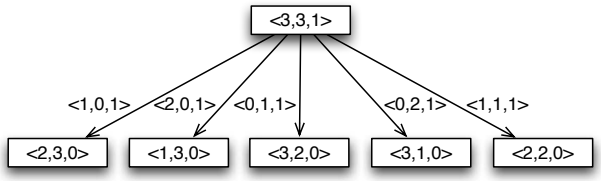

Fig. 1: Missionaries-Cannibals search tree with branching factor $b = 5$, depth $d = 1$.

The simplest tree search algorithms examine all possible sequences of moves until goal states are reached. If a non-goal state is discovered then the current state is expanded by applying admissible actions. This process enables the procedure to obtain a new set of states to analyse. The choice of which state to expand is determined by a search strategy. Strategies that can only distinguish between goal states and non-goal states, without being able to determine if one state is more promising than another, are referred to as uninformed search strategies. Alternatively, some problems allow the use of information in order to expand the most promising state. This is the method employed by informed and adversarial search strategies. Furthermore, the algorithms can choose to expand child nodes either up to a predetermined depth limit or iteratively increase the depth in order to find the best bound.

Typically, the computational complexity of tree search algorithms is represented as a function of the number of paths that need to be considered. For non-trivial branching factors and depths, performing an exhaustive examination of all possible sequence of actions with this type of search requires probing an exponential growth search space. Some of the best-know classical tree search procedures, alongside the respective time complexities are presented in Table 1, which emphasizes that the complexity of these algorithms grows exponentially fast and

has a form resembling $O(b^d)$. This characteristic complexity form is observed for the majority of these algorithms, with one notable exception namely that of the minimax algorithm. The original minimax formulation was based on zero-sum games, thus limiting its application to problems where no additional information can be employed in order to restrict the dimension of the search space.

Search	**Reference**	**Strategy**	**Time**
Depth-first	[2]	Uninformed	$O(b^m)$
Iterative-deepening	[3]	Uninformed	$O(b^d)$
Minimax	[4]	Informed/Adversarial	$O(b^m)$
Minimax $\alpha - \beta$ pruning	[5]	Informed/Adversarial	$O(\sqrt{b^m})$

Table 1: Tree Search Algorithm Comparison (b - branching factor, d - depth of a solution, m - maximum depth).

However, the advent of quantum computation has allowed for new search mechanisms that were previously unknown, and that are capable of producing important speedups. What follows in an analysis of some of the most important methods for quantum search and our proposition for a possible extension. Namely, the paper is organised as follows. Section 2 describes some of the main efforts focusing on performing tree search through quantum computation and respective complexity improvements. Section 3 describes an iterative extension to an existing methods. Section 4 elaborates on the computational complexity of the iterative version. We present the conclusions of this work in Section 5.

2 Quantum Tree Search

Tree search is closely related to graph transversal which, from a quantum computation perspective, has been approached through quantum random walks on graphs. Quantum random walks are the quantum equivalents of their classical counterparts ([6] provides an excellent introduction to the area). Quantum random walks were initially approached in [7], [8], [9] and [10] in one-dimensional terms, i.e. walk on a line. These concepts were then extended to quantum random walks on graphs in [11], [12], [13] and [14]. Quantum random walks can also provide a probabilistic speedup relatively to their classical parts, namely the hitting time for some specific graphs, *i.e.* the time it takes to reach a certain vertex B starting from a vertex A, can be shown to be exponentially smaller [15]. Other examples of quantum random walks include how to adapt the models to perform a search [16], [17], [18], and [19].

Additionally, tree search also has similarities with boolean constraint satisfaction problems. In [20] a continuous-time quantum walk was developed that showed how to compute the value of a balanced binary NAND tree in $O(\sqrt{N})$ time, where N is the number of leaf nodes. This result was subsequently mapped

into discrete-time in [21] and [22]. Ambainis developed a nearly optimal discrete query quantum algorithm for evaluating NAND formulas [23]. This result was later extender in order to prove that for any AND-OR formula of size N, there exists a bounded-error $N^{\frac{1}{2}+o(1)}$ time quantum algorithm, based on a discrete-time quantum walk [24]. These results provided the theoretical basis to show that it was possible to develop quantum algorithms for evaluating MIN-MAX trees with $N^{\frac{1}{2}+o(1)}$ queries. This procedure was done through a simple reduction of a MIN-MAX tree to an AND-OR tree [25].

However, most of these approaches focused on either graph transversal or the calculation of a value associated with the root node, whereas the central task in classical tree search consists in determining a path from the root node to a solution state, if one exists. An alternative approach was suggested in [26] which: (1) builds a superposition $|\psi\rangle$ of all possible sequences of actions up to a fixed depth limit d, as illustrated in Expression 1; and (2) employs Grover's algorithm in order to search through the state space of $|\psi\rangle$ alongside purposely designed oracles $O^d|q\rangle|a\rangle \rightarrow |q\rangle|a \oplus f(q)\rangle$, where $|q\rangle$ is a query register, $|a\rangle$ an answer register and $f(q)$ is a function responsible for outputting 1 if q leads to a goal state, and 0 otherwise (please refer to [27] and [28]). Expression 1 employs set A^d consisting of all the possible sequences of actions up to depth-level d. Each element of $|a\rangle \in A^d = |a_0, a_1, \cdots, a_d\rangle$ where a_k represents the action taken at depth level k, $\forall k \in [0, d]$. Accordingly, a measurement of $|\psi\rangle$ will yield a state containing the sequence of possible moves that lead to a solution. Since all possible paths are covered, and therefore all of the leaf nodes, this effectively means that set A^d has cardinality b^d. Consequently, through Grover's algorithm, the authors are able to achieve a running time of $O(\sqrt{b^d})$.

$$|\psi\rangle = \frac{1}{\sqrt{b^d}} \sum_{a \in A^d} |a\rangle \tag{1}$$

In this work we purpose a simple iterative extension to the quantum tree search method discussed in [26], [27] and [28]. The classical iterative-deepening procedure is as an optimal admissible form of tree search capable of finding the best depth limit [29]. In addition, classical iterative-deepening also preserves the original complexity, respectively $O(b^d)$. This procedure is carried out through a systematic increase of the search of the depth until a limit d, representing the depth of the shallowest goal node [30]. What follows is a description of our algorithm accompanied by the analysis of its complexity.

3 Iterative Quantum Tree Search

In order to obtain the desired iterative behaviour we choose to enclose our procedure in a while loop, as illustrated in Algorithm 1. This way it becomes possible to systematically go through all possible depth values d alongside oracle O, *i.e.*

O_d, until a goal state is found. Since we need to evaluate if applying O_d leads to a solution (line 7) we can combine the oracle alongside Grover's iterate for a total of $\sqrt{b^d}$ times in order to evaluate a superposition of all the available sequences of actions up to depth-level d, *i.e.* A^d (line 8). After applying Grover's algorithm, we can perform a measurement M on the superposition, if the state ξ obtained is a goal state (line 9), then the computation can terminate since a solution was found at depth d (line 11). Notice that if goal states are present in the superposition, then Grover's amplitude amplification scheme allows for one of them to be obtained with a high probability. Otherwise, we need to expand the search depth to level $d+1$ (line 13) and repeat the process from the start. As a result, this procedure requires building a new superposition of actions A^{d+1} each time a solution was not found in A^d. Algorithm 1 does not guarantee that variable d will ever be found, *i.e.* the search may not terminate. Accordingly, the while loop will execute forever unless the state ξ in line 10, obtained after the measurement, is a goal state.

Algorithm 1 Iterative Quantum Tree Search

1:	$d \leftarrow 0$	
2:	$\xi \leftarrow \emptyset$	
3:	$\|s\rangle \leftarrow i$	
4:	**while** *true* **do**	
5:	$\|q\rangle \leftarrow \frac{1}{\sqrt{b^d}} \sum_{\forall x \in A^d} \|x\rangle$	▷ Build superposition of actions
6:	$\|a\rangle \leftarrow \frac{1}{\sqrt{2}} (\|0\rangle - \|1\rangle)$	
7:	$\|\psi_1\rangle \leftarrow O_d \|q\rangle \|a\rangle$	▷ Mark if goal states exist at depth d
8:	$\|\psi_2\rangle \leftarrow G^{\sqrt{b^d}} \|\psi_1\rangle$	▷ Apply Grover's iterate
9:	$\xi \leftarrow M \|\psi_2\rangle$	▷ Measure the superposition
10:	**if** $\xi \in S_g$	
11:	**return** ξ	▷ If a goal state was found terminate
12:	**else**	
13:	$d \leftarrow d + 1$	▷ Otherwise, continue searching

4 Complexity Analysis

Quantum iterative deepening search may seem wasteful, because each time we apply O^d to a superposition spanning A^d we are necessarily evaluating the states belonging to previous depth levels multiple times, $\forall d > 0$. However, the bulk of the computational effort comes from the dimension of the search space to consider, respectively b^d, which grows exponentially fast. As pointed out in [29] if the branching factor of a search tree remains relatively constant then the majority of the nodes will be in the bottom level. This is a consequence of each additional level of depth adding an exponentially greater number of nodes. As a

result, the impact on performance of having to search multiple times the upper levels is minimal.

This argument can be stated algebraically by analysing the individual time complexities associated with each application of Grover's algorithm for the various depth levels. This argument is illustrated in Expression 2 which gives an overall time complexity of $O(\sqrt{b^d})$ remaining essentially unchanged from that of the original quantum tree search algorithm discussed in [26].

$$\sqrt{b^0} + \sqrt{b^1} + \sqrt{b^2} + \cdots + \sqrt{b^d} = O(\sqrt{b^d}) \tag{2}$$

Our proposal is therefore able to perform tree search with an inherent quadratic speedup relatively to its classical counterparts. This speedup can be perceived in two ways namely either as cutting the depth factor in half or as reducing the original branching factor from b to $\sqrt{b}$. Notice that this speedup is only obtained when the space has a branching factor of at least 2, otherwise the search is not influenced by the parallelism provided by quantum computation.

In addition, employing Grover's algorithm requires some sort of binary encoding mechanism so that the states present in the superposition $|\psi\rangle$ analysed map into adequate sequences of actions. Depending on specific conditions of the search tree being considered, namely the average branching factor and the maximum branching factor, respectively b_{avg} and b_{max}, it becomes possibles to determine if a significant portion of the states belonging to $|\psi\rangle$ maps into an admissible path. In [26] it was shown that deciding on whether or not this version of quantum tree search outperforms classical methods can be expressed as a function of b_{avg} and b_{max}. More concretely, classical tree search will deliver better performance when $b_{avg} \leq 2^{\frac{\lceil log_2 b_{max} \rceil}{2}}$ [26].

5 Conclusions

An analysis of the state space generated by classical search algorithms typically requires $O(b^d)$ time. Quantum tree search enables a quadratic improvement over its classical counterpart. However, some form of iteration needs to be applied when the depth of the shallowest solution is not known *a priori*. In this work we presented a simple iterative quantum tree search method based on an uninformed search strategy that can be perceived as the quantum counterpart of the classical iterative-deepening algorithm. Our approach combines Grover's algorithm alongside a set of measurements in order to deliver a performance of $O(\sqrt{b^d})$.

Acknowledgements

This work was supported by Fundação para a Ciência e Tecnologia (FCT) (INESC-ID multiannual funding) through the PIDDAC Program funds and FCT grant DFRH - SFRH/BD/61846/2009.

References

1. S. Amarel, *On representation of problems of reasoning about action.* Edinburgh University Press, 1971.
2. J. Hopcroft and R. Tarjan, "Algorithm 447: efficient algorithms for graph manipulation," *Commun. ACM*, vol. 16, no. 6, pp. 372–378, 1973.
3. D. Slate and L. R. Atkin, "Chess 4.5 - northwestern university chess program," in *Chess Skill in Man and Machine*, (Berlin), pp. 82–118, Springer-Verlag, 1977.
4. J. von Neumann, "Zur theorie der gesellschaftsspiele," *Mathematische Annalen*, vol. 100, no. 1, pp. 295–320, 1928.
5. T. Hart and D. J. Edwards, "The tree prune (tp) algorithm. artificial intelligence project memo 30," tech. rep., Massschusetts Institute of Technology, Cambridge, Massachusetts, 1961.
6. B. D. Hughes, *Random Walks and Random Environments*, vol. Volume 1: Random Walks. Oxford University Press, USA, 1995.
7. Y. Aharonov, L. Davidovich, and N. Zagury, "Quantum random walks," *Phys. Rev. A*, vol. 48, pp. 1687–1690, Aug 1993.
8. D. Meyer, "From quantum cellular automata to quantum lattice gases," *Journal of Statistical Physics*, vol. 85, pp. 551–574, 12 1996.
9. A. Nayak and A. Vishwanath, "Quantum walk on the line," tech. rep., DIMACS Technical Report, 2000.
10. A. Ambainis, E. Bach, A. Nayak, A. Vishwanath, and J. Watrous, "One-dimensional quantum walks," in *ACM Symposium on Theory of Computing*, pp. 37–49, 2001.
11. E. Farhi and S. Gutmann, "Quantum computation and decision trees," *Phys. Rev. A*, vol. 58, pp. 915–928, Aug 1998.
12. T. Hogg, "A framework for structured quantum search," *PHYSICA D*, vol. 120, p. 102, 1998.
13. D. Aharonov, A. Ambainis, J. Kempe, and U. Vazirani, "Quantum walks on graphs," in *Proceedings of ACM Symposium on Theory of Computation (STOC'01)*, pp. 50–59, July 2001.
14. A. M. Childs, E. Farhi, and S. Gutmann, "An example of the difference between quantum and classical random walks," *Quantum Information Processing*, vol. 1, no. 1, pp. 35–43, 2002.
15. A. M. Childs, R. Cleve, E. Deotto, E. Farhi, S. Gutmann, and D. Spielman, "Exponential algorithmic speedup by quantum walk," in *Proceedings of the 35th ACM Symposium on Theory of Computing (STOC 2003)*, pp. 59–68, Sept. 2003.
16. N. Shenvi, J. Kempe, and K. B. Whaley, "Quantum random-walk search algorithm," *Phys. Rev. A*, vol. 67, p. 052307, May 2003.
17. A. Ambainis, "Quantum search algorithms," *SIGACT News*, vol. 35, no. 2, pp. 22–35, 2004.
18. A. Ambainis, J. Kempe, and A. Rivosh, "Coins make quantum walks faster," 2005.

19. A. Ambainis, "Quantum walk algorithm for element distinctness," *SIAM Journal on Computing*, vol. 37, p. 210, 2007.
20. E. Farhi, J. Goldstone, and S. Gutmann, "A quantum algorithm for the hamiltonian nand tree," *Theory of Computing*, vol. 4, no. 1, pp. 169–190, 2008.
21. A. M. Childs, B. W. Reichardt, R. Spalek, and S. Zhang, "Every NAND formula of size N can be evaluated in time $N^{\frac{1}{2}+o(1)}$ on a quantum computer," *eprint arXiv:quant-ph/0703015*, Mar. 2007.
22. A. M. Childs, R. Cleve, S. P. Jordan, and D. Yonge-Mallo, "Discrete-query quantum algorithm for nand trees," *Theory of Computing*, vol. 5, no. 1, pp. 119–123, 2009.
23. A. Ambainis, "A nearly optimal discrete query quantum algorithm for evaluating NAND formulas," *ArXiv e-prints*, Apr. 2007.
24. A. Ambainis, A. Childs, and B. Reichardt, "Any and-or formula of size n can be evaluated in time $n^{\frac{1}{2}+o(1)}$ on a quantum computer," in *Foundations of Computer Science, 2007. FOCS '07. 48th Annual IEEE Symposium on*, pp. 363 –372, oct. 2007.
25. R. Cleve, D. Gavinsky, and D. Yonge-Mallo, "Quantum algorithms for evaluating min-max trees," in *Proceedings of Theory of Quantum Computation, Communication, and Cryptography (TQC 2008)* (Y. Kawano and M. Mosca, eds.), vol. 5106, pp. 11–15, Springer Berlin / Heidelberg, 2008.
26. L. Tarrataca and A. Wichert, "Tree search and quantum computation," *Quantum Information Processing*, vol. 10, no. 4, pp. 475–500, 2011. 10.1007/s11128-010-0212-z.
27. L. Tarrataca and A. Wichert, "Problem solving and quantum computation," *Cognitive Computation*, vol. 3, no. 4, pp. 510–524, 2011.
28. L. Tarrataca and A. Wichert, "A hierarchical sorting oracle," in *Proceedings of the Fifth International Quantum Interaction Symposium* (M. Melucci, D. Song, and I. Frommholz, eds.), 2011.
29. R. E. Korf, "Depth-first iterative-deepening : An optimal admissible tree search," *Artificial Intelligence*, vol. 27, no. 1, pp. 97 – 109, 1985.
30. S. J. Russell, P. Norvig, J. F. Canny, D. D. Edwards, J. M. Malik, and S. Thrun, *Artificial Intelligence: A Modern Approach (Second Edition)*. Prentice Hall, 2003.