Session 12A4

Mixed Strategy For Tree Search

S.S. Iyengar, Takahisa Miyata
Enamul Haq
Computer Science
Louisians State University
Baton Rouge, Louisiana

ABSTRACT

The standard methods for tree searching, such as breadth first and depth first searches suffer from space and time limitations respectively. The depth first iterative deepening tree searching method attempts to overcome the limitations of these standard methods of tree searching. However, this is not optimal in space.

A new method for searching tree that uses reduced amount of space is developed. The proposed method is a combination of breadth first, minimum space pebbling strategy and the modified iterative deepening depth first searches.

I. INTRODUCTION

Efficient searching of trees is fundamental to many areas of study, such as Artificial Intelligence, Operation Research, and Computer Science at large. Most Artificial Intelligence problems, especially game-playing, use heuristic information to direct the search that is relevant to the goal. Since the size of a decision or game tree encountered in such problems is very large, it becomes physically impossible to completely search the tree. Thus it becomes important to develop an efficient method for searching and/or pruning the tree. For more details see Knuth and Moore [1], Nilsson [2] , and Slagle and Dixon [3].

The methods widely used in searching trees are *breadth first* (BFS) and *depth first* (DFS) searches and both of them have serious limitations. For example, breadth first search requires large amounts of space, and depth first search requires too much time for searching and does not guarrantee the shortest path solution for a graph. Recently, Korf [4] demonstrated the generality of *depth first iterative deepening search* algorithm (DFID), proved its optimality in time for exponential tree searches, and further showed its application to different Artficial Intelligence problems. The depth first iterative deepening has some inherent disadvantages. In fact, DFID is optimal in time but not in space. The algorithms that trade time for reduced space are given in [5] and these algorithms are the modifications of the existing depth first, breadth first and heuristic tree searching algorithms.

In this paper, we propose a new mixed strategy of tree searching that is efficient in space. The proposed method is a combination of the breadth first, the *minimum space pebbling strategy* and the *modified depth first iterative deepening* algorithms (MDFID). The MDFID is optimal in space while retaining the time optimality.

We will analyze the complexity of the algorithms using three parameters : the depth (d) of the tree, the node branching factor (b), and the computational efficiency described as the total number of nodes generated during the search. The branching factor is the number of children of each node, averaged over the entire tree. The time cost of a search in a tree is the number of nodes that are examined during the search. The space cost is the number of nodes that need to be stored during the search .

The remainder of the paper is organized as follows : Section - II reviews the current search methods, section - III introduces the new modified depth first iterative deepening method and presents an analysis of the complexity, and section - IV concludes the paper.

CH2674-5/89/0000-0995$01.00©1989IEEE

II. CURRENT SEARCH METHODS

Several algorithms have been suggested over the years for doing efficient tree searching. In this section some of the well known existing tree searching methods are reviewed.

Breadth First Search (BFS)

The Breadth first search examines all the nodes one step away from the root node, then two steps away from the root node and so on until the goal node is reached. In the worst case, the BFS must examine all the nodes up to a depth d which can be described as follows:

$$T = b + b^2 + b^3 + \ldots + b^d = O(b^d) \qquad (2.1)$$

On the average, half of these nodes must be examined , leading to a time complexity of $O(b^d)$. Also since all nodes at the previous step must be stored to generate the nodes at the next step, BFS has a space complexity of $O(b^{d-1})$ which is $O(b^d)$.

Depth First Search (DFS)

The depth first search examines the descendants of the most recently examined node and continues until some cutoff depth is reached . Only the nodes on the path from the initial node to the current node need to be stored, and hence DFS has space complexity of $O(d)$ and time complexity of $O(b^d)$. Thus DFS is time bounded rather than space . It also has the problem that the improper selection of the arbitrary cutoff depth could result in the failure of the search.

Minimum Space Pebbling Strategy (MSPS)

In minimum space pebbling strategy, all the leaves in the tree can be pebbled by only one pebble that follows a unique path connecting the root to the leaf [5]. Once the single pebble used reaches a leaf, then to pebble the next leaf, it is required to start from the root and entirely repeat the same process. Thus the space required for this method is $O(1)$. However, this method suffers from the time complexity of $O(db^d)$.

Depth First Iterative Deepening Method

An algorithm that eliminates the drawbacks with earlier two techniques, as suggested by Korf [4] is now described. It is called Depth First Iterative Deepening Method (DFID) and can be described as follows:

1) First perform a depth first search to depth one.
2) If no solution is found, then discard all the nodes generated in the earlier search, do a depth first search to depth two, three and so on until the solution is found or the entire tree has been traversed or the cutoff depth has been reached.

Now we analyze the DFID by the three parameters mentioned earlier. The nodes at depth d are generated once during the final iteration of the search, the nodes at depth $(d-1)$ are generated twice and so on. Thus the total time, which is proportional to the total number of nodes generated, can be computed as follows:

$$T = b^d + 2b^{d-1} + \ldots + db \qquad (2.2)$$

which is less than $b^d(1 - 1/b)^{-2}$. However $(1 - 1/b)^{-2}$ is a constant, leading to a time complexity of $O(b^d)$. Since at any stage, DFID is doing a depth first search for one level at a time, the space complexity is $O(d)$ and the path to the solution is the shortest. However, it has the shortcoming that computations done at lower levels are wasted.

III. THE PROPOSED METHOD

The proposed method has three phases. In each of these three phases, an appropriate search is performed so that the computation and space requirements can be reduced.

PHASE I

Starting at the root of the search tree , perform BFS for the next $\log(\log d)$ levels. Let $k = \log d$, where d is the cutoff depth of the search. All logarithmic quantities used in this paper have base b (branching factor). The time complexity for this phase is

$$T_1 = \sum_{i=0}^{\log k} b^i = b + b^2 + \ldots + b^{\log(\log d)} = O(\log d) \qquad (3.1)$$

and the space required for this phase is

$$S_1 = k = O(\log d) \tag{3.2}$$

At the end of this phase the $\log d$ nodes at depth $\log k$ are stored. These are used to generate the lower level nodes and are needed until the entire search tree has been traversed.

PHASE II

If the goal has not been found in phase I, then DFID is executed from depth $(\log k + 1)$ down to depth $(\log k + \log d)$. This is repeated for all subtrees having their root at depth $\log k$. This phase is terminated when $\log d$ iterations have been carried out or a goal node is reached.

Since DFID is performed for $\log d$ subtrees each with $\log d$ levels. The time required for this phase is derived from (2.2) as follows. Let $Q = (\log d - 1)$.

$$\begin{aligned} T_2 &= k \sum_{i=0}^{k-1} (k-i) b^{i+1} = (\log d) \cdot \\ &\quad ((\log d) b + Q b^2 + \ldots + 2b^Q + b^{\log d}) \\ &= O(d \log d) \end{aligned} \tag{3.3}$$

Because it is required to store $\log d$ nodes at depth $\log k$ and another $\log d$ nodes for DFID, the space required for this phase is

$$S_2 = k + k = 2 \log d = O(\log d) \tag{3.4}$$

PHASE III

This phase is invoked if a goal node has not been found in phase I and II. The search technique used in this phase is called Modified Iterative Deepening Depth First search (MDFID). First, for each subtree having its root at depth $\log(\log d)$, perform MSPS down to $\log(\log d)+1$ and then DFS down to $\log(\log d) + \log d + 1$ for the $b^{\log(\log d)+1}$ subtrees. Then again for each subtree having its root at depth $\log(\log d)$, perform MSPS down to depth $\log(\log d) + 2$ and then DFS down to $\log(\log d) + \log d + 2$ for all the $b^{\log(\log d)+2}$ subtrees. This process is continued until either a goal node is reached or all the nodes at depth d are generated.

The time complexity of this phase is analyzed as follows. Let $P = (d - k - \log k + 1)$. Nodes examined for $\log d$ subtrees having their root at $\log(\log d)$ during minimum space pebbling from depth $\log(\log d)$ to depth :

$$\log\log d + 1 : \textit{nodes examined} = (2b^1) \log d$$

$$\log\log d + 2 : \textit{nodes examined} = (3b^2) \log d$$

$$\log\log d + (d - k - \log k) : \textit{nodes examined} = (Pb^{d-k-\log k}) \log d$$

Therefore the total nodes examined by the minimum space pebbling strategy is

$$T_s = k[2b + 3b^2 + \ldots + Pb^{d-k-\log k}] \tag{3.5}$$

At each stage DFS is performed for a depth of k. The number of nodes examined by DFS from depth $\log k + 1$ to $\log k + 1 + k$ is $(b + b^2 + \ldots + b^k) b^{\log k + 1}$. Therefore the total number of nodes examined by DFS, till it reaches the depth d is

$$\begin{aligned} T_d &= (b + b^2 + \ldots + b^k) \cdot \\ &\quad (b^{\log k+1} + b^{\log k+2} + \ldots + b^{d-k-\log k}) \end{aligned} \tag{3.6}$$

Thus the total nodes examined in phase III is given by

$$T_3 = T_d + T_s \tag{3.7}$$

Let $l = (d - \log k - k)$ and $p = (d - 2\log k - k)$. After some algebraic manipulation T_3 can be written as

$$\begin{aligned} T_3 = \frac{kb(b^k-1)}{(b-1)} \frac{b(b^p-1)}{(b-1)} + k[\frac{(l+1)b^{l+2}}{(b-1)^2} - \\ \frac{1}{(b-1)^2}\{(l+2)b^{l+1} + b^2 - 2b\}] \end{aligned} \tag{3.8}$$

Neglecting $(b^2 - 2b)$ from equation (3.8), we get

$$\begin{aligned} T_3 = \frac{kb^2}{(b-1)^2}(b^k-1)(b^p-1) + \\ \frac{k}{(b-1)^2}[(l+1)b^{l+2} - (l+2)b^{l+1}] \end{aligned} \tag{3.9}$$

Assuming $b^k >> 1$ and $b^p >> 1$, we get

$$\begin{aligned} T_3 = b^{d-\log k}[\frac{kb^2}{(b-1)^2 b^{\log k}} + \frac{kb^2}{(b-1)^2}\frac{(l+1)}{b^k} - \\ \frac{kb^2}{(b-1)^2}\frac{(l+2)}{bb^k}] \end{aligned} \tag{3.10}$$

Now, using the fact that $b^k = b^{\log d} = d$ in equation(3.10) T_3 can be further simplified as

$$T_3 = \frac{b^d b^2}{(b-1)^2}[\frac{1}{k} + \frac{(l+1)}{d} - \frac{(l+2)}{bd}]$$

Assuming $\frac{b^2}{(b-1)^2} \approx 1$, T_3 is written as

$$T_3 = b^d[\frac{1}{k} + \frac{(l+1)}{d} - \frac{(l+2)}{bd}] \qquad (3.11)$$

Therefore

$$T_3 = O(b^d) \qquad (3.12)$$

Since the $\log d$ nodes at a depth $\log(\log d)$ and another $\log d$ nodes for the performance of DFS must be stored, the space requirement for this phase is

$$S_3 = k + k = 2\log d = O(\log d) \qquad (3.13)$$

Based on the above analysis, we now state a theorem for the performance of our new method.

Theorem 3.1

For $k = \log d$, the total time and space required for this search are $T = O(b^d)$ and $S = O(\log d)$ respectively.

proof:

From equation (3.1),(3.3) and (3.12), the total number of nodes generated during the search can be expressed as follows:

$$T = T_1 + T_2 + T_3 = O(b^d)$$

Also from equation (3.2),(3.4) and (3.13), the total space required for this search is

$$S = O(\log d)$$

Performance Analysis of The Proposed Method

The performance of the proposed method over the existing tree searching methods is summarized in table (3.1). Carlson's method [5] of tree searching is also included in table (3.1) for the comparison of the performance of the proposed method. Carlson modified the standard DFS and BFS methods of tree searching by using the minimum space pebbling strategy with DFS and BFS. On the other hand, the proposed method is based on the modified iterative deepening depth first search. Even though the proposed method and Carlson's method have the same order of time and space requirements, the proposed method has the advantage that it finds the solution along the shortest path.

Table 3.1 - Performance of the proposed method compared with the existing methods

Method	Time	Space
Proposed Method	$O(b^d)$	$O(\log d)$
BFS	$O(b^d)$	$O(b^d)$
DFS	$O(b^d)$	$O(b^d)$
DFID	$O(b^d)$	$O(d)$
MSPS	$O(db^d)$	$O(1)$
Carlson's Method	$O(b^d)$	$O(\log d)$

IV. CONCLUSION

The proposed method of tree searching reduces the space requirements compared to the existing BFS, DFS and DFID methods, without increasing the order of the time complexity. However, there is a time space trade off inherent in this problem, since the strategies using minimum space require time greater than linear in the size of the tree. This method may be applied in any searching problem where BFS,DFS,DFID or any other standard searching methods is applicable.

REFERENCES

[1] Knuth,D. E. and Moore, R. W., An Analysis of Alpha-Beta Pruning, Artificial Intelligence, pp. 293-326, 1975.

[2] Nilsson, N. J., Principles of Artificial Intelligence, Tioga Publication Co., 1980.

[3] Slagle, J. R., and Dixon, J. D., Experiments with Soe Program that Search Game Trees, J. Assoc. Comput., March, pp. 189-207, 1969.

[4] Korf,R. E., Depth First Iterative Deepening : An Optimal Admissible Tree Search, Artificial Intelligence, Vol. 27, pp. 97-109, 1985.

[5] David A. Carlson,Time-Space Trade Offs for Tree Search and Traversal , IEEE, ch2345-7, 1986, pp. 585-594.