Journal of the Korean
OR/MS Society
Vol. 14 No. 1 June 1989

—Research Paper—

An AND-OR Graph Search Algorithm Under the Admissibility Condition Relaxed

Chae Y. Lee*

ABSTRACT

An algorithm that searches the general *AND-OR* graph is proposed. The convergence and the efficiency of the algorithm is examined and compared with an existing algorithm for the *AND-OR* graph. It is proved that the proposed algorithm is superior to the existing method both in the quality of the solution and the number of node expansions.

1. Introduction

The research on the *AND-OR* graph [2, 4, 5] has provided an important background for the problem solving and the knowledge representation in artificial inteligence. As an example, the behavior of a rule based system that works by problem decomposition can be represented by the *AND-OR* graph. The problem decomposition is performed by splitting the high-level goals into a series of subgoals that must be achieved. Each of the subgoals may have their own associated subgoals and so on. The nodes in the graph correspond to the states of working memory, while links correspond to possible rule applications.

In the literature on the *AND-OR* graph, it is customary to place certain restriction on the heuristic estimates of nodes in the search graph. Bagchi and Mahanti [1], however, compare several algorithms for searching OR graphs [1, 3, 5] that has no restriction on the heuristic estimates. They prove that the suggested algorithm is superior to the other ones when applied to the OR graph with no such restriction.

This paper provides a serach algorithm that works efficiently on the *AND-OR* graph that has no restriction on the heuristic estimates. The performance of the algorithm will also be examined by comparing it with an existing algorithm which is known to be one of the best.

2. *AND-OR* Graph Search Algorithms

Consider an *AND-OR* graph G in which each node of the graph represents a problem statement. A problem and its subproblems are linked by arcs pointing from the node representing

* Korea Institute of Technology

the problem to the nodes representing its subproblems. A Boolean function is employed in disjunctive normal form [3] to represent the relationship between a problem and its subproblems. A proposition N is denoted as the problem statement corresponding to a node n. An example of the *AND-OR* graph is shown in Figure 1. In the graph, the original problem S is decomposed into A, B and C. The relationship between the initial problem and its subproblems are expressed as follows :

$$\begin{aligned} S &= A \text{v} BC \\ &= D \text{v} (E \text{v} F) G \\ &= D \text{v} E (J \text{v} K) \text{v} F (J \text{v} K) \\ &= D \text{v} EJ \text{v} EK \text{v} HIJ \text{v} HIK \end{aligned}$$

For the development of the *AND-OR* graph search algorithms a directed graph G with q starting nodes $s_1, \cdots, s_q$ and a set of goal nodes is considered. Each arc in G is assumed to have a positive are cost c(m, n) and $c(m, n) \geq \delta > 0$, where δ is a given small positive real number. In the graph, we are interested in finding a minimum cost path from starting nodes to goal nodes. Associated with each node n in G nonnegative heuristic estimates are employed in the search process. Before introducting the heuristic estimates we consider the following definitions :

Definition 1 Let n be a node in an *AND-OR* graph. Suppose n is related to its immediate successor nodes by a Boolean function

$$N = C_1 \text{v} C_2 \text{v} \cdots \text{v} C_m ,$$

where C_i, $i = 1, \cdots, m$, are conjunctions of propositions. Then each C_i is called an *immediate implicant* of N.

Definition 2 Let a conjunction $Q = N_1 \cdots N_r$, where $r \geq 1$. Then Q' is said to be an *immediate implicant* of Q iff Q' is a conjunction obtained from Q by replacing an N_k by one of its immediate implicants, $1 \leq k \leq r$.

Definition 3 A conjunction Q is an *implicant* of a conjunction P iff there is a sequence of conjunctions $R_1, R_2, \cdots, R_n$ such that $P = R_1$ $Q = R_n$, and R_i is an immediate implicant of R_{i-1} for $i = 2, \cdots, n$.

The algorithm that will be presented in this paper is based upon an evaluation function $f(P)$ for each implicant P of S which is the initial problem associated with the Q starting nodes. This $f(P)$ can be written as $f(P) = g(P) + h(P)$, where if $P = N_1 \cdots N_r$, then $g(P)$ is the cost of a minimal path graph from $s_1, \cdots, s_q$ to the nodes $n_1, \cdots, n_r$, and $h(P)$ is the cost of a minimal solution graph started with $n_1, \cdots, n_r$. Since $f(P)$ is not known in general, we make use of an estimate $\hat{f}(P) = \hat{g}(P) + \hat{h}(P)$ of $f(P)$, where $\hat{g}(P)$ and $\hat{h}(P)$ are the estimates of $g(P)$ and $h(P)$.

In most of the previous research on the *AND-OR* graph, it is customary to assume that the heuristic estimate $\hat{h}$ satisfies the admissibility condition given below.

Definition 4 The heuristic estimate $\hat{h}$ is admissible if for every conjunction P in the search

graph G, $\hat{h}(P) \leq h(P)$.

In this paper, however, a general case of *AND-OR* graph is considered. In other words, a graph with no restriction on the heuristic estimates is taken into consideration. An algorithm by Chang and Slagle [2] and a modified version of the algorithm are now presented as follows :

AND-OR graph search Algorithm *AO* by Chang & Slagle [2]

Step 1 Let $W=\{S\}$, $R=\phi$ and $\hat{f}(S)=0$.

Step 2 Calculate $\hat{f}$ for each element in the set W. Select a P in W such that $\hat{f}(P)$ is smallest. Resolve ties arbitrarily, but always in favor of an element of W which is a conjunction of propositions associated with terminal nodes.

Step 3 Let $P=P_1 \cdots P_r$, where P_i is the proposition associated with the node p_i, $i=1, \cdots, r$. If $p_1, \cdots, p_r$ are terminal nodes, terminate *AO* ; a solution graph has been found. Otherwise, go to the next step.

Step 4 If P is already expanded, go to Step 6. Otherwise, go to the next step.

Step 5 Expand all the unexpanded non-terminal nodes of $p_1, \cdots, p_r$.

Step 6 Let V be the set of all the implicants of S constructed from $P=P_1 \cdots P_r$ by replacing each (non-terminal) P_i by one of its immediate implicants, $i=1, \cdots, r$. Let $R=R \cup \{P\}$.

Step 7 Let $W=(W \cup V)-R$. If W is empty, terminate *AO* ; there is no solution graph. Otherwise, go to Step 2.

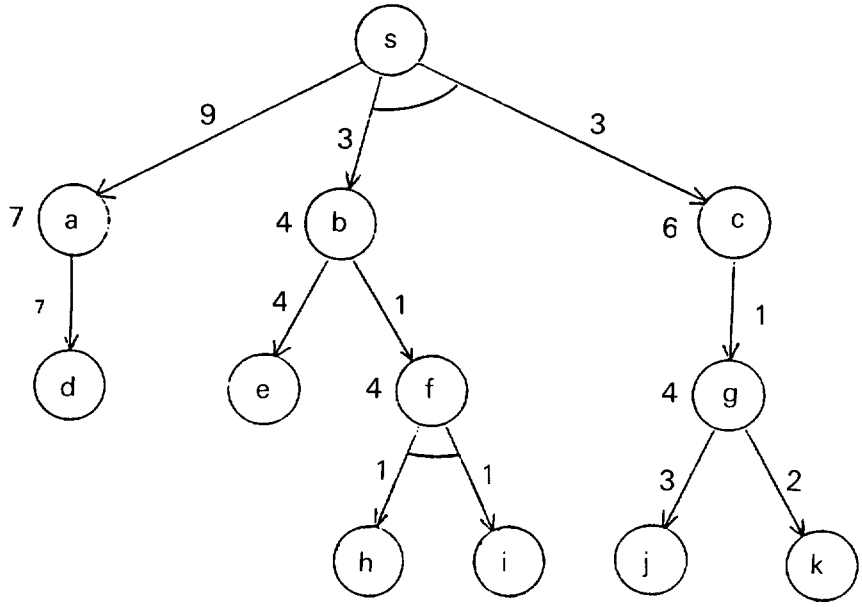


Figure 1

Algorithm *MAO* for the *AND-OR* graph

Same as the algorithm *AO* by Chang & Slagle, except for Step 1 and Step 2, which are to be replaced by the steps given below.

Step 1 Let $W=S$ and $R=\phi$ $\hat{f}(S)=0$, $F\leftarrow 0$.

Step 2 Calculate $\hat{f}(Q)$ for each element Q in W. If there are some implicants in W with $\hat{f}\leq F$, select among them a P such that $\hat{g}(P)$ is smallest. Otherwise, consider those implicants in W with minimal $\hat{f}$ value, select among them a P such that $\hat{g}(P)$ is smallest, and set $F\leftarrow\hat{f}(P)$.

Example 1 An *AND-OR* graph is shown in Figure 1 to illustrate the Algorithm *MAO*. In Figure 1 the number beside each node n is the estimated cost $\hat{h}(N)$ of $h(N)$ and the one beside each are is the arc cost. The estimated cost is defined as $\hat{h}(P)=\sum_{i}^{k}[c(P, N_i)+\hat{h}(N_i)]$ for $P=N_1 \cdots N_k$. Clearly, the admissibility condition is relaxed in the graph. We now describe how the algorithm is applied to obtain a minimal solution graph in the following :

(1) Exanding the node s, we obtain $W=\{A, BC\}$. From $\hat{f}(A)=\hat{g}(A)+\hat{h}(A)=9+7=16$ and $\hat{f}(BC)=\hat{g}(BC)+\hat{h}(BC)=6+10=16$, we choose FG for expansion since $\hat{g}(BC)<\hat{g}(A)$.

(2) Since $BC=(E+F)G=EG+FG$, we obtain $W=\{A, EG, FG\}$. We also have $\hat{f}(EG)=11+4=15$ and $\hat{f}(FG)=8+8=16$. Since $F=16$, we choose FG for expansion.

(3) Expanding FG, we obtain $FG=HI(J+K)=HIJ+HIK$. We have $W=\{A, EG, HIJ, HIK\}$. Since $\hat{f}(HIJ)=13+0=13$ and $\hat{f}(HIK)=12+0=12$, we choose EG for expansion whose $\hat{g}$ value is the smallest among the conjunctions with $\hat{f}\leq F$.

(4) Expanding EG, we have $EG=E(J+K)=EJ+EK$ and $W=\{A, EJ, EK, HIJ, HIK\}$. Since $\hat{f}(EJ)=14+0=14$ and $\hat{f}(EK)=13+0=13$, we choose A.

(5) By expanding A we have $W=\{D, EJ, EK, HIJ, HIK\}$. Since $\hat{f}(D)=16+0=16$ we choose HIK.

(6) We terminate the algorithm since h, i, and k are terminal nodes. The cost of the solution graph is 12 which is true minimal.

Note that the algorithm *AO* gives output 16 or 13 for the same search graph shown in Figure 1. Since algorithm *AO* chooses a conjunction with smallest $\hat{f}$ value at each iteration, it is terminated with D or EK. For the search graph Algorithm *MAO* gives better solution than Algorithm *AO*. Is it the case that the result of Algorithm *MAO* is always at least as good as that of Algorithm *AO*? We will show in Section 4 that it is indeed so.

3. Convergence of the Algorithm *MAO*

We here dicuss the convergence of the algorithm in the *AND-OR* graph where the admissibility condition is relaxed

Definition 5

(i) A Solution path is a path in an *AND-OR* graph G from the starting nodes $s_1, \cdots, s_q$ to the goal nodes.

(ii) Let A_1, A_2, $\cdots$ be the solution paths in G. We write $P \varepsilon A_i$ if nodes p_1, $\cdots$, p_r lies on the solution path A_i, for each i, let M_i be

$$M_i = \max_{p \& A_i} [C(A_i, P) + \hat{h}(P)]$$

where $c(A_i, P)$ is the cost of the path A_i from s_1, $\cdots$, s_q to p_1, $\cdots$, p_r.

(iii) Define T as follows:

$$T = \min_{i \geq 1} M_i$$

(iv) Let A_{i1}, A_{i2}, $\cdots$, A_{ik}, for some $k \geq 1$, be the minimal-cost solution paths in G. Define T_{opt} as follows:

$$T_{opt} = \min_{1 \leq j \leq k} M_{ij}$$

(v) The set of conjunctions Z is defined as follows:

(a) $S \varepsilon Z$.

(b) $P \varepsilon Z$, if there is a path A from s_1, $\cdots$, s_q to p_1, $\cdots$, p_r such that $c(A, P) + \hat{h}(P) \leq T$ and the immediate predecessor of p_1, $\cdots$, p_r on A is in Z.

(vi) Let N be the number of conjunctions in Z.

Example 2 The solution paths of the *AND-OR* graph in Figure 1 are

$A_1 = sad$,
$A_2 = s(be)\ (cgj)$,
$A_3 = s(be)\ (cgk)$,
$A_4 = s(bf(h)(i))\ (cgj)$,
and $A_5 = s(bf(h)(i))\ (cgk)$,

where the nodes in the parethesis are in the *AND* relationship. We obtain

$T = \min\{16, 16, 16, 17, 17\} = 16$
and $T_{opt} = 17$.

The value of T is determined by any one of the three paths A_1, A_2 and A_3 and that of T_{opt} is by the path A_5 which is the minimal cost solution path. We also have

$Z = \{A, D, BC, EG, EJ, EK, FG, HIJ, HIK\}$
and $N = |Z| = 9$.

Lemma 1 At any step before Algorithm *MAO* terminates there is a conjunction $P \varepsilon W$ such that $\hat{f}(P) \leq T$.

Proof At any step before Algorithm *MAO* terminates, W contains at least one conjunction from

every solution path. Consider a solution path A that determines the value of T. Then there is a conjunction P on A which is in W all of whose predecessors on A have already been expanded. Clearly, $\hat{g}(P) \leqq c(A, P)$ and $\hat{f}(P) = \hat{g}(P) + \hat{h}(P) \quad C(A, P) + \hat{h}(P) = T$. *So* $\hat{f}(P) \leqq T$.

Theorem 1 Algorithm *MAO* terminates successfully, that is, it finds a set of goal nodes.
Proof By Lemma 1 W always contains a conjunction P such that $\hat{f}(P) \leqq T$. Since we have assumed that the cost of an arc is $\geqq \delta$, where δ is a small positive real number, it follows that the search graph has only finitely many paths starting from S with cost $\leqq T$. As $\hat{h}$ values are nonnegative, by Lemmal 1 Algorithm *MAO* can not continue forever.

For the convergence of the Algorithm *AO* Chang and Slagle [2] proved that it terminates at a minimal solution graph under the assumption that the admissibility condition is satisfied. However, the convergence is not guaranteed when the admissibility condition is relaxed.

4. Efficiency of the Algorithm *MAO*

In this section we analyze and compare the performance of the two algorithms. For that purpose the following two factors are considered as in Bagchi and Mahanti [1] :

(i) Quality of the solution : Since neither of the algorithms may give the minimal cost solution, it would be of interest to know which one gives the best solution.

(ii) Number of node expansions : This measure is expressed in terms of N, since only nodes in Z get expanded.

Definition 6

(i) Let

$$S^{AO} = f^{AO}_1 \; f^{AO}_2 \; \cdots \; f^{AO}_a \qquad a \geqq 1.$$

be the time sequence of $\hat{f}$ values of nodes selected for expansion by Algorithm *AO*. Similarly, let

$$S^{MAO} = f^{MAO}_1 \; f^{MAO}_2 \; \cdots f^{MAO}_b \qquad b \geqq 1.$$

Note that $f^{AO}_a = f_t$ and $f^{MAO}_b = f_B$.

(ii) Let

$$F^{AO}_{max} = \max_{1 \leqq i \leqq a} f^{AO}_i ,$$

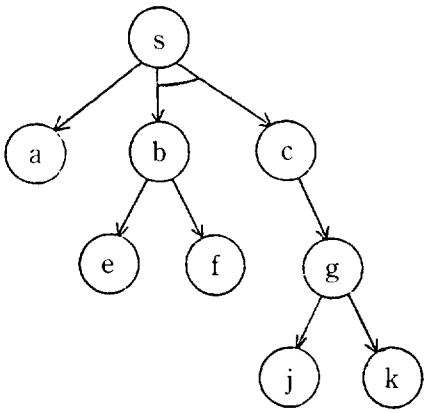
$$F^{MAO}_{max} = \max_{1 \leqq i \leqq b} f^{MAO}_i .$$

(iii) Let G^{AO}_i denote as much of the search graph G as has been expanded by Algorithm *AO* when it executes step 3 for the ith time. Each node in G^{AO}_i is assumed to have associated with it the following attributes :

(a) Its current $\hat{g}$ *value*, $\hat{h}$ value, and $\hat{f}$ value.
(b) *A* set W or R, indicating whether it is currently in the set W or R.

G^{AO}_{1} just has the single conjunction S and no arcs. The G^{AO}_{u+1} is used to represent the situation at termination, that is, after the goal node has been selected from W and put in R. Thus G^{AO}_{u+1} and G^{AO}_{u} differ only in that the goal node is in W in G^{AO}_{u} and in R in G^{AO}_{u+1}.

Example 3 Figure 2 represents G^{AO}_{1} for the search graph G of Figure 1.



Conjunction	$\hat{g}$	$\hat{h}$	$\hat{f}$	W or R
S	0	0	0	R
A	9	7	16	W
BC	6	10	16	R
EG	11	4	15	R
FG	8	8	16	W
EJ	14	0	14	W
EK	13	0	13	W

Figure 2

Definition 4

Let

$$S^{AO} = f^{AO}_{i1}\ f^{AO}_{i2}\ \cdots\ f^{AO}_{im}\ ,\quad u \geq 1,$$

be the strictly increasing subsequence of $\hat{f}$ values in S^{AO}. Similary, let

$$S^{MAO} = f^{MAO}_{j1}\ f^{MAO}_{j2}\ \cdots\ f^{MAO}_{jv}\ ,\quad v \geq 1,$$

be the strictly increasing subsequence of $\hat{f}$ values in S^{MAO}

Lemma 2 Every *AND-OR* graph can be transformed into a pure *OR* graph.

Proof Every node except the goal nodes in the *AND-OR* graph is represented by the Boolean function as the disjunctive normal form of the immediate implicants. Thus, the Lemma is true regardless of the estimates $\hat{g}(P)$ and $\hat{h}(P)$ in the *AND-OR* graph.

Example 4 The transformation of the *AND-OR* graph (see Figure 1) into a pure *OR* graph is illustrated in Figure 3. In the transformation the following estimates are used :

$$\hat{h}(P) = \sum_{i=1}^{k} \hat{h}(N_i)$$
$$\text{and } \hat{g}(P) = \sum_{i=1}^{k} \hat{g}(N_i)$$
$$\text{where } P = N_1 N_2 \cdots N_k$$

Theorem 2 Algorithm *AO* makes $O(2^N)$ node expantions at worst.

Proof Clear from Lemma 2 and Theorem 3.1 of Martelli [3].

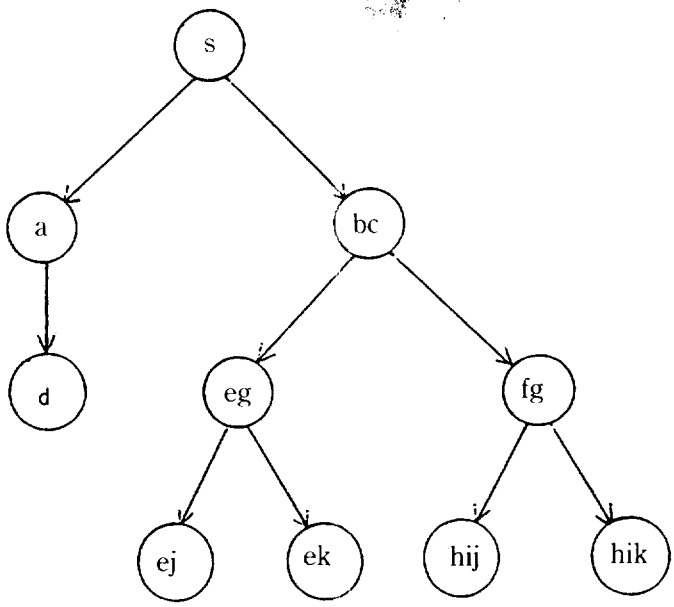


Figure 3

Theorem 3 Algorithm *MAO* makes $O(N^{\prime})$ node expantions at worst.

Proof Clear from Lemma 2 and Theorem 4.1 of Martelli [3] by considering strictly increasing $\hat{f}$ subsequence when $b=j_i$. When $b>j_i$, since we choose conjunction P with $\hat{g}$ value smallest among those with $\hat{f}\leq F$, the conjunction P cannot be reopened between the two conjunctions $\hat{f}_{i_k}$ and $\hat{f}_{i_{k+1}}$.

Theorem 4 Algorithm *AO* [3] never gives a solution of lower cost than that given by Algorithm *MAO* assuming identical resolution of ties.

Proof Clear from Lemma 2 and the proof of Theorem 3.5 of Bagchi and Mahanti [1]. Only observe that $\hat{f}\leq F^{*}{}_{min}$, since the time sequence of $\hat{f}$ values are strictly increasing by the Algorithm *AO*.

5. Conclusion

A modified version of the *AND-OR* graph search algorithm (Algorithm *MAO*) is presented. The algorithm deals with the general case of *AND-OR*. In other words, Algorithm *MAO* provides a good solution even if the heuristic estimates does not satisfy the admissibility condition of the usual *AND-OR* graph.

It is proved that the algorithm converges to a set of goal nodes even though it is not optimal. The performance of the algorithm is compared with a well-known *AND-OR* graph search algorithm (Algorithm *AO*). It is proved that the quality of the solution by *MAO* is at least as good as that by the Algorithm *AO*. It is also proved that Algorithm *MAO* makes only $O(N^{\prime})$ node expansions while Algorithm *AO* makes $O(2^{N})$ node expansions.

References

1. Bagchi *A.* and *A.* Mahanti, "Search Algorithms under Different kinds of Heuristics-A Comparitive Study", ***Journal of the Association for Computing Machinery 30***, 1983, 1-21.
2. Chang, C. L. and J. R. Slagle, "An admissible and Optimal Algorithm for Searching *AND-OR* Graphs", ***Artificial Intelligence 2***, 1971, 117-128.
3. Martelli, A., "On the Complexity of Admissible Search Algorithms", ***Artificial Intelligence 8***, 1977, 1-13.
4. Nau, D. S., V. Kumar and L. Kanal, "General Branch and Bound, and its Relation to A^* and AO^* ", ***Artifical Intelligence 23***, 1984, 29-58.
5. Nilsson, N. J., "Principles of Artificial Intelligence", Springer-Verlag, 1980.