THE COMPLEXITY OF SEARCHING SEVERAL CLASSES OF AND/OR GRAPHS

Howard E. Motteler
Laveen N. Kanal *
The Machine Intelligence and Pattern Analysis Laboratory
Department of Computer Science
University of Maryland
College Park, MD 20742

Abstract

The complexity of searching for a minimum cost solution graph of an AND/OR graph is analyzed for the class of AND/OR graphs representable by a context free grammar with cost functions; finding a minimum cost solution graph is then equivalent to finding a lowest cost derivation. Several classes of search problems are defined, based on properties of the cost functions and grammar. We show that certain of these classes have different search complexities- specifically, we show that there are distinct classes for which the complexity of finding a minimum cost solution graph is non-recursive, exponential, *NP*-complete, and $Q(n^2)$, where n is the size of the grammar representing the problem. The correspondence between problem structure and search complexity may serve as a guide for modeling real problems with AND/OR graphs.

1. Introduction

The complexity of A.I. search procedures is a topic of current interest [13,2,6]. In this paper we analyze the complexity of searching several classes of AND/OR graphs. The AND/OR graphs in which we are interested are represented by the *composite decision process* (CDP) [4,8]. This is a generalization of the sequential decision process (SDP) proposed by Karp and Held in [5]. The classes of AND/OR graphs represented by the composite decision process are a model for a wide variety of search procedures, problems in pattern recognition, and dynamic programming problems. Applications and examples of relevance to practical problems may be found in [4,8,9].

A CDP is defined by giving a context-free grammar, and a cost function associated with each production and terminal symbol. This is essentially Knuth's "synthesized attributes" **[7],** with a different motivation. The minimalization *problem* is to find a lowest cost parse tree for any string the grammar might derive. This is equivalent to the problem of finding a minimum cost solution graph of the AND/OR graph corresponding to the grammar [11].

The correspondence between AND/OR graphs and context free grammars is that connectors (the AND arcs) correspond to productions, and OR arcs correspond to a choice of productions. The simple **grammar and graph of figure 1 shows the** correspondence. **This** correspondence **was first** noted **by Hall** [3]; **following the treatment in** Nilsson, **nodes are not restricted** to **be only of** types **AND or OR** [12]. Our **AND/OR graphs** may contain **cycles, and in general a** solution **graph** may include **a walk around some cycle any number of times. Thinking in terms of grammars, a solution graph is just a parse tree; a solution graph may be much larger than the grammar graph, and there may be infinitely many parse/solution trees. Figure 1 gives two solution trees for the AND/OR graph shown there; note that the larger of the two solution trees has minimal cost.**

***Research supported by NSF grants to the Machine Intelligence and Pattern Analysis Lab., Dept. of Computer Science, University of Maryland.**

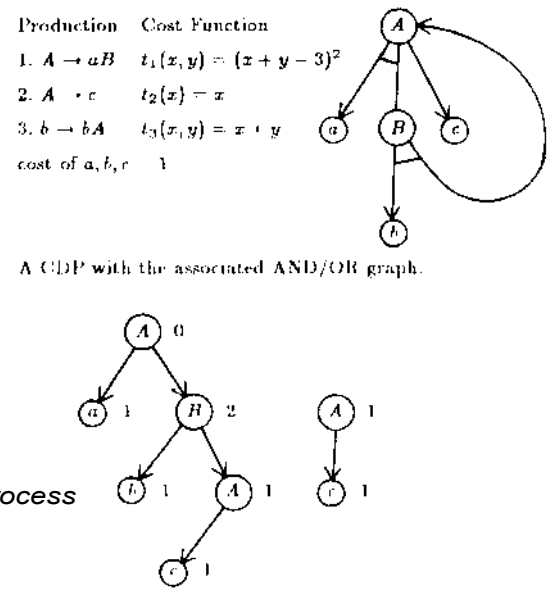

Figure 1. Example of a CDP and solution graphs

We define classes of CDPs based on properties of the cost functions and the grammar, and for each such class analyze the complexity of the minimalization problem. These classes are intended to have some realistic motivation. For example, positive monotone CDPs [8] include generalizations of Dijkstra's algorithm, /-bounded CDPs are a formulation of bounded search. Suppose n is the size of some CDP. In the /-bounded CDP, we assume that any possible minimum value must occur in a tree with no more than *f(n)* nodes. Acyclic CDPs are just the common restriction that the underlying AND/OR graph have no cycles.

The size of a CDP is defined as the size of the grammar (the sum of the lengths of the productions, written as \G\). We shall show upper and lower bounds on the complexity of solving the minimalization problem for various classes of CDP. An upper bound expressed as some function / means that every member of the class can be solved in /(|G|) steps, and a lower bound of / means that infinitely many members of a class require more than /(|G|) steps. The notion of "step of a computation" and bounding computation time are formalized in Alio, et al. [1]. We assume their "logarithmic cost criterion", where the cost of storing or manipulating a large number is then proportional to the length of the number. We assume unit cost for storing or manipulating a single grammar symbol, and cost proportional to the length of a string for manipulating a string of such symbols.

If the cost, functions are easy to compute, we can ignore them in analyzing search complexity. The notion of "easy to compute" must be made precise, because if we just ignore the complexity of computing the cost functions, our conclusions may not be correct. For example, if we did not. consider the complexity of computing the cost functions in giving proofs of upper bounds, by solving a CDP we could compute any recursive function with a constant cost. For let / be some such function. Define cost functions c(d) — *d,* giving the value of a digit as its cost, and $t(x_1 \ldots, X_k)$ — / o $g(x_1, \ldots, x_k)$, where *g* translates a string of digits of length *k* to an integer. Let *G* be the single production *St* -> x, where x is the string of *k* digits representing *x.* Then to compute /(x), find the minimum (the only) value of this CDP. If we charge only for search steps and we have only one step, then we have a constant cost. This difficulty is circumvented by requiring our CDPs to be "honest", that is for some function /, we guarantee that no computation of a cost function *t* takes more than /(|G|) steps. Here we shall assume (unless otherwise noted) that every CDP is honest for some fixed polynomial *h.* This restriction is not necessary for proofs of lower bounds, as computation of the cost functions can only add to the complexity.

3. The Complexity of Several Classes of CDPs

The complexity of the natural CDP, acyclic CDPs, p-bounded CDPs for some polynomial p, and monotone CDPs are analyzed in this section. Figure 2 shows the structural relationships among various CDPs considered here. Figure 3 summarizes our results, showing the relationship of the complexity of the various minimalization problems. In the structure hierarchy, the positive monotone CDP is contained in the ACDP only in the sense that any minimum cost solution graph of the former is guaranteed to be acyclic; an arbitrary parse tree of a positive monotone CDP may not be acyclic. Formal proofs and further examples are presented in [11]; we present only an outline here.

If the range of cost functions is over real numbers, there may be no minimum cost parse tree. Restricting the range of the cost functions to natural numbers guarantees that a solution tree will exist, but as the following theorem shows, there may still be no effective way to find it.

Theorem 1. *The minimalization problem for the* natural *CDP is not recursive.*

Proof. The problem of generating the shortest program *P* that outputs m and halts is reduced to a minimization problem for a CDP. Details are presented in [11]. {x]

This result is not surprising, since *L[G)* may be infinite, giving an unbounded number of parse trees whose backed up value must be tested. If we try to find the minimum by simply generating and testing parse trees, and save the lowest value encountered so far as *min,* there is in general no guarantee that if we stop at any given point, the next, untested, value may not be lower than min. The obvious way to get a decidable problem is to simply bound the space of all parse trees (or all parse trees that could possibly contribute to a minimum) in some way. The ACDP, /-bounded CDPs and MCDPs bound their search space in various ways, giving decidable problems of varying complexity.

To show an upper bound on the time to solve an ACDP, we must first find a bound for the size of any parse tree and the number of distinct parse trees that may be generated by an acyclic context free grammar. Let $exp2(x) — 2^{x^2}$, to keep our exponents from stacking too high. The following lemmas and theorem are proved in [II|.

Lemma 1. *No parse tree for an acyclic grammar G has more than* $2|G|^{1} - 1$ *nodes.*

Lemma 2. No *acyclic grammar G generates more than* $exp2((\backslash G\backslash \sim I)^2)$ *distinct parse trees.*

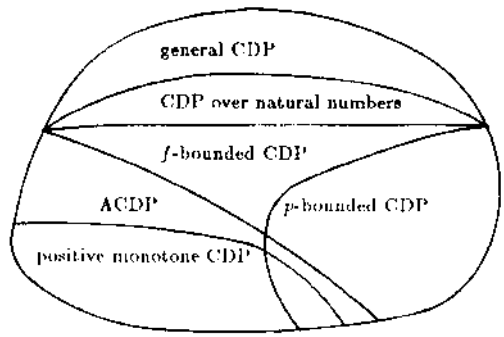

figure 2. Structure Hierarchy

General CDP	May have no solution
CDP over natural numbers	May have no computable solution
f-bounded CDP	Arbitrarily hard
ACDP	Solvable in doubly exponential time Requires exponential time i.o.
p-bounded CDP	*NP*-complete
Positive monotone CDP	$O(n^2)$

Figure 3. Complexity Hierarchy

Theorem 2. The *minimalization problem for any member of ACDP may be solved in 0(cxp2(\G\^2 2)) steps, and requires more than* $2^{|G|/3}$ *steps for infinitely many members of ACDP.*

Proof of the upper bound is by outlining a procedure MIN^Z) that finds a minimal cost derivation for any Z f- ACDP by doing an exhaustive search of the space of all possible parse trees. To show the lower bound it is sufficient to show that there are infinitely many acyclic grammars where every parse tree has more than $2^{|G|/3}$ nodes. [·1

A simple grammar *G* with on the order of exp2(|G|) distinct parse trees is presented in [Ilj. With such a potentially large search space, it would be reasonable to expect (although this does not constitute a proof) that there are infinitely many ACDPs which require on the order of exp2(jc7|) steps for their solution.

We conclude that the minimalization problem for ACDPs (and /-bounded CDPs for exponential /) is not practically solvable. The difficulty arises from the. very compact representation of large trees given by an acyclic grammar. We now consider other means of bounding the size of the search tree. The following theorem shows that even if we bound the size of the tree by a polynomial in the size of the grammar, we still have a hard problem.

Theorem 3. *For every polynomial p such that p{x) > x, the minimalization problem for the p-bounded CDP is NP-complete.*

Proof. Assume *p* is some fixed polynomial, *p(x) > x.* We first show the problem is JVP-hard [1,10]. The knapsack problem with integer weights is easily reduced to a p-boutided CDP. (That is, we show that if we can solve the minimalization problem, then we can solve the knapsack problem, which is known to be NP-complete.) Details are

presented in 11j. The same reduction shows that acyclic CDPs and the natural CDP are NP-hard, since the CDP we have defined is a member of these classes. The acyclic, and natural CDP classes do not appear to be NP-complete, since there is no obvious nondeterministic polynomial time algorithm that solves them.

To complete the proof it must be shown that p-bounded CDP minimalization can be solved by a nondeterministic algorithm in polynomial time. This is done by defining an algorithm which repeatedly makes a nondeterministic selection (a guess) of a production until some string is derived, and for each such string tests its backed up value aginst successively increasing values of a counter. It is not sufficient to simply guess a parse tree and return its value; tliis would produce a tree with fewest possible nodes, but not necessarily minimum cost. [x]

Finally, we consider an easily solved dass of CDPs. A *positive monotone* CDP has all t, monotone nondecreasing, and also satisfying $t_1,(x_1.....x_k) > x_{ji}$

Theorem 4. *The minimalization problem for the positive monotone CDP* can *be solved in 0[\G\$^2$) steps.*

Proof is by presenting an algorithm and proving its runtime and correctness. The proof closely parallels proofs of the runtime and correctness of various algorithms that it generalizes, e.g., Dijkstra's algorithm and certain dynamic programming problems. A similar algorithm appears in [8]. Details are presented in [11]. [x]

Conclusions

Slight variations in the structure of the cost functions or grammar can cause a large change in the complexity of the minimalization problem. If only a single cost function fails to be monotone increasing, the result is a problem of much greater complexity. These results have the following practical application: When modeling some real problem as an AND/OR graph search, every effort should be made to create a graph model in the structural dass with lowest search complexity. The proof of a large lower bound for the complexity of some problem is not necessarily the last word as to whether the problem is practically solvable. This sort of analysis does not take into account the distribution of members of a dass which are most, difficult. Even though we know there are infinitely many difficult members, they could still be very rare, i.e., most members of the dass could be easily solved.

Positive monotone CDPs are the only easily solvable CDPs considered here. There is an interesting natural dass of CDPs, which satisfy only the monotone restriction; these are considered in [8,9], Let c*(W) be the lowest cost for any tree rooted at *W*, the monotone CDP satisfies the equations

1 If *W* is a terminal node, then c*(W) — c(W),

2 If *W* is a nonterminal node, we have c*(W) — min{t_i(c*(A'i), . .. , *c*(X_{ki}) \p_i=-*W*->X_1...X_{kt}}.

We conjecture that the monotone CDPs are not in general solvable in polynomial time (in the context of definitions and restrictions presented here). It would also be of interest to find other natural or easily definable dasses of CDPs with an easily solved minimalization problem.

References

1. Aho, A. V., Hopcroft, J. E., and Ullman, J. D. *The Design And Analysis of Computer Algorithms.* Addison-Wesley, 1974.
2. Carter, L., Stockmeyer, L., and Wegman, M. "The complexity of backtrack searches." In *Proceedings of the 17th Annual Symposium on Theory of Computing* (Providence RI, May 6-8). ACM, New York, 1985.
3. Hall, P. A. V. Equivalence between AND/OR graphs and context-free grammars. *Communications of the ACM* If), 7 (July 1973).
4. Kanal, L. N., and Kumar, V. "Some New insights into the relationships among dynamic programming, branch and bound, and heuristic search procedures." University of Maryland technical report, 1982.
5. Karp, R. M., and Held, M. II. Finite-State Processes and Dynamic Programming. *SI AM J. Appl. Math 15*,, (1907), 693 718.
0. Karp, R. M., Upfal, E., and Wigderson, A. "Are search and decision problems computationally equivalent?" In *Proceedings of the 17th Annual Symposium on Theory of Computing* (May 0-8, Providence, R.I.). ACM, New York, 1985.
7. Knuth, D. E. Semantics of context-free languages. *Math. Systems Theory* 2, 2 (1968), 127-145.
8. Kumar, V. "A Unified Approach to Problem Solving Search Procedures." Ph.D. thesis, Dept. of Computer Science, Univ. of Maryland, Dec. 1982.
9. Kumar, V. A General Bottom-up Procedure for Searching And/Or Graphs. In *Proceedings of AAA]-1984,* 1984.
10. Machl.cy, M., and Young, P. *An Introduction* to the Genera/ 7he*ory of Algorithm..* Elsevier North Holland, 1978.
11. Motteler, 11. E., and Kanal, I,. N. "The Complexity of Searching Several dasses of AND/OR Graphs." University of Maryland Technical Report, 1985.
12. Nilsson, N.J. *Principles of Artificial Intelligence.* Tioga, 1980.
13. Pearl, .1- *Heuristics: Intelligent Search Strategies for Computer Problem Solving.* Addison-Wesley, 1984.