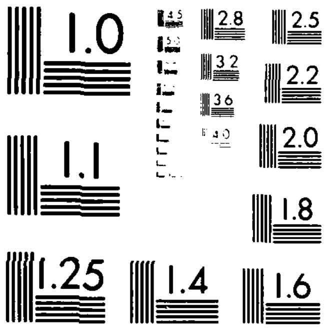
AD-A149 206 SEARCH ALGORITHMS AND THEIR IMPLEMENTATION(U) DUKE UNIV DURHAM NC DEPT OF COMPUTER SCIENCE D W LOVELAND AUG 84 AFOSR-TR-84-1160 AFOSR-83-0205 1/1

UNCLASSIFIED F/G 9/2 NL

END

FILMED

DTIC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0

3

SEARCH ALGORITHMS AND THEIR IMPLEMENTATION

ANNUAL REPORT

GRANT AFOSR-83-0205

FOR THE PERIOD 1 JULY 1983 - 30 JUNE 1984

Donald W. Loveland
Computer Science Department
Duke University, Durham NC 27706

August 1984

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

AD-A149206

REPORT DOCUMENTATION PAGE

1a REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited.
2b DECLASSIFICATION/DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S)	5 MONITORING ORGANIZATION REPORT NUMBER(S) AFOSR-TR- [illegible] 1160
6a NAME OF PERFORMING ORGANIZATION Duke University / 6b OFFICE SYMBOL (If applicable)	7a NAME OF MONITORING ORGANIZATION Air Force Office of Scientific Research
6c ADDRESS (City, State and ZIP Code) Computer Science Department Durham NC 27706	7b ADDRESS (City, State and ZIP Code) Directorate of Mathematical and Information Sciences, Bolling AFB DC 20332
8a NAME OF FUNDING/SPONSORING ORGANIZATION AFOSR / 8b OFFICE SYMBOL (If applicable) NM	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER AFOSR-83-0205
8c ADDRESS (City, State and ZIP Code) Bolling AFB DC 20332	10 SOURCE OF FUNDING NOS PROGRAM ELEMENT NO: 61102F; PROJECT NO: 2304; TASK NO: A7; WORK UNIT NO:
11 TITLE (Include Security Classification) SEARCH ALGORITHMS AND THEIR IMPLEMENTATION	
12 PERSONAL AUTHOR(S) Donald W. Loveland	
13a TYPE OF REPORT Interim / 13b TIME COVERED FROM 1/6/83 TO 31/5/84	14 DATE OF REPORT (Yr., Mo., Day) AUG 84 / 15 PAGE COUNT 25
16 SUPPLEMENTARY NOTATION	
17 COSATI CODES FIELD / GROUP / SUB GR	18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

Research resulting in papers or theses accepted or completed includes (1) the A* search algorithm, (2) search within parsing, (3) search using expectation, and (4) parallel implementation of semantic nets. Research areas where papers are now being written up include (a) parallel implementation of a test-and-treatment algorithm and (b) knowledge evaluation with uncertainties. Research continues in areas ranging from foundations of search theory (e.g., search with limited resources) to applications of search methodology (e.g., search within parsing).

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED [✓] SAME AS RPT ☐ DTIC USERS ☐	21 ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a NAME OF RESPONSIBLE INDIVIDUAL Dr. Robert N. Buchal	22b TELEPHONE NUMBER (Include Area Code) [illegible] 4939 / 22c OFFICE SYMBOL [illegible]

DD FORM 1473, 83 APR EDITION OF 1 JAN 73 IS OBSOLETE.

[illegible]

SECURITY CLASSIFICATION OF THIS PAGE

ABSTRACT

Research resulting in papers or theses accepted or completed includes (1) the A* search algorithm, (2) search within parsing, (3) search using expectation, and (4) parallel implementation of semantic nets. Research areas where papers are now being written include (a) parallel implementation of a test-and-treatment algorithm and (b) knowledge evaluation with uncertainties. Research continues in areas ranging from foundations of search theory (e.g., search with limited resources) to applications of search methodology (e.g., search within parsing).

ANNUAL REPORT

Search Algorithms and Their Implementation

(AFOSR-83-0205)

July 1, 1983-June 30, 1984

Overview

This is the first year annual report for the Air Force Office of Scientific Research grant 83-0205, and the third year of support by the AFOSR. Most of the effort this past year has been on projects initiated under AFOSR-81-0221 but some new efforts are underway and some projects have had a change of emphasis if not direction. We have continued our productivity, with new papers written or presented at conferences, and others accepted for publication.

As suggested in our previous annual report and proposal, work on minimax and related search techniques over game trees seems to be winding down, after several resulting publications. The study of parallel algorithm implementation for the test and treatment problem is concluding, with a paper now being written. The work in parallel algorithms is expected to continue but on a broader plane than just studying one particular search problem. A continuing project is the analytic study of test and treatment problems and a search for good heuristic (or approximation) algorithms. Projects that continue but with a slight change in direction are the search under limited resources study, where the model being studied has changed somewhat, and the knowledge evaluation work, where we have added the interesting study of automatically finding weighting functions after the expert has given a collection of rules to perform a task. The dynamic nature of rule base changes makes the idea of automated or semi-automated weight adjustment attractive. We are very interested also in the computational power of hierarchies of modest-sized rule bases, but this is in a preliminary stage still.

New efforts have arisen, as expected, in the area of natural language parsing and related topics, where search plays an important part. Work on search using expectation to disambiguate noisy voice input has already had a first success. Work on search limiting and pruning devices within natural language parsers is now under study in conjunction with a project in transportable natural language processing (partly funded elsewhere). Here, user-supplied information about the words and concepts that arise in a new domain of interest is utilized during parsing, which leads to unusually "rich" parse structures that contain a good deal of semantic information useful in the subsequent understanding and processing of user inputs. Thus, we can provide most of the search pruning capabilities of so-called "semantic" grammars, yet avoid the severe domain-specific limitations they entail. Finally, some innovative thinking yielded two approaches to developing parallel algorithms for handling semantic nets, an important data structure in natural language processing.

Regarding publication and other presentations of our work to the outside world, several papers supported by AF grants previously accepted have now appeared in print. (But the time lag for processing can be long; one paper, on the test-and-treatment problem (Loveland), has been accepted subject to revision for two years. One reviewer, who is apparently very interested in the details of the paper, has iteratively suggested good, but subtle, detailed improvements to a substantial proof. We expect this is the last pass.)

New papers now accepted for publication are a paper on the computational complexity of heuristics for the A* algorithm (Valtorta), which shows that a certain wide class of initially appealing heuristics for guiding search are actually not efficient, and a paper on a Transportable natural language grammar framework (Ballard, Tinkham) that permits easier study of various search-related problems regarding the accuracy and efficiency of parsing. Two theses were concluded, a Ph.D. thesis on search using expectation (Fink) and an A.M. thesis on parallel implementation of semantic nets (Jackoway). (Both students were funded by other sources but the supervising faculty, who were active in these projects, were funded by this AFOSR grant.) Several papers are in the writing stage, including a paper on parallel implementation of the test-and-treatment problem (Wagner, Han, Deval, Loveland) and a paper on an example technique for knowledge evaluation within rule-based knowledge systems (Loveland, Valtorta; a significant extension of the IJCAI-83 paper). Several conference papers have been given and are to be given this next half year; these are listed after the publications and reports summary.

Research Objectives

Our research objectives remain that of studying search techniques and the methodology of search, both in abstract settings and within context of applications. We have begun in-depth search studies within the context of natural language processing, and continue our research in aiding knowledge acquisition for knowledge-based systems, but also continue the more general search problems of test-and-treatment procedures and search with limited resources. Our work with parallel algorithms has been relatively intense recently (but may slack some as current students leave).

It is important not to place too much emphasis in apparent trends within our research pattern; for example, we did not predict last year the strong interest and success in studying parallel implementations of semantic nets. However, the ingredients were present: the parallel machine model (the BVM), the bright and motivated students, and the ongoing interest in AI search-related problems.

Research Status

The status report is divided into projects with the personnel involved named in parentheses. Graduate students' names appear before their advisor's name when the advisor's role is limited to problem formulation and research guidance.

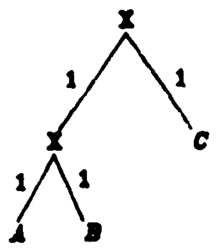
Search with limited resources (Mutchler, Loveland)

We have previously developed and investigated a model for studying search with limited resources [4]. During the past year, we considered additional models. Here is a description of one for which we have some results.

Generate trees as Fuller [2], Ballard [1], and Karp and Pearl [3] have done. That is, we consider complete binary depth N trees. Arcs are independently assigned 1 or 0 with probability p and $q = 1-p$ respectively. Each leaf value is the sum of the arc values on the path from the root to the leaf. Hence values of sibling leaves are positively correlated. One argument for using the arc-sum model is that real-world search trees often possess this correlation property.

Search such trees as Karp and Pearl [3] did. One begins at the root and ***expands*** nodes, thereby finding the values of the two arcs below it. No node can be expanded until its parent has been expanded. Now suppose that you have done some node expansions and have sufficient search resources left for exactly n more expansions. After that expansion you must choose a leaf. Your goal is to minimize the expected value of the chosen leaf. The question is: what policy should you use to select nodes for your n node expansions?

To show that the answer may be surprising, consider the following simple-minded example. Suppose you are searching the depth 5 tree whose top two levels look as pictured below after two node expansions. (Nodes A, B and C are the roots of unexpanded subtrees. The numbers beside arcs indicate expanded nodes.) Let $p = 0.8$. Suppose you have resources left for only *one* more node expansion. Which node would you expand?



The "common-sense" answer is to expand the node that "looks best" at this point. In this case that would be node C. We have shown that this heuristic (a member of the class of "greedy algorithms"), while correct under many circumstances, is not always best. Here the optimal expansion is of either of nodes A or B.

We have two results so far. First, we can show which decision policy is the optimal policy for $n = 1$. Second, for $n \leq N$, we have a proof that for p a unit fraction (1/2, 1/3, 1/4, ...), the greedy policy is optimal. The mathematics used to do the modeling is that of Markov decision processes (see, for example, [5]).

We will continue our investigation of this model. We will also look more closely at a variant of the problem. The variant is appropriate in situations wherein the search proceeds through "easy" and "difficult" terrain.

[1] Ballard, Bruce W. The *-minimax search procedure for trees containing chance nodes. *Artificial Intelligence 21* (1983), 327-350.

[2] Fuller, S.H., Gaschnig, J.G. and Gillogly, J.J. An analysis of the alpha-beta pruning algorithm. Dept. of Computer Science Report, Carnegie-Mellon University, Pittsburgh, PA, 1973.

[3] Karp, R.M. and Pearl, J. Searching for an optimal path in a tree with random costs. *Artificial Intelligence 21* (1983), 99-116.

[4] Mutchler, David C. Search with limited resources. Duke Technical Report CS-1983-1. Duke University, Durham NC, February 1983.

[5] Ross, Sheldon M. *Introduction to Stochastic Dynamic Programming* (Academic Press, New York, 1983).

Test and treatment procedures (Loveland)

The test and treatment problem is a problem in finding optimal or near optimal procedures that indicate when to test and when finally to treat. This problem was formulated while working on the binary testing problem the first year of the preceding grant. The test and treatment (TT) problem is an enrichment of the binary testing problem where now one allows treatments to administered before the faulty object has been isolated.

The test and treatment problem requests the selection of a minimum test and treatment procedure under an expected cost criteria. The problem arises whenever a fault (disease, system malfunction) must be treated. The classic example is medical diagnosis and treatment, but other applications also are important, such as computer system fault location and correction and logistical system breakdown correction. In general, the problem exists whenever a sizable population of complex objects (people, ships, computers) must be maintained at reasonable cost.

The problem specification consists of a universe $U = \{0,1,\ldots,k-1\}$ of k objects, each with an associated weight p_i, and a set of tests and treatments T_i, $1 \le i \le n$, each with an associated cost. The T_i, $1 \le i \le m$, denote tests, and the T_i, $m < i \le n$, denote treatments. We assume that only one object is actually faulty, its identity is unknown, and each object i has an *a priori* likelihood p_i of being the faulty object.

Each test and treatment is specified by a subset of the universe; if the unknown object is in the test or treatment set then the test responds positively, or "is successful", or the treatment is successful. If the test is successful, one eliminates the other objects from consideration (and if negative, one eliminates the test set of objects), while a successful treatment ends the procedure. A failed treatment means the processing must continue. Cost of a procedure is an expected cost; i.e.,

$$\sum_i (\textit{cost of detecting object } i)\,(p_i)$$

Because nothing is known about this class of problems (at least, judging from a literature search), we have had to formulate the problem before attempting a solution. The general problem is formulated in a paper under preparation by R. Wagner, Y. Han, D. Deval and D. Loveland where a dynamic programming technique is used to solve the problem. The technique is classical so the interest in this paper lies primarily in the parallel implementation of the particular dynamic programming formulation.

However this general solution has exponential running time in the number of objects and this is almost surely the case for all general solutions (the problem is NP-hard), although parallel processing helps greatly here. We have therefore also focussed on fast (polynomial) algorithms for special cases and approximation algorithms for the general case. Let us call the case where all tests and treatments are available the ***complete TT problem*** We have had hopes that the complete TT problem (with certain assumptions regarding costs) has a nice algorithm for selecting the minimal cost TT procedure because the well-known algorithm is a elegant fast algorithm for the related binary testing problem. We have now had time to pursue this and see that interesting anomalies are arising that seem to thwart a clean algorithm formulation. It appears that the minimizing moves for treatment sometimes move counter to the minimizing moves for tests. We understand many characteristics analytically but need to turn to the computer to assess experimentally whether further anomalies can occur in other special cases.

Parallel search algorithms (Wagner, Han)

During the past year, we have completed the design of an efficient parallel algorithm for the Test and Treatment Problem, which is also under investigation by Loveland. Our effort has been concentrated on the development of algorithms for the Boolean Vector Machine for this problem. The machine in question is designed as an array of many thousands of processing elements, connected by a general permutation network called the Cube Connected Cycles network, a well-known interconnection network. It uses bit-serial algorithms and communication links. Nonetheless, it is expected to outperform a conventional machine of equal cost by roughly the of the number of words in the conventional machine's memory to 8 times the width of those words.

The performance advantage of the BVM can only be achieved by redesign of algorithms, to take advantage of massive parallelism at the algorithm level. Our algorithm achieves this for the computationally intensive dynamic programming algorithm which solves the Test and Treatment problem exactly. We hope that the speed-up achieved is sufficient that experimental comparison of the "true" answers to examples of this problem can be made with the heuristic answers of Loveland's methods. In addition, there is some hope of being able to use the exact algorithm for modest-size problems without spending years awaiting the answers. Such usage awaits the construction of a Boolean Vector Machine which is large, compared to the prototype machine currently under construction at Duke and UNC.

Our algorithm has been written, and simulated on a functional simulator of the BVM, which is capable of handling some 64 PE's. The algorithm will be moved to the "prototype" BVM, containing some 2048 PE's, when that machine's construction is completed.

Yijie Han has devoted one chapter of his Master's thesis to this algorithm. This chapter is currently being expanded and revised into a paper, which should be ready for submission to a journal in September. The paper describes the methods used in obtaining this parallelization of the dynamic programming algorithm, and points out those aspects of the parallel algorithm which seem significant to other algorithm designers.

Actual experimental use of the algorithm, in investigating modest-size problems, awaits construction of a larger BVM.

Perhaps just as important as speed is the question of the number of PE's needed for a problem of given size. This number corresponds roughly to the number of words of memory needed on a conventional machine. The algorithm we devised requires (n**2)*(2**n) PE's as written, and so becomes infeasible on a 2**16 PE array for very small values of n. An algorithm running on n*(2**n) PE's might well be preferable. In addition, these algorithms should be generalized, to take advantage of a smaller than optimal PE array, at a corresponding sacrifice in time. These investigations have not yet been completed.

Allocation of resources in parallel searches (Reibman, Wagner)

During the past year, we began to study the problem of deciding how best to use an arbitrary collection of Processing Elements (PE's) working in parallel, in solving heuristic search problems. A model of one class of such searches has been defined, in which the search space can be represented as a depth 2 AND/OR tree. Each PE is given a single (not necessarily unique) region of the search space by assigning it an "OR-node" of the tree. At each step, each PE randomly probes its search space by sprouting and evaluating a child of its assigned Or-node, thus selecting a candidate and evaluating whether it in fact satisfies the criteria for a solution.

This investigation has focused on the general questions of how many PE's can usefully be employed in such an effort. An attempt has been made to determine how to best allocate the PE's to the individual regions of the search space. Reibman noted that more than one PE can be assigned to each sub-task (i.e. each OR-node of the tree). This might be both convenient (if more PE's are available than subtasks), and efficient (if the probability of two investigations of the same candidate solution is low). Under appropriate assumptions on the probability of locating a solution, and of selecting any given candidate from the available space, Reibman has developed a probabilistic model of the expected time to solve a set of tasks, as a function of the number of PE's allocated to each task.

One of the outcomes of this investigation may be a simple, effective heuristic for performing PE-to-task allocation. Roughly, if it is possible to guarantee that the same candidate solution is not investigated more than once, then any PE-to-task allocation which keeps all available PE's busy is near-optimal (apart from edge effects which arise because tasks are assumed indivisible). We suspect that, if communication restrictions make it impossible to guarantee that candidate solutions are not investigated more than once, random selection of candidate solutions by the PE's assigned to a given task may work almost as well as fully informed selection.

This model is being described in a working report which is under preparation.

Semantic Networks on a Massively Parallel Machine (Jackoway, Tomboulian, Biermann, Wagner)

As explained in several sections in this report, an important strategy for dealing with search is through the use of parallelism. Instead of sequentially following the many possible paths, separate processors are assigned to each path, and in some cases, exponentially expensive calculations can be reduced to linear time. Two of the endeavors along these lines relate to the implementation of semantic networks on the Boolean Vector Machine. Semantic networks are a very important knowledge representation scheme in many artificial intelligence projects so that results in this project will have important implications.

In the typical semantic network, modes represent objects in the domain and transitions from node to node represent relationships between objects. For example, the fact that John is the parent of Bill would be represented by two nodes labelled respectively, John and Bill, and with an arc from one to the other labelled "parent of". One can build massive amounts of knowledge about a given application into such a network and then query it by following out the appropriate arcs. For example, one could answer the question "Who are the ancestors of Bill?" by first locating the node "Bill" and then following all "parent of" arcs. With traditional computing methods, one would sequentially follow out each of the paths, but with the Boolean Vector Machine, all paths can be followed simultaneously.

The method is to spread the network across the parallel machine with computational power available at each node to compute locally queries radiating through the net. Two of our graduate students, Gary Jackoway and Sherry Tomboulian, have independently developed methods for implementing semantic nets on the BVM. The Jackoway method assigns two adjacent elementary processors to each arc. The computation proceeds by alternately passing information across arcs and then employing his powerful "share" algorithm to spread information from a node at the end of one arc to all other arcs that might be traversed next in a computation. The Tomboulian method places one full semantic net arc on an elementary processor and then enables sequential arcs to communicate by migrating the necessary information across the net. Both methods offer the possibility of increasing semantic net query speeds by orders of magnitude. These methodologies are extremely attractive in that as Jackoway has pointed out:

"The time required to process a query is dependent strictly on the pattern of the query, not on the size of the classes being processed. A system built using this knowledge representation will give consistent semantic processing performance.

The order of processing a query does not affect speed. Thus there is no need for heuristics and monitors to determine the most efficient way to process a query."

Knowledge evaluation (Loveland, Valtorta)

This last August (1983) we presented a paper at IJCAI-83 in Karlesruhe on the type of subject matter we mean by "knowledge evaluation" within the expert system domain. The paper "Detecting ambiguity: an example of knowledge evaluation" developed an aid for testing the compatibility of a new rule added to a rule base for an expert system of the classification type. (Classification expert systems include many of those doing diagnosis, such as MYCIN.) When a new rule is added, a great deal of testing is sometimes necessary to determine the compatibility of the new rule with the rest of the knowledge base. (Note that in expert systems it is not reasonable to *prove* the compatibility of the rule with the rule base because the problem does not have clear formal specifications. If it did there would probably be little use for an expert system.) The method proposed reduced greatly the number of input vectors needed to test for compatibility.

The paper has restrictions used to make the given method work. One requirement is that a "superposition" property hold; that is, that if a certain set of symptoms cause a certain fault (or disease), then adding other symptoms can only cause possible added faults to occur. In other words, no inhibiting inputs were allowed that could remove a conclusion that was present if the input signal were absent. Another constraint on our method was the use of 0-1 value logic; there was no allowance for weighted evidence, or certainty factors. We now have realized some extensions to our original method that allows for certain types of certainty calculi. Certain other types do not seem feasible to employ with our method. This will be in an expanded version of the paper, intended for a journal and to be written this year.

Another project underway in this area is seeking algorithms to automatically calculate appropriate weights to assign rules in a rule-base given a set of tests. The system Seek (see Politakis and Weiss [1]) did weight adjustment (and, in fact, rule modification) of the top level, but this is not sufficient in general. One could, of course, ask that entire rules be learned from the test set (this is the subject of research in learning) but there are at least three reasons to consider a less ambitious approach as well. First, learning is not that well understood yet; second, a much larger test set may be needed to induce rules than needed just to set their confidence factor; and third, often experts exist who can supply at least many of the rules (or many may be natural and almost self-evident) so computation time is not needed to induce them. Of course, the minimal cost of solving the weight setting problem could be exponential in the number of weights to be set or the number of tests to process but we hope for better results. (It is clear that weights can be determined, if they exist, given sufficient time, so the issue is to seek fast algorithms). We have some ideas regarding fast algorithms but only very preliminary results.

The advantages of a fast algorithm for general weight setting are significant. Inconsistent rule sets can be discovered, and meaningless rules deleted. Somewhat trickier but possible is detecting bad test cases. The presence of bad test cases will distort the appraisal of the value of the system at best, and often can influence the design of the system. For these reasons we feel the problem is an important one to pursue.

Politakis, P. and S. Weis, Using empirical analysis to refine expert system knowledge bases. *Artif. Intell., 1984, p. 23-48.*

Graduate Course Adviser (Valtorta, Smith, Loveland, Harrison)

The Graduate Course Adviser (GCA) is an expert system to advise graduate students regarding course selection, built as a learning tool to help us understand the techniques and problems of developing expert systems. It is a rule-based system, initially patterned after MYCIN (see [1]), which has evolved into a multi-stage system using algorithms and tables as well as rules. Although our main research in expert systems lies elsewhere, lessons learned in the design of GCA have led to two "invited" papers to be given in workshops on knowledge-based systems ([2],[3]). We have received volunteer help from Bruce Smith, a Ph.D. candidate at the University of North Carolina, and work will continue with Tim Harrison, a Duke graduate student beginning work on determining the equivalence of courses for those seeking recognition for non-Duke courses. Tim is not funded by the AFOSR grant at present.

Because our work in knowledge evaluation has been viewed as "too theoretical" by some in the knowledge-based systems field, we have felt it wise to establish our credentials by gaining some experience as practitioners. Most important, of course, is the actual experience itself, helping us to judge what will be needed in future knowledge-based systems.

[1] Shortliffe, E. H. *MYCIN: Computer-based Medical Consultations*. Amer. Elsevier, New York, 1976.

[2] Valtorta, M. Knowledge refinement in rule bases for expert systems: an appliation-driven approach. *First Int'l Workshop on Expert Database Systems*, Kiawah Island, SC, Oct. 1984.

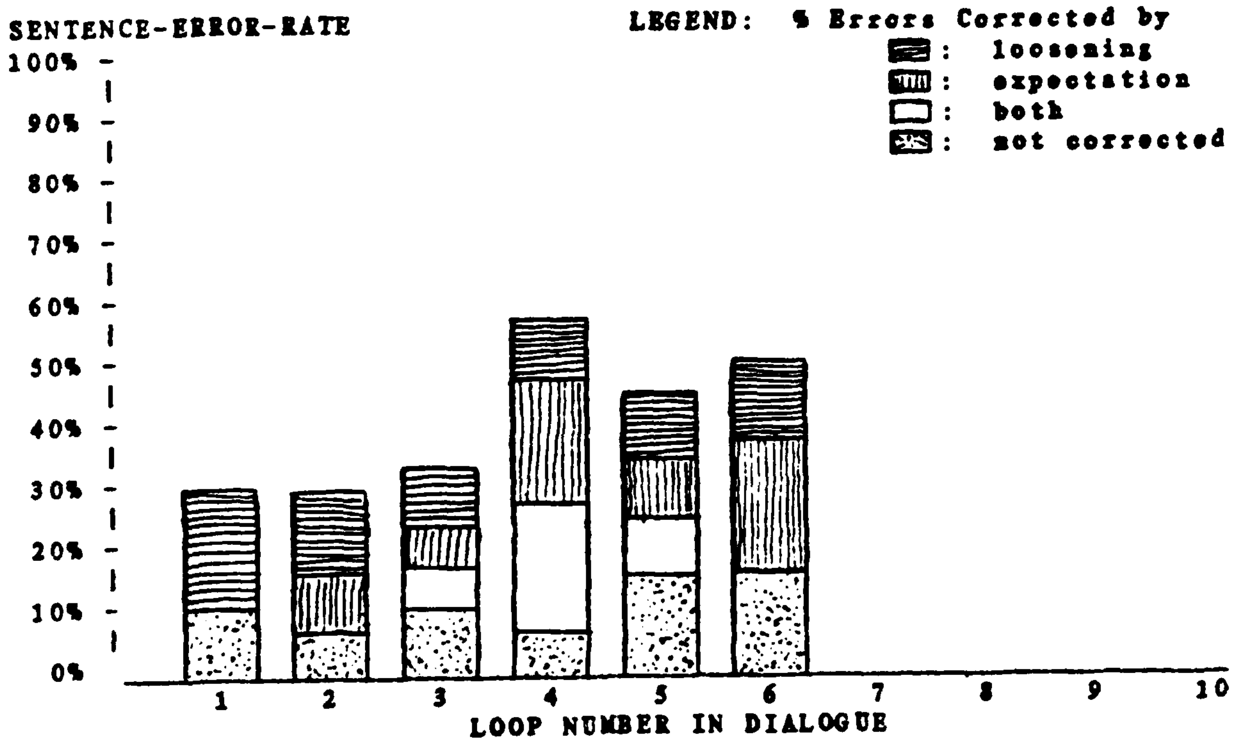
[3] Valtorta, M., B. Smith and D. W. Loveland. The Graduate Course Adviser: a multiphase rule-based expert system. *IEEE Workshop on Principles of Knowledge-Based Systems*. Denver, Dec. 1984.

Search using expectation (Fink, Biermann)

One of the basic tenets of artificial intelligence is the necessity of using past experience to guide future behavior ("learning") in order to have truly "intelligent" systems. Although systems that learn have been built since the beginning of AI research, they have not been particularly notable, partly because the domains chosen have not been both tightly constrained yet developed enough to be interesting. Recent work in AI suggests that areas and systems exist that offer reasonable opportunity to explore at least limited forms of learning. One limited form of learning is modeling events to use to predict future events.

In order to gain experience with this concept ourselves, we decided to assemble an error correcting natural language parser for voice input which collects dialogues on a particular subject area and which attempts to use them to predict incoming sentences. In early examples of the dialogue, the system has no predictive ability and depends on its basic search algorithm to parse incoming sentences and to analyze their meaning. However, after one or more dialogues have been completed on the given subject, a dialogue model is assembled which is used to guide the search efficiently toward the user's intended meaning. This is a particularly good application for predictive search because voice recognizers tend to make many errors and a good guessing scheme for error correction is useful.

Such an error correcting scheme has been built as a part of Pamela Fink's doctoral dissertation project and is described in *The Acquisition and Use of Dialogue Expectation in Speech Recognition*, Report CS 1983-101. This system was tested as an error correction parser for connected speech recognized by a Nippon Electric Corporation DP-200 on a vocabulary of limited size. For example, in one task, the subjects were asked for solve six sets of linear equations with two equations and two unknowns. The following graph shows the results where a bar is given for each problem and the height of each bar gives the percentage of sentences that would have failed if no error correction were available. The horizontally cross hatched portion of each bar shows the percentage of sentences that were connected using non-expectation based methods. The dotted portion of each bar shows the portion of the sentences that were not correctable by the system. After the first dialogue, the expectation system was able to error correct some of the sentences as shown by the vertically cross hatched regions. The white regions in the bars indicate errors corrected by a combination of expectation and other methodologies. The higher raw error rates in the last three dialogues probably were the result of faster and less precise speech.



The results of the tests show substantial performance increases are possible through the use of history based predictive mechanisms. We are now examining more general forms of this type of learning for application to search methodologies in general.

A Search-Based Approach to Natural Language Parsing (Ballard, Tinkham)

In order for a computer to understand and carry out an English command, or answer an English question, a determination must first be made as to the syntactic structure of the input. That is, the input must be *parsed*. As is well known, the parsing problem is fundamentally a ***search*** problem, where the ***grammar rules*** used to specify acceptable structures for a sublanguage of interest correspond to ***production rules*** in an abstract problem-solving situation. In effect, these rules tell how words may be combined to form phrases, how one phrase may be attached as a modifier to another, and so forth. Formally, then, the "parse" of a sentence constitutes the "goal" state of the associated search, while intermediate phrases correspond to subgoals. As an (oversimplified) example, the rule

<PrepPh> -> Prep <NP>

would mean that a prepositional phrase is formed from a preposition and a noun phrase, in that order.

In typical natural language parsers, the search for a set of mutually acceptable choices for how each word is to be interpreted assumes combinatoric proportions. Some of the specific problems that contribute to the size of the search space are (1) alternative attachments for a given word or phrase will present themselves, for example in "Jenkins returned to the base without adequate fuel" the inadequacy of fuel could refer to either Jenkins or the base; (2) a given word may be used in several ways, for example "pilots" could be either a plural noun or a singular verb; (3) in a realistic, noisy voice environment, one cannot be certain what individual words were spoken without considering the context of surrounding words, for example "forward" and "toward" might be confused, as might "four degrees" and "forty grease".

A careful study of the types of problems suggested above reveals that many of them can be effectively dealt with during parsing if proper domain-specific knowledge is available at the crucial moment. By "effectively dealing with" these problems, we mean that either the correct structure will be found more quickly or, for truly ambiguous structures, the correct structure will be found more often. Our question then becomes how to acquire and utilize domain-specific knowledge durig parsing.

As described in Ballard and Tinkham (1984) we have developed a grammatical formalism in which "augmentations" to a basically context-free specification for a class of desirable inputs enable the grammar writer to have domain-specific information consulted during parsing. Our overall approach, which allows for the pruning of the search for acceptable syntactic structures, involves

1. a phrase-structured ***grammatical formalism;***

2. a required ***dictionary*** format and associated ***compatibility*** file; and

3. an implied format for ***parse structures.***

Our grammar rules assume that an input to be parsed will be presented as a sequence of sets of token candidates, where each token candidate corresponds roughly to a word or inflection of a word found in the system dictionary. Each dictionary listing for a word is made up of one or more ***meanings***, each meaning being comprised of (a) the ***word*** itself; (b) its ***part of speech;*** (c) the associated ***root*** word; and (d) zero or more associated ***features***, each with one or more possible values. As an example, the entry

```
((pilots    Noun      pilot  (nt person)   (sp plural))
 (pilots  Vpresent   pilot   (sp sing)))
```

says basically that the word "pilots" can be a either (i) the plural for the noun "pilot", which refers to domain objects of type "person", or (ii) a present tense verb for a singular noun.

In addition to the "features" found in the dictionary, which provide for simple context dependencies, a *compatibility* file is assumed to be available which contains information on acceptable attachments for such units as prepositional phrases and relative clauses. This information is responsible for much of the pruning of the search for an acceptable parse. For example, the triple

(aircraft at place)

(seacraft in place)

might indicate that "at Wright-Pat" will acceptably attach to "bombers" and "in the Atlantic" to "subs", but not vice versa, so that the the sentences

"How many bombers fired on subs in the Atlantic?"

"How many bombers fired on subs at Wright-Pat?"

will be assigned different structures, although their surface structures are the same.

At the heart of our formalism is a sophisticated but simple method for the designer of a grammar to assure that *compatibilities* of the type suggested above are checked for. For example, the grammar rule

(Call Nounmod (agree ntarg nt))

says that the "ntarg" information associated with whatever dictionary words are successfully incorporated into the "Nounmod" phrase must be compatible with the "nt" information of the current phrase. In effect, this assures that the nountype (nt) of the current phrase agrees with the nountype-of-the-argument (ntarg) of the Nounmod phrase to be built up. That is, don't get a modifier that isn't legal in the current context! As a second and more sophisticated example,

(Call PrepPh (Prepinfo (Head Prep Arg) Head))

says to look for a prepositional phrase so that the Head of the current phrase, the actual preposition (Prep) found, and the argument (Arg) of the preposition are compatible, as specified by the "Prepinfo" set of tuples (that is, the aircraft-at-place sorts of triples).

Although our long-term goals deal primarily with general-purpose processing, especially of complex noun phrases with elaborate relative clause and other post-nominal modifiers, our current prototype ("LDC") has been designed to solve database-like retrieval problems, as described in Ballard, Lusth and Tinkham 1984. Accordingly, most of our present grammar consists of (a) a fairly elaborate noun phrase grammar, and (b) a case-like specification of sentence-level and fairly complex relative clauses. For example, we presently provide for relative clauses of many varieties (e.g. "by whom a book was given to Bill" as well as "that gave a book to Bill"). Information on the scope of the current syntactic and semantic coverage of our overall processor can be found in Ballard 1984, and the results of several experiments with our current grammar and parser can be found in Ballard and Tinkham (1984). For example, we found that our "start" files, which in essence allow certain grammar rules (alias production rules) to be summarily dismissed as unviable, without doing any search inside them, reduced the average number of such rule applications by more than half.

Ballard, B. The syntax and semantics of user-defined modifiers in a transportable natural language processor. *Proc. of the 22nd Annual Meeting of the ACL* (joint with *COLING-84)*, Stanford University, July 1984, pp. 52-56.

Ballard, B. and Tinkham, N. A phrase-structured grammatical framework for transportable natural language processor. To appear in *Amer. J. Computational Linguistics*, 1984.

Ballard, B., Lusth, J., and Tinkham, N. LDC-1: A transportable, knowledge-based natural language processor for office environments. *ACM Trans. on Office Information Systems*, Volume 2, No. 1, 1984.

Supported Personnel

Loveland, Donald W. (Principal Investigator)

Ballard, Bruce W. (Co-principal Investigator)

Biermann, Alan W. (Co-principal Investigator)

Wagner, Robert A. (Co-principal Investigator)

Mutchler, David (Research Assistant)

Reibman, Andrew (Research Assistant)

Valtorta, Marco (Research Assistant)

Publications and Reports

July, 1982 - June, 1984

Chronologically ordered

1. Biermann, A., J. Fairfield and T. Beres. Signature tables and learning. *IEEE Trans. on Systems, Man and Cybernetics*, Oct. 1982, pp. 635-648.

2. Biermann, A. Dealing with search. *Automatic Program Construction Techniques* (Eds. Biermann, Guiho, Kodratoff). MacMillan Publ. Co., 1984.

3. Ballard B. The *-minimax search procedure for trees containing chance nodes. *Artif Intelligence* (to appear). Oct. 1983, pp. 327-350.

4. Loveland, D. Performance bounds for binary testing with arbitrary weights. To appear in *Acta Informatica*.

5. Mutchler, D. Search with limited resources. Duke C.S. Report CS-1983-1, Jan., 1983.

6. Ballard, B. Non-minimax search strategies for minimax trees: theoretical foundations and empirical studies. Duke C.S. report CS-1983-13, July, 1983. (submitted for publication)

7. Fink, P. The acquisition and use of dialogue expectation in speech recognition. Ph.D. thesis, Computer Science Department, Duke University, 1983. Also C.S. Report CS-1983-101.

8. Jackoway, G. Associative networks on a massively parallel computer. A.M. Thesis, Computer Science Department, Duke University, 1984.

9. Ballard, B., N. Tinkham. A phrase-structured grammatical framework for transportable natural language processors. To appear in *Amer. J Comput. Linguistics*, 1984.

10. Valtorta, M. A result on the computational complexity of heuristic estimates for the A* algorithm. To appear in *Infor. Sciences*.

Conference Presentations

July 1982 - December 1984

1. Ballard, B. A search procedure for perfect information games of chance. *Second National Conf. on Artif. Intell. - 82*, Pittsburgh, Aug. 1982.

2. Loveland, D. Knowledge acquisition and evaluation. Army Conf. on AI Application to Battlefield Info. Systems, Silver Springs, Md., April, 1983.

3. Ballard, B.W. and A.L. Reibman. What's wrong with minimax? *1983 Conf. on Artif. Intell.*, Rochester, Mich., April, 1983.

4. Reibman, A.L. and B.W. Ballard. Non-minimax search strategies for use against fallible opponents. *Third Natl. Conf. on Artif. Intell.*, Washington, D.C., August, 1983.

5. Reibman, A.L. and B.W. Ballard. Non-minimax search strategies for use against fallible opponents. *21st ACM Southeast Region Conf.*, Durham, N.C., April, 1983.

6. Loveland, D.W. and M. Valtorta. Detecting ambiguity: an example of knowledge evaluation. *Eighth Intern. Joint Conf. on Artif. Intell.*, Karlsruhe, W. Germany, August, 1983.

7. Valtorta, M. A result on the computational complexity of heuristics for the A* algorithm. *Eighth Intern. Joint Conf. on Artif. Intell.*, Karlsruhe, W. Germany, August, 1983.

8. Ballard, B. The syntax and semantics of user-defined modifiers in a transportable natural language processor. *Proc. of the 22nd Annual Meeting of the Assoc. Comput. Linguistics*, Stanfod University, July, 1984.

9. Valtorta, M. Knowledge refinement in rule bases for expert systems: an application-driven approach. *First Intern. Workshop on Expert Database Systems*, Kiawah Island, S.C., October, 1984.

10. Valtorta, M. B. Smith and D. W. Loveland. The Graduate Course Adviser: a multiphase rule-based expert system. *IEEE Workshop on Principles of Knowledge-Based Systems*. Denver, December, 1984.