Technical Report CS74019-R

AN ARTIFICIAL INTELLIGENCE APPROACH TO THE SYMBOLIC FACTORIZATION OF MULTIVARIABLE POLYNOMIALS

Billy G. Claybrook

April 1974

Department of Computer Science, Virginia Polytechnic Institute and State University, Blacksburg, Virginia 24061

ABSTRACT

A new heuristic factorization scheme that uses learning to improve the efficiency of determining the symbolic factorization of multivariable polynomials with integer coefficients and an arbitrary number of variables and terms is described. The factorization scheme makes extensive use of Artificial Intelligence techniques, e.g. model-building, learning, and automatic classification in an attempt to reduce the amount of searching for the irreducible factors of a polynomial. The approach taken to polynomial factorization is quite different from previous attempts because: (1) it is distinct from numerial techniques, (2) possibilities for terms in a factor are generated from the terms in the polynomial, and (3) a reclassification technique is used to allow the application of different sets of heuristics to a polynomial during factorization attempts on it.

Tables are presented that demonstrate the importance of learning to the efficiency of operation of the scheme. Factorization times of polynomials factored by both the scheme described in this paper and Wang's implementation of Berlekamp's algorithm are given and compared, and an analysis of variance experiment provides an indication of the significant sources of variation influencing the factorization time.

I. INTRODUCTION

Before 1967, attempts to determine the symbolic factorization of multivariable polynomials with an arbitrary number of terms and variables using both heuristic methods and algebraic methods proved to be largely unsuccessful. These attempts were able to factor only simple polynomials and were time consuming. In 1967 Berlekamp [3] published a new algebraic algorithm for factoring univariate polynomials with integer coefficients. Various modifications and extensions of Berlekamp's algorithm [19], [24], [26] have led to other algorithms for factoring univariate and multivariate polynomials. In particular, Wang's algorithm [24] appears to be one of the most efficient for both univariate and multivariate polynomials. Wang's algorithm factors many multivariate polynomials efficiently; however, there are some types of polynomials with which it has problems.

This paper describes a new heuristic factorization scheme that uses learning and other heuristic programming techniques to improve the efficiency of determining the symbolic factorization of multivariable polynomials with integer coefficients and an arbitrary number of variables and terms. This factorization scheme is applied only to multivariable polynomials. We do not try univariate ones (except for a few special cases) since Berlekamp's algorithm [2] is a clear winner in most instances. This factorization scheme makes extensive use of Artificial Intelligence techniques, e.g. model-building [18], learning, and automatic classification in an attempt to reduce the amount of searching for all of the irreducible factors of a polynomial. The reader should keep in mind that we are not claiming that this scheme is a general factorization algorithm for there are a few instances where factorization of a reducible polynomial could fail. However, it does succeed in most instances. Polynomial factorization using heuristic techniques is basically a combinatorial problem. Our approach to factorization

is distinct from numerical techniques, e.g. Kronecker's algorithm [12], and advanced algebraic techniques, e.g. Berlekamp's algorithm [2].

The primary purposes of this paper are:

1. to provide an overview of the implementation of the factorization scheme in the learning program POLYFACT [5],

2. to present the fundamental details of the heuristic factorization scheme and discuss its usefulness to polynomial factorization,

3. to describe Berlekamp's algorithm and indicate where it works well and where it has problems,

4. to show that an Artificial Intelligence approach to factoring multivariable polynomials is feasible for many multivariable polynomials, and

5. to demonstrate that learning can improve the efficiency of operation in the solution of a complex problem such as polynomial factorization.

The reader can interpret this paper jointly as one that describes a heuristic factorization scheme, and as one that demonstrates the use of learning and other Artificial Intelligence techniques in solving complex problems.

Section II presents an overview of POLYFACT and the implementation of this factorization scheme. Section III provides related work in polynomial factorization and learning programs. Section IV gives a detailed description of the factorization scheme, and Section V presents results and possible modifications.

II. OVERVIEW OF POLYFACT

This section provides a brief description of the implementation of this factorization scheme in the learning program POLYFACT and gives an overview of the scheme itself (a detailed description appears in Section IV). The primary objectives in the development of POLYFACT were:

1. to design a multivariable polynomial factoring program that can be used as a vehicle in a complex-learning environment,

2. to develop a representation for heuristics that allows for dynamic creation and modification during program execution,

3. to show that learning through the dynamic modification of heuristics can be used successfully in a complex environment to increase the efficiency of the program,

4. to demonstrate that a classification scheme can be used to allow the program to extend itself to newly classified polynomials, and

5. to show that a classification scheme can be used as a mechansim for implementing localized learning.

Description of POLYFACT

POLYFACT is written in FORTRAN V and implemented on a UNIVAC 1108. POLYFACT consists of over 230 subroutines and functions. Fig. 1 gives a flow diagram describing the operation of POLYFACT. Polynomials are input to POLYFACT in FORTRAN notation in either factored form (this form is allowed for convenience) or in expanded form. Polynomials input in factored form are expanded by taking the product of the factors (the factors may or may not be irreducible factors).

Prior to a factorization attempt each polynomial is completely simplified (i.e. has all like terms combined) when the polynomial is placed into canonical form. In this paper a simplified polynomial is one that has had like terms combined. We consider a polynomial to be unsimplifiable if it has no terms that can be combined during polynomial simplication (there are many reducible polynomials whose factors when multiplied out do not have any like terms that can be combined). We make the distinction between simplified and unsimplifiable polynomials to avoid any misinterpretations by the reader that could occur in succeeding sections of this paper.

Representation of Polynomials

POLYFACT operates in a list processor environment with each cell consisting of two consecutive FORTRAN words (each word on the 1108 consists of 36 bits). The polynomials in POLYFACT are represented internally as a matrix of coefficients

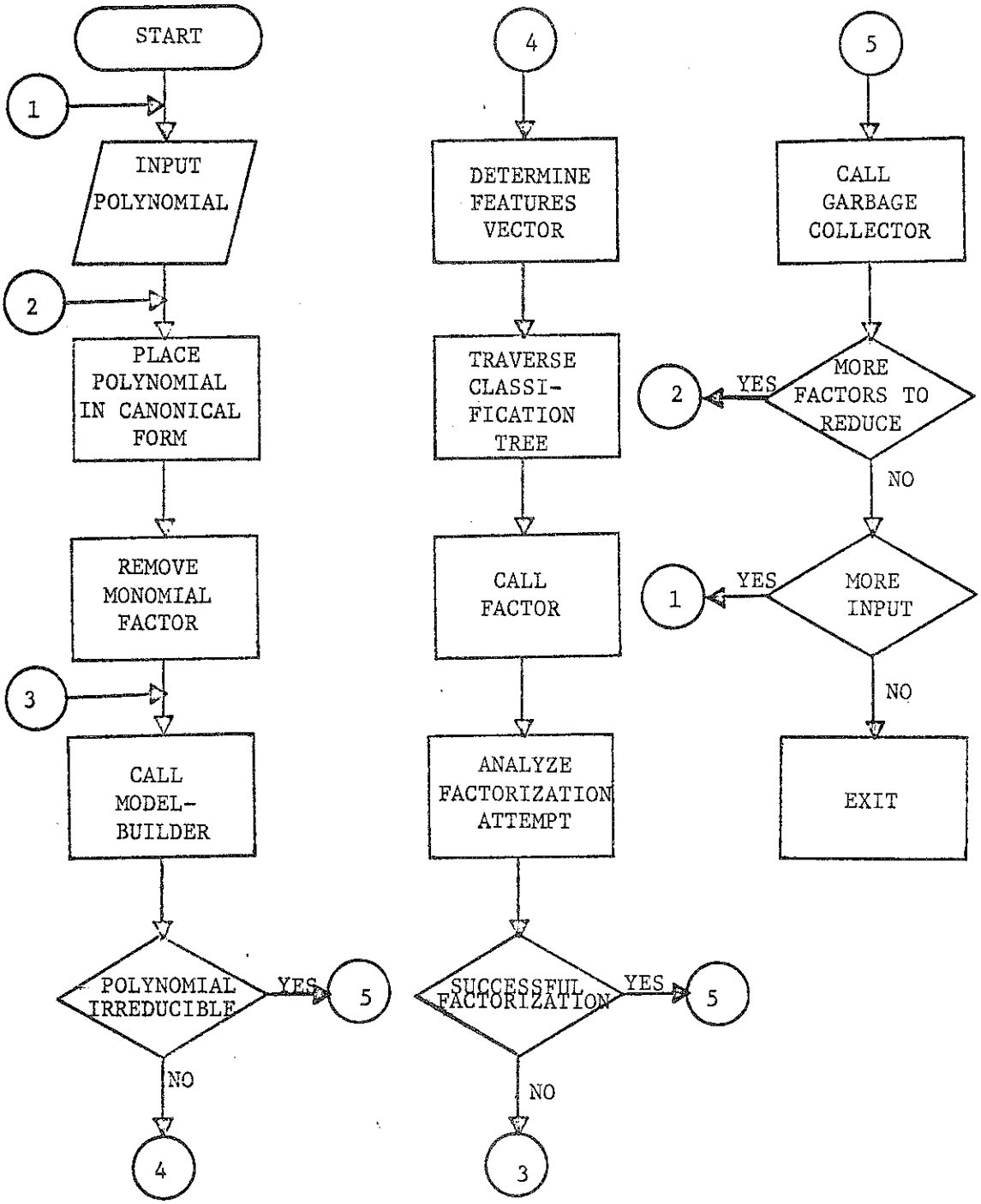

Fig. 1. Macro flow diagram of POLYFACT

and exponents. The matrix is represented by a list structure with all elements in a row linked together as a right-linked list and all elements in a column linked together as left-linked list. Each column vector in the canonical form corresponds to a term in the polynomial. The list representation of polynomials allows POLYFACT to accept polynomials with an arbitrary number of variables and terms. The internal representation of the polynomial $3x^2y+5xz-9yz^2$ is given in Fig. 2.

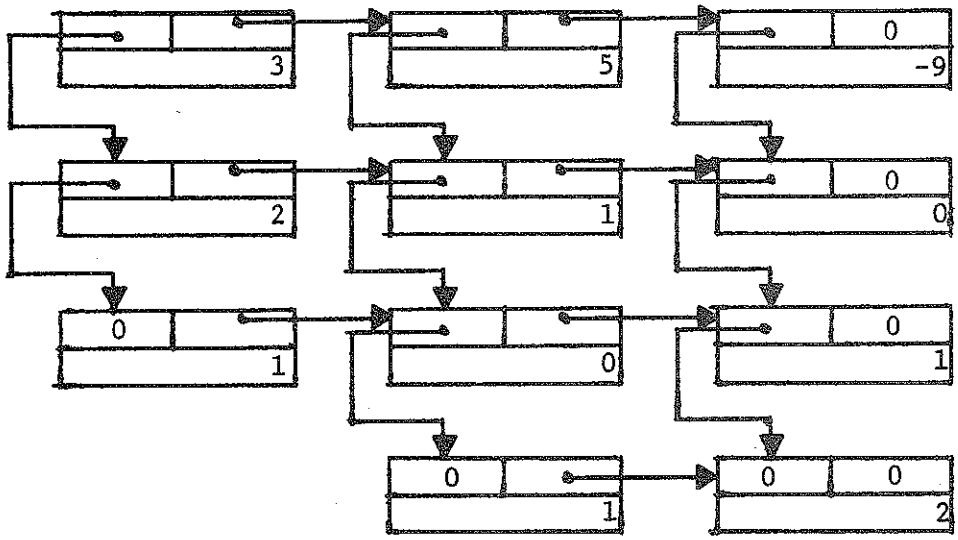

Fig. 2 Canonical form of $3x^2y+5xz-9yz^2$

Row one in Fig. 2 contains the coefficients of the terms in the polynomial; and rows two through four hold the exponents of variables x, y, and z, respectively. The zeros in the left and right links of some of the cells are list terminator indicators. We select this canonical form for its simplicity of representation, ease of locating specific terms in the poly-

nomial, and provision of the requirements of the model-builder.

Operation of POLYFACT

POLYFACT operates either in a training mode or learning mode. POLYFACT also learns in both modes. While in the training mode, the values of M and N and a simplified or unsimplifiable polynomial indicator are input. The user can train POLYFACT for as long as he deems necessary and then change the operation mode to the learning mode. POLYFACT can begin operation in the learning mode instead of training mode, but the training mode speeds up the rate of learning because less time is required to factor each polynomial. When POLYFACT is in the learning mode, no helpful information is given to it. The results of the learning process can be saved, and the training cycle omitted in subsequent uses of POLYFACT.

Overview of the Factorization Scheme

The factorization scheme implemented in POLYFACT relies on the fact that a reducible polynomial can be written as the product of two factors, one with M-terms and one with N-terms. During a factorization attempt the M-term factor is sought, and the N-term factor is determined by division of the M-term factor into the subject polynomial. Then both factors are saved and later reduced. POLYFACT attempts to minimize the amount of searching for the M-term factor by: (1) building a model for each polynomial, (2) using learning for term selection to initiate the factorization process, and (3) using learning to select term possibilities in the M-term factor.

Classification of Polynomials

POLYFACT classifies polynomials according to certain features that each exhibits (the features are given in Section IV). Through classification the capability exists for applying specific heuristics to a designated polynomial. Two types of features are used in classification: surface features and hidden features. Surface features are those features that can be determined by visible examination of the subject polynomial. Hidden features are those features not immediately visible to either a human or a pattern recognition program. The hidden features are detected during a factorization attempt, i.e. during the factorization of a polynomial characteristics are discovered that are not obvious from the initial examination. The detection of hidden features during a factorization attempt usually results in a reclassification of the polynomial unless the current factorization attempt is successful. The reclassification process is a powerful one since it provides the capability to automatically apply different sets of heuristics to a single polynomial during its factorization. The classification process is closely associated with the ability to dynamically create and modify heuristics [6].

The features (both surface and hidden) that a polynomial exhibits determine the heuristics used during a factorization attempt. In POLYFACT, the classification scheme is implemented as a binary classification tree. The classification tree is automatically constructed during program execution. The binary features vector of the polynomial is the address of a terminal node in the tree. This terminal node contains a pointer to the set of heuristics for this particularly classified polynomial.

The Model-Builder

A model [18] is created for each polynomial. The model is a means for reducing the amount of searching for a factor in a polynomial. There is no learning associated with the model-builder in POLYFACT. The model-builder tries to determine the values of M and N and the valid possibilities for terms in the M-term factor (the word "possibility" is used to denote terms that are candidates for terms in the M-term factor). If the factorization attempt is not successful with the current values of M and N then either the polynomial is reclassified and the process repeated with M and N unchanged; or the model-builder determines new values for M and N, and the process is repeated. A complete description of the model-builder is presented in Section IV.

The initial implementation of this factorization scheme in POLYFACT requires the explicit determination of the correct values of M and N before the factorization of a polynomial can be successful. This requirement is necessary because: (1) POLYFACT uses a heuristic division algorithm [5] that requires the correct values of M and N for a successful division, and (2) the determination of the M x N possibility list (discussed in Section IV) requires these two quantities. In Section V we provide another more operable method for determining the value of N. This new method frees the model-builder from the unenviable task of trying to determine the value of N prior to a factorization attempt. This modification also allows the replacement of the heuristic division algorithm by a conventional more efficient one.

Learning in POLYFACT

The learning schemes depend very heavily on the capability of POLYFACT to modify the heuristics dynamically. The importance of a representation for heuristics that allows for dynamic modification is discussed by Claybrook and Nance [6]. The heuristics in POLYFACT are represented in the first-order predicate calculus language [16] and are interpreted during program execution. Interpretation of the heuristics in POLYFACT is necessary since they are modified dynamically during program execution.

We have stated that the amount of searching for the M-term factor is reduced by using learning to aid in the selection of a term to initiate the factorization process and to select term possibilities for the M-term factor. The primary objective in term selection is to choose a term that leads to a small search space. The heuristic associated with directing learning in term selection utilizes the presumption that the term exhibiting the fewest number of possibilities leads to the minimum search space.

The learning associated with term selection is as follows. After a successful factorization attempt is complete, the number of possibilities in each term of the polynomial is determined. The features of the term(s) with minimum number of possibilities have their frequency count(s) increased (these features are given in Section IV). Then heuristics are constructed dynamically (and ordered) to reflect the importance of the features in selecting a term T to initiate the factorization process, i.e. if a feature has the highest frequency count, then all terms that do not have this feature are removed from consideration for T. The order of the heuristics for term selection can vary during program execution since POLYFACT adapts to the sequence of input polynomials. Learning associated with term selection can

also occur after an unsuccessful factorization attempt since the original terms in the polynomial are not changed regardless of a successful or unsuccessful factorization attempt. In fact, the learning associated with term selection can be primed prior to any attempts at factorization.

An example of a term selection heuristic in POLYFACT is the following:

H1.1 (E T IN S_0) ((N H1(G11(T),MINDEG) A N H1(G21(T),MINVAR) C FIX123)) $.

This heuristic with name H1.1 resets the use flag (the use flag indicates membership in a set) of each term T in set S_0 with degree (degree of T is the value of the function G11(T)) exceeding MINDEG and with number of variables (value of function G21(T)) exceeding MINVAR. The value of MINDEG is the degree of the minimum degreed term in S_0, and the value of MINVAR is the number of variables in the term with the minimum number of variables in S_0. H1 is a predicate, and FIX123 is a consequent that is executed if the antecedent

N H1(G11(T),MINDEG) A N H1(G21(T), MINVAR)

is satisfied. N, A, and C are the negation, conjunction, and conditional symbols, respectively.

The possibilities that can be selected as terms in the M-term factor are ranked according to their apparent merit in determining the correct M-term factor, and during a factorization attempt the highest ranked usable possibilities are selected. A usable possibility is one whose use flag is set. The features (given in Section IV) for possibility selection determine the rank of each possibility. After a polynomial has been factored, each of the terms in the M-term factor is examined to determine its set of characteristic features. A binary vector is created with nonzero entries indicating the features present. Then a heuristic is constructed in first-

order predicate calculus notation using as predicates those features characterizing the M-term factor.

To facilitate the construction of this set of heuristics, a matrix is maintained providing a history of the features of terms that have appeared in factors of previous polynomials. After the vector of features has been created for a term, it is compared with each row in the matrix to determine if the vector is already present. If so, the frequency count for the matching row is incremented. If the vector is not in the matrix, it is added and the corresponding heuristic is created.

When these heuristics are used to rank the possibilities in sets F_0 and F_1 (F_0 and F_1 are sets of possibilities described in Section IV), a binary vector is created with nonzero entries appearing in the vector positions corresponding to the predicates in the satisfied antecedent. When an antecedent is satisfied, the vector (created by the satisfied antecedent) is compared to each row in the history matrix. If the vector is in the matrix, the rank of the possibility is the frequency count, kept as an augmented column entry in the matched row; otherwise, the rank is zero. The learning associated with possibility selection can also be primed prior to any factorization attempts by inputting polynomials in factored form and analyzing the terms in the factors or simply by factoring polynomials in the training mode.

The possibility selection heuristics are maintained in complete predicate calculus notation and also in the encoded matrix form. The matrix form is convenient for determining the need for modifications to the heuristics. The term selection heuristics are kept in an encoded form for all classes of polynomials and then expanded prior to execution into first-order predicate calculus notation. The learning associated with term selection is within

classes of polynomials and learning associated with possibility selection carries over from polynomial to polynomial regardless of its classification.

The type of learning described above is generalized learning [18], [20], [21]. In Section V we show that learning is of considerable importance in improving the efficiency of operation of this factorization scheme. The composite of learning and polynomial classification provides a powerful mechanism for: (1) implementing localized learning for each class of polynomials, (2) the automatic extension of heuristics to previously unclassified polynomials, and (3) the dynamic construction and modification of heuristics.

Analysis of a Factorization Attempt

Regardless of failure or success, the results of each factorization attempt are analyzed. During this analysis, POLYFACT determines if the heuristics for term and possibility selection require modification and whether or not the polynomial warrants reclassification in the case of failure. If the user desires, learning associated with term and possibility selection can be ended after an appropriate period of time. Then heuristics are not modified after a factorization attempt until the learning indicator is reset.

Nature of the Heuristics in POLYFACT

The creation and modification of all the heuristics in predicate calculus notation are directed by the learning programs in POLYFACT. The set of heuristics determined by the classification scheme for a particular polynomial guides the actual factorization attempt. Each set of heuristics

consists of several subsets, with each subset having a specific function to perform. These subsets are responsible for:

(1) selecting a term T to initiate a factorization attempt,

(2) creating the set F_0 of all possibilities in the term T,

(3) ranking the possibilities in the set F_0 according to their probable merit in creating the M-term factor,

(4) selecting a possibility P from F_0 , where P is the first term in the M-term factor,

(5) creating the set F_1 of possibilities used to complete the M-term factor,

(6) ranking the possibilities in F_1 , and

(7) creating the remainder of the M-term factor (terms 2 through M) by the selection of terms from F_1.

III. RELATED WORK

Unlike POLYFACT, most factorization algorithms [2], [12], [19] and symbolic and algebraic manipulation systems [8], [9], [14], [15] have been implemented without any consideration for learning. We try to relate, in a succint manner, some of the previous research in polynomial factorization and learning schemes to the research described in this paper.

Factorization Algorithms

The classical algorithm for polynomial factorization is Kronecker's algorithm [12]. This algorithm has been implemented independently by Monove, Bloom, and Engleman [14] in their MATHLAB system, and by Johnson [11]. Jordan, Kain, and Clapp [12] describe a method for finding the symbolic factorization of multivariable polynomials by using a generalized version of Kronecker's algorithm. Later references to this work do not establish the implementation of the method.

Musser [19] has described algorithms for factoring polynomials with arbitrarily large integer coefficients. His algorithms are based on the use of modulo p factorizations and constructions based on Hensel's Lemma [23]. He has implemented algorithms for the univariate case and is currently implementing the algorithms for multivariable polynomials. Wang [24] has an efficient implementation of a version of Berlekamp's algorithm [2] which factors multivariable polynomials over the integers. His algorithm begins by making substitutions for all but one of the variables, producing a polynomial in just one variable. This univariate polynomial is then factored using Berlekamp's algorithm. After this factorization is done, the multivariable factors are recovered. Berlekamp's factorization algorithm has had such an impact on the factorization of polynomials that we outline it below.

Berlekamp's Algorithm

Let $f(x)$ be a univariate squarefree primitive polynomial. All arithmetic is done modulo q (q a prime). We want to determine r irreducible polynomials $p_i(x)$, $i=1, 2, \ldots, r$ such that

$$f(x) = p_1(x)\, p_2(x) \ldots p_r(x). \tag{1}$$

If (1) holds and $(s_1, s_2, \ldots, s_r)$ is any r-tuple of integers modulo q then by the Chinese remainder theorem, there is a unique polynomial $v(x)$, deg $v(x) <$ deg $f(x)$, such that

$$v(x) \equiv s_i \bmod (p_i(x)) \quad i = 1, 2, \ldots, r. \tag{2}$$

The Chinese remainder theorem applies since the $p_i(x)$ are irreducible and therefore relatively prime.

Information about the $p_i(x)$ can be obtained from solutions of (2). Since

$$s_i^q \equiv s_i \pmod{p_i(x)} \tag{3}$$

We find that

$$v(x)^q \equiv v(x) \bmod (p_i(x)) \quad i = 1, 2, \ldots, r. \tag{4}$$

Using (1) and (4) we get

$$v(x)^q \equiv v(x) \bmod (f(x)). \tag{5}$$

Since

$$x^q - x \equiv (x-0)\ (x-1)\ \ldots\ (x-(q-1)) \bmod q \tag{6}$$

by (3), it follows that

$$v(x)^q - v(x) = (v(x)-0)\ (v(x)-1)\ \ldots\ (v(x)-(q-1)). \tag{7}$$

If $v(x)$ satisfies (5) then every irreducible factor of $f(x)$ must divide one of the q relatively prime factors of the right-hand side of (7). Since deg $f(x)$ > deg $v(x)$, we have $\gcd(f(x), v(x)-s) \neq 1$ or $f(x)$, i.e. it is a nontrivial factor of $f(x)$. In fact

$$f(x) = \prod_{0 \leq s \leq q} \gcd(f(x), v(x)-s). \tag{8}$$

This shows that given any $v(x)$ satisfying (5), we can determine a nontrivial factorization of $f(x)$ using gcd's. Since fast algorithms for finding gcd's exist [4], [8], the factorization of $f(x)$ is fast when the $v(x)$ is determined.

There are q^r solutions of (5), and any solution of (5) must satisfy (3). Knuth [13] provides a method for finding solutions to (5). We outline briefly the steps for finding solutions to (5). Let deg $f(x) = n$; we can construct the $n \times n$ matrix

$$Q = \begin{bmatrix} q_{0,0} & q_{0,1} & \cdots & q_{0,n-1} \\ \vdots & & & \\ q_{n-1,0} & q_{n-1,1} & \cdots & q_{n-1,n-1} \end{bmatrix}, \tag{9}$$

where $x^{qk} \equiv q_{k,n-1}\ x^{n-1} + \ldots + q_{k,1}\ x + q_{k,0} \bmod (f(x))$. Then $v(x) =$

$v_{n-1} x^{n-1} + \ldots + v_1 x + v_0$ is a solution to (5) if and only if $(v_0, v_1, \ldots, v_{n-1})\ Q = (v_0, v_1, \ldots, v_{n-1})$ (Knuth [13] gives an algorithm for constructing Q). After Q has been constructed triangularize Q - I, where I is the n x n identity matrix, finding its rank n-r and r linearly independent vectors $v^{[1]}, v^{[2]}, \ldots, v^{[r]}$ such that $v^{[i]}\ (Q - I) = (0, 0, \ldots, 0)$ for i = 1, 2, ..., r. At this point, r is the number of irreducible factors of f(x).

The solutions to (5) are the q^r polynomials corresponding to the vectors $t_1 v^{[1]} + \ldots + t_r v^{[r]}$ for all choices of integers $0 \leq t_1, \ldots, t_r < q$. Next calculate gcd $(f(x), v^{[2]}(x) - s)$ for $0 \leq s < q$, where $v^{[2]}(x)$ is the polynomial represented by vector $v^{[2]}$ ($v^{[1]}$ represents the trivial solution to (5)). Then by (8) we have a non-trivial factorization of f(x). If $v^{[2]}(x)$ does not succeed in splitting f(x) into r factors, further factors can be obtained by calculating gcd $(v^{[k]}(x) - s, w(x))$ for $0 \leq s < q$ and all factors w(x) of f(x) found so far, for k = 3, 4, ... until r factors are obtained.

The algorithm outlined above works quite well when q is small; however, if q is large then (8) becomes a problem area because it becomes impractical to compute the gcd of f(x) and v(x) -s for each $s \in GF(q)$. Berlekamp's algorithm also suffers a loss in efficiency when the degree of the polynomial is large.

Berlekamps [2] discusses a method for dealing with (8) when q is large. This method was originally proposed by Zassenhaus [26]. The Zassenhaus algorithm transforms the problem of factoring f(x) over $GF(q^m)$ into the problem of finding the roots of a polynomial over GF(q). We outline Zassenhaus' approach for factoring polynomials f(x) over Z (rational integers) below.

Zassenhaus Algorithm

In general it is more difficult to factor a monic polynomial f(x) of degree n over Z. If C is the maximal absolute value attained by the coefficients of f(x), the coefficients of a factor $g(x) = x^m + b_1 x^{m-1} + \ldots + b_m$ satisfy the inequalities $|b_j| \leq \binom{m}{j} (C + 1)^j$ (j = 1, 2, ..., m) [27]. If these bounds are not too large for $m \leq \left[\frac{n}{2}\right]$, the factorization of f(x) over Z can be accomplished by picking a prime q which is greater than the maximal bound and by then factoring f(x) over GF(q). Zassenhaus used Hensel's Lemma [23] to develop an efficient means for factoring polynomial f(x) over Z. He choses any prime q not dividing the discriminate [13] d(f) and decomposes f(x) into two monic factors modulo qZ[x]. f(x) is irreducible over Z if it has no proper decomposition modulo qZ[x]. Otherwise, the decomposition modulo qZ[x] is raised to a modulus which involves a higher power of q, proceeding from

$$f(x) \equiv f_1(x)\, f_2(x) \mod (q^r Z[x]) \text{ to}$$
$$f(x) \equiv f_1'(x)\, f_2'(x) \mod (q^{2r} Z[x]).$$

Eventually, it will be possible for a suitable power of q to decide whether or not the decomposition of f(x) modulo $q^t Z[x]$ gives rise to a decomposition of f(x) over Z. The power of q in the modulus will quickly approach the above mentioned coefficient bounds. This procedure has to be carried on recursively for each of the two factors of f(x) over Z that results from the first step. Finally f(x) is decomposed into a product of irreducible factors over Z. The weakness of this approach occurs when f(x) has many more factors modulo qZ[x] for the prime q chosen than it has over Z itself.

The Berlekamp-Hensel algorithm (e.g. Wang's algorithm) does particularly poorly in one case: the "bad zero" case. This case occurs when a substitution

of zero for all but one of the variables causes the leading coefficient to go to zero. The heuristic algorithm described in this paper has few problems with this type of polynomial. More will be said about this in Section V.

Cost of Factorization in Berlekamp-type Algorithms

Musser [19] has shown that Berlekamp's 1967 algorithm [3] is dominated by $n^3L(q)^2 + n^2L(q)^3 + n^2rqL(q)^2$. The L operator is defined as the number of digits in q (L(q) can also be interpreted as log (q)). In the multivariate case it is Hensel's Lemma [23] that dominates in computing costs, not Berlekamp's algorithm. Musser has shown that the time for application of Hensel's algorithm, to univariate polynomials, is dominated by $n^2L(m)^3 + nL(m)^2\ L(c)$, where $n = \deg f(x)$, $m > 2B$ (B is a bound on the coefficients of the divisors of f(x)), and $c = |f(x)|_1$ ($|f(x)|_1$ is the sum of the moduli of the numerical coefficients of f(x)). Comments on the cost of the factorization scheme described in this paper appear in Section V.

Learning Programs

In many cases learning has been studied in a simple environment [17], [20], [22] so that more attention can be paid to learning schemes than to the problem environment. We limit the discussion of learning here to a few generalized learning [21] schemes. Generalized learning has been used more frequently by researchers in Artificial Intelligence than any other learning technique. The reason for their frequent use seems to be twofold: (1) humans tend to use generalized learning in their learning habits, and (2) other learning schemes, e.g. rote learning [21] and concept learning [10] are applicable only in certain instances.

Generalized learning has usually appeared in the form of modifying weights in an evaluation function, e.g. in Samuel's checker program [21], Michie and Ross's Graph Traverser [17], and Slagle and Farrell's MULTIPLE program [22]. Waterman [25] has presented some generalized learning techniques in his poker playing program. His program generalizes by modifying heuristics, represented in production form, to catch state vectors that represent game situations.

This discussion of learning programs is brief; the reason being that very little use has been made of learning in solving practical problems. We hope this paper will convince the reader of the importance of learning as a problem solving tool.

IV. DETAILS OF THE FACTORIZATION SCHEME

In Section II the reader was introduced to the basic ideas behind this factorization scheme. In this section a detailed description of the scheme is given. Before describing the factorization scheme, we point out that not only is learning itself used to reduce the amount of searching for the factors of a polynomial, but other heuristics are employed as well, e.g. creation of the M x N possibility list and its use in creating the set F_0, the g.c.d. constraint, etc. The reader should be able to pick out these heuristics in this section.

Responsibilities of the Model-Builder

The model-builder is the most important part of this factorization scheme. In addition to determining the values of M and N ($M \leq N$), the model-builder also:

(1) creates the M x N possibility list (this list contains all the possibilities that can be tried as terms in the M-term factors), (2) decides if the polynomial is simplified, and (3) determines whether or not the polynomial is irreducible.

If a polynomial can be factored into an M x N factorization, then the terms in the M-term factor must be in the M x N possibility list created by the model-builder. If a polynomial is reducible and has a term in the smallest factor (the M-term factor) that is not in the possibility list, then the polynomial cannot be factored into an M x N factorization, i.e. the values of M and/or N are incorrect.

Calculation of M and N

The values M and N are the two most important quantities determined by the model-builder. They are the basis for determining the possibility list mentioned above. Few features of a polynomial give any insight as to their values. The approach to determining M and N in the initial implementation of POLYFACT is the following (another more operable approach is described in Section V):

1. Try to detect whether or not the polynomial is a simplified polynomial.

2. If it is not possible to determine that the polynomial is a simplified polynomial then assume that it is an unsimplifiable polynomial.

3. Using NTERMS, the number of terms in the polynomial, determine all possible combinations of M and N such that $M \leq N$ and N=NTERMS/M (M must divide NTERMS).

4. Try all possible values of M and N as determined by 3 above until factorization is successful, or the pairs of values are exhausted.

5. If factorization has failed as outlined in 4, then assume that the polynomial is a simplified polynomial.

6. Once POLYFACT has assumed that the polynomial is a simplified polynomial, it uses some more surface features to determine how the terms were added out or combined.

7. If the coefficients are all unity, then terms are added out in pairs.

8. If some or all of the coefficients are greater than unity, then some of the terms can be added out in pairs or may simply be combined into one term.

9. Using 7 and 8, POLYFACT decides whether to increment NTERMS by 1 or 2 and then proceeds as in 3 and 4, respectively.

Each time the value of M and/or N changes, a new M x N possibility list is created.

The success or failure of determining the factorization of a polynomial depends on the ability to determine the correct values for M and N. The determination of M and N depends on how many terms were in the polynomial prior to being placed in canonical form. Some types of polynomials, e.g. repeated factor, difference of powers, or single variable polynomials, usually have a large number of the terms combined during simplification. However, in many instances these three types of polynomials can be classified as special ones and the appropriate heuristics applied. If a polynomial cannot be classified as a special polynomial or an attempted factorization of it as a special polynomial fails, then factorization proceeds in the normal manner outlined in this paper. Repeated factor and difference of power polynomials are the only special case polynomials considered by this technique. In some instances substitution is necessary to place the polynomial in a form recognizable as a special polynomial. Polynomials 10 and 12 in Appendix A are examples of repeated factor polynomials. Polynomial 13 is a difference of power polynomial.

The M x N Possibility List

As we stated previously, the M x N possibility list is the list of valid possibilities that can be tried as terms in the M-term factor. All possibilities appear in this list with implicitly defined unit coefficients. This consideration saves considerable memory in storing the possibility list and is a valid approach because POLYFACT has not yet created the F_0 and F_1 sets. These sets are much smaller than the M x N possibility list, and the possibilities in them are the only ones that require the coefficients to be determined. This discussion becomes much clearer when we discuss the creation of the F_0 and F_1 sets.

We point out now that one reason the model-builder tries to specify the correct values of M and N is that the size of the M x N possibility list decreases as N increases. Thus, in most instances the size of the M x N possibility list decreases during a sequence of factorization attempts on a polynomial, thereby decreasing the search space for the terms in the M-term factor. The suggested modification to the determination of N in Section V removes, to a certain extent, this advantage; but it simplifies the model-builder's tasks considerably.

The procedure for determining the M x N possibility list is described below. The first possibilities to be placed in the possibility list are the single variable possibilities. For each variable x in the polynomial a sequence of quantities t_x^i is calculated. The quantity t_x^i is the number of terms in the polynomial in which x^i ($1 \leq i \leq \text{maxdeg}(x)$) appears as a factor. The quantity $l_x^i = (t_x^i - K \cdot N)/(M-K)$, $1 \leq K < M$, where K is the number of terms in the M-term factor in which x^i can appear, is calculated for each x^i. If $l_x^i \geq 0$, then x^i is a valid possibility. An interpretation of l_x^i is that

it is the number of terms in the N-term factor in which x^i appears as a factor. After all single variable possibilities have been determined and entered into the M x N possibility list, the multivariable possibilities, i.e. possibilities that contain more than one variable, are determined. The above procedure for determining single variable possibilities is generalized to multivariable possibilities. The prospective multivariable possibilities are determined by forming products, in a systematic manner, of the single variable possibilities already in the M x N list. Every such product of single variable possibilities is not neessarily a valid possibility, i.e. if x^i, where x^i is now a multivariable quantity, does not satisfy the requirement that $1_x{}^i \geq 0$, then it is not included in the M x N possibility list.

When a possibility is added to the possibility list, the quantities $t_x{}^i$, $1_x{}^i$, K, and an indicator as to whether it is a single variable or multivariable possibility become a part of the possibility entry. Some of these quantities are used later in ranking possibilities.

The Factorization Process

The reader may want to refer back to Fig. 1 as he begins reading this section. The material discussed here is associated with subroutine FACTOR in Fig. 1, and the heuristics used during a factorization attempt. Fig. 3 describes the operation of FACTOR during a factorization attempt.

The set of heuristics determined by traversing the classification tree guides the actual factorization of a polynomial. The features used in POLYFACT to classify polynomials are the following:

1. Coefficients are units.
2. The polynomial is a simplified polynomial (versus an unsimplifiable one).

3. Set F_1 is empty.
4. The number of possibilities in set F_1 is less than (M-1) but greater than zero.
5. Single variable polynomial.
6. Difference of powers polynomial.
7. Repeated factors polynomial.

Features 2, 3, and 4 are hidden features.

Initially, all terms in the polynomial are members of the set S_0. Each term in S_0 is a candidate for the term T[1] selected to initiate a factorization attempt. The following features of terms are used to induce learning for term selection as described in Section II:

1. Degree of the term.
2. Number of variables in the term.
3. Size of the coefficient.
4. Is the coefficient prime?

The term T is selected from the terms remaining in S_0 after the term selection heuristics are applied. Another set S_1 is created that consists of all terms in the polynomial with the exception of T. The terms in S_1 are those available for creating the set F_1.

The set F_0 is created next as indicated in Fig. 3. F_0 consists of all the valid possibilities in T. F_0 contains those factors of T that are in the M x N possibility list. In the process of creating F_0, the coefficient of T is used to include in F_0 not only possibilities with unit coefficients; but all other valid possibilities (that are factors of T) with coefficients dividing the coefficient of T. Note that we have to include in F_0 only possibilities with positive coefficients.

---

[1] T is used to indicate the term selected to initiate a factorization attempt.

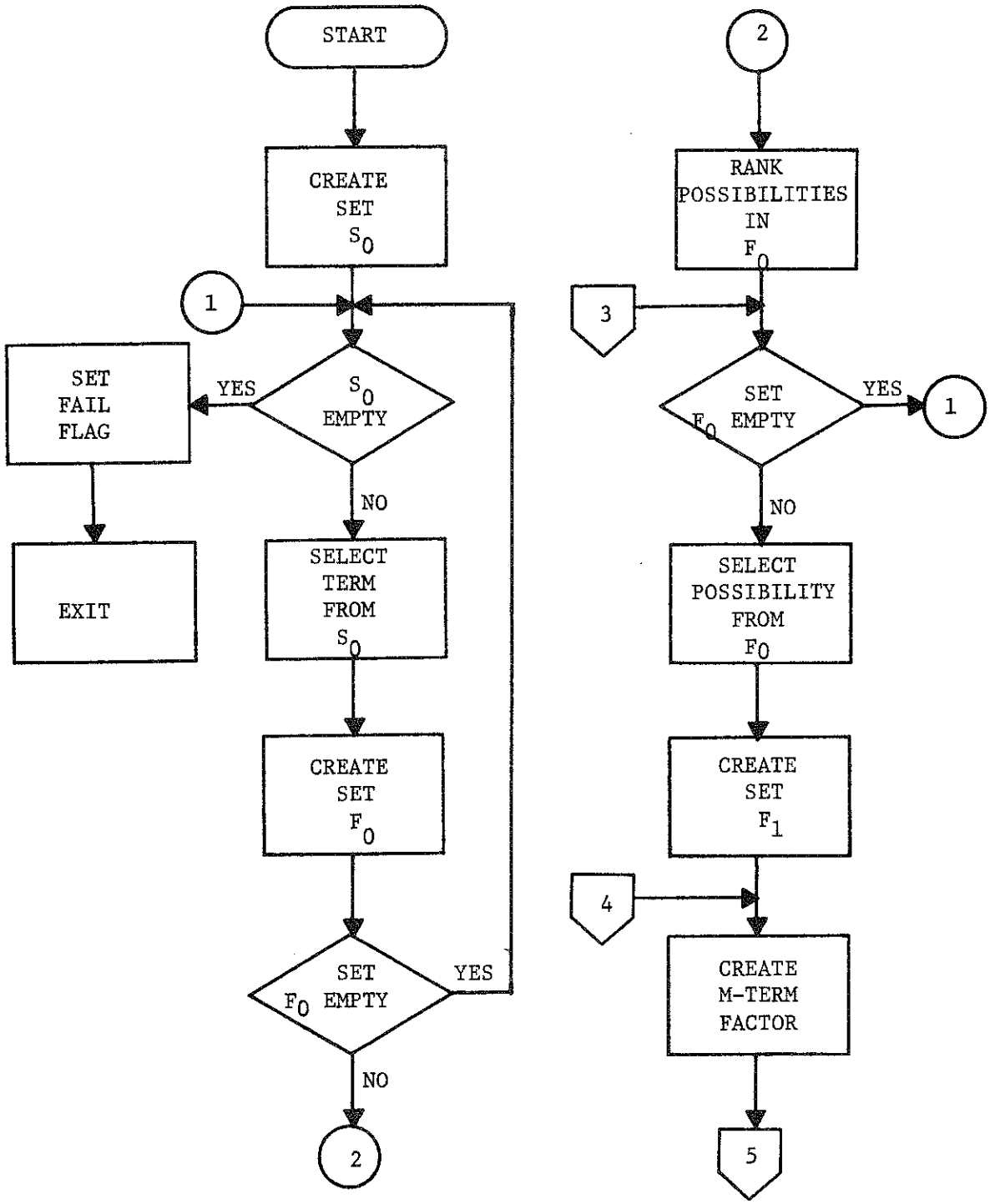

Fig. 3. Flow diagram of subroutine FACTOR

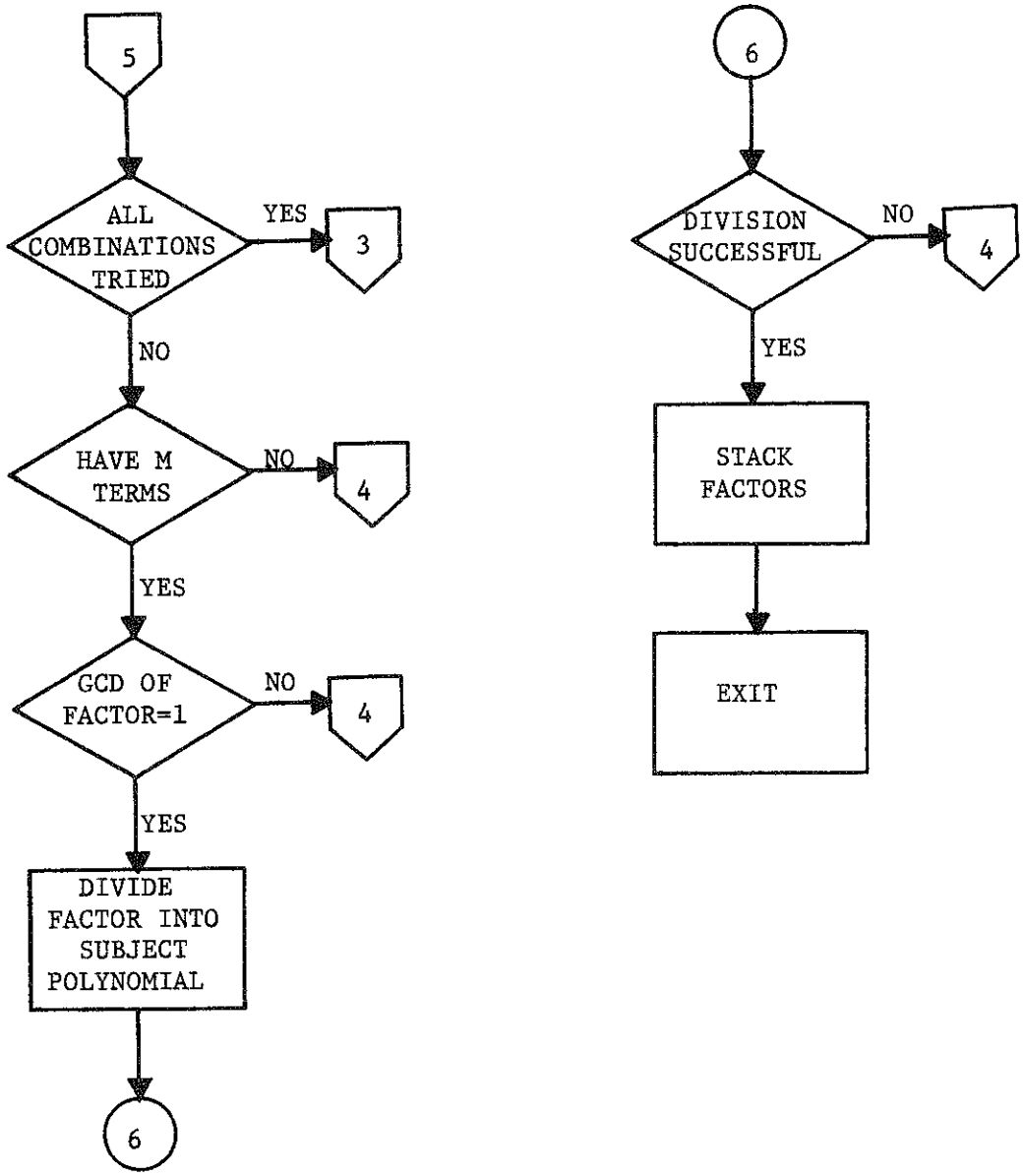

Fig. 3. Continued

The possibilities in F_0 are ranked according to their merit (the ranking process is described in Section II). The features used to rank the possibilities are the following:

1. Is the possibility a single variable possibility?
2. Does the possibility have a low t_x^i value?
3. Does the possibility have a high t_x^i value?
4. Are all the coefficients in the polynomial units?
5. Is the coefficient of the possibility a unit?
6. Is the term T a multivariable term?

The highest ranked usable possibility P in F_0 is selected as a candidate for the first term in the M-term factor and flagged so that it cannot be selected again. A second set of possibilities F_1 is created using P. The set F_1 consists of all those possibilities TP determined by TP=T'/TF where: (1) T' is any term in S_1, (2) TF=T/P, and (3) TP is a valid possibility. For a possibility to be considered a candidate for a valid possibility for F_0 and F_1, its variable part must be contained in the M x N possibility list. The set F_1 contains all those possibilities that are candidates for terms two through M in the M-term factor. The possibilities in F_1 are ranked using the same heuristics used to rank the possibilities in F_0.

An M-term factor is formed by selecting the highest ranked usable term from F_0 and the rest of the terms from F_1. If the greatest common divisor of the terms in this newly formed M-term factor is not one, then another M-term factor is formed. When the greatest common divisor constraint is satisfied, the M-term factor is divided into the subject polynomial. If the division is successful, the quotient is the N-term factor; otherwise, another combination of possibilities is selected to create another M-term factor, and the process is repeated. The greatest common divisor constraint is imposed to reduce the

number of possible factors that can be generated for the M-term factor. The use of this constraint is justified because any monomial factor is removed from the polynomial before the M-term factor is sought.

If all of the possibilities in F_0 have been selected without a successful factorization attempt and there are no more terms in S_0 (S_0 after the term selection heuristics are applied), then the model-builder decides the fate of the polynomial. If a factorization attempt is successful, both factors are saved and processed.

Because a record is maintained of all combinations of possibilities tried for a given factorization attempt, the construction of a previously formed factor is avoided. In POLYFACT, a possibility search tree (depth-first) is used to record each combination of possibilities as it is tried. When a new factor is formed, the tree is searched to determine if it has previously been used. This tree can be used later in the analysis of a factorization attempt.

The irreducibility of a polynomial is determined by the model-builder. This is the most difficult problem associated with polynomial factorization. The author has found no surface features of polynomials that give a hint to the solution of the problem. For unsimplifiable polynomials, the scheme described in this paper can easily determine whether or not a polynomial is irreducible by simply trying all valid M x N factorizations with $M \leq N$. A simplified polynomial is considered irreducible by POLYFACT when the M x N possibility list has less than M possibilities in it. The M x N possibility list decreases in size as M and/or N increase in size. We suspect that new features discovered to improve the detection of irreducibility will be hidden features. Note that we mentioned earlier that a few reducible polynomials can fail to be factored by POLYFACT (these are described later in this section);

hence, POLYFACT will consider these to be irreducible polynomials. This is a deficiency that Wang's algorithm can handle since his algorithm does not terminate until it has proven that it has found all irreducible factors.

A Very Simple Example

We demonstrate the factorization process by factoring the following simple polynomial (this polynomial happens to be unsimplifiable):

$$x^3y^2-x^2y^4-x^4z+x^3y^2z-xy^3z+y^5z+x^2yz^2-xy^3z^2.$$

We choose this simple polynomial to keep the M x N possibility list and the F_0 and F_1 sets small.

The above polynomial has no monimial factor. The model-builder determines the initial values for M and N (M=2 and N=4) and the possibility list: $\{x, x^2, y, y^2, z, xy, x^2y, xy^2, xz, yz, y^2z\}$. The features vector for the polynomial is the ordered set {1,0,0,0,0,0,0}. The only feature that POLYFACT detects on the initial classification of this polynomial is that the coefficients are units; hence, component one in the features vector is set.

Initially, S_0 contains all eight terms in the polynomial. Next suppose the term selection heuristics determine that the term $-x^4z$ is T. Set S_1 now consists of $\{x^3y^2,-x^2y^4,x^3y^2z,-xy^3z,y^5z,x^2yz^2,-xy^3z^2\}$. The set of possibilities in T, F_0, is $\{x,x^2,z,xz\}$. Suppose that x has the highest rank among the four possibilities in F_0. The set F_1, i.e. the set of possibilities determined by calculating TP=T'/TF, where T' is any term in S_1 and $TF=T/P=-x^4z/x=-x^3z$, contains only $-y^2$ since y^2 is the only possibility TP in the 2 x 4 possibility list (the coefficient of TP is ignored when determining whether it is in the possibility list).

At this point we have x as the first term in the 2-term factor. Next we select the highest ranked usable possibility from F_1 to create the 2-term factor, $(x-y^2)$. The greatest common divisor of this 2-term factor is unity, so we divide $(x-y^2)$ into the subject polynomial. The division is successful resulting in the 4-term factor $(-x^3z+x^2y^2+xyz^2-y^3z)$. The 2-term and 4-term factors are stacked for later reducing and control is returned to the main program so the factorization analysis can be carried out.

Problems Associated with Simplified Polynomials

Three types of problems arise in treating simplified polynomials:

1. determination of the correct values for M and N (the suggested modification for calculating M and N in Section V removes this problem entirely),
2. selection of a term to initiate factorization, and
3. creation of the M-term factor necessary for factorization to be successful.

Term selection for a few simplified polynomials presents a problem because POLYFACT may select a term that has been formed by the addition of two or more terms during polynomial simplification. If POLYFACT selects such a term, then the factorization attempt may fail since the correct M-term factor can never be created. However, in many instances there exists more than one term (left in S_0 after the term selection heuristics have been applied) satisfactory for initiating a factorization attempt, and this problem can be avoided by selecting another term.

The creation of the M-term factor necessary for factorization is the most serious and difficult problem associated with simplified polynomials. To clarify this discussion, remember that the M-term factor is created by selecting the first term from F_0 and the remaining (M-1) terms from F_1.

In many instances a sufficient number of terms for creating this factor are present in the polynomial and no terms have to be added. Whenever terms must be added to the polynomial to create the correct M-term factor for successful factorization, POLYFACT adds them in pairs. POLYFACT adds terms only when the size of set F_1 is less than (M-1). Added terms are placed in S_1, and the corresponding possibilities are placed in F_1. The new terms in S_1 have their use flags turned off until the corresponding possibility in F_1 is chosen as a term in the M-term factor. The new possibilities in F_1 are assigned rank zero so that those already in F_1 are selected first.

If the size of F_1 is less than (M-1) but not zero, POLYFACT assumes those possibilities already in F_1 are correct and necessary for construction of the M-term factor. The new possibilities in F_1 are selected in serial order to determine efficiently all the possible combinations that can be used to create the M-term factors. If the size of F_1 is zero, then all possible combinations of the new possibilities are tried until success or failure occurs. In this case POLYFACT forms all possible combinations of the possibilities in F_1 in normal fashion, i.e. the possibilities are ranked, and the highest ranked ones are selected first.

The terms missing in polynomials that have unit coefficients are added in by selecting possibilities from the M x N possibility list and forming the product with TF (TF is described above and is the first term in the N-term factor). The terms missing in polynomials with coefficients larger than one are determined by dividing TF into every term in the polynomial not associated with a possibility already in F_0 and F_1, i.e. the usable terms in S_1. If the quotient (ignoring coefficients) F=T'/TF is a possibility in the M x N possibility list, then the term T_0=F*TF is a candidate for a missing term; however, the coefficient of F, and hence T_0, may be incorrect.

In general, T_0 is not the same as T'; however, the variable part of each term is identical. The coefficients of F and T_0 are reconstructed by POLYFACT. POLYFACT initializes the coefficient of F to zero. When F is selected from F_1 as a possibility for the M-term factor, its coefficient is incremented until it divides at least two usable terms in S_1. When the coefficient of this possibility becomes so large that the possibility has no chance of dividing at least two usable terms in S_1, it has its coefficient reset to zero and another possibility is selected from F_1.

Whenever POLYFACT selects a new possibility from F_1 that is associated with an added term, the use flags of the corresponding pair of new terms in S_1 are set. Then POLYFACT checks the usable terms in S_1 to determine which of these terms can be combined with a new term. POLYFACT uses this technique to reconstruct the polynomial as it existed before simplification occurred.

V. RESULTS

This section provides some of the results of factoring over 300 polynomials using POLYFACT. Most of the 300 plus polynomials factored by POLYFACT are random polynomials with number of terms (after being placed in canonical form) [2,84], number of variables [2,5], degree of each variable [0,12], and coefficients [-10000,10000]. The random polynomials are generated by generating factors randomly, multiplying them together, and then simplifying the expanded form. The random factors are developed by generating terms randomly by selecting the number of variables, degree of each variable, and coefficient from the range of values for each given above. Appendix A contains a subset of these polynomials. Space (the 300 plus polynomials in factored form occupy 20 typed pages) does not allow us to include in Appendix A all of the polynomials

factored by POLYFACT during this research; therefore, in some instances we provide tables to illustrate important results without giving all the polynomials associated with them, e.g. Table I. Tables I - IV demonstrate the importance of learning to the efficiency of operation of POLYFACT. An analysis of variance experiment is also described, and the sources located that tend to influence the factorization times.

Results Associated with Learning

These results were obtained to demonstrate that the capabilities of POLYFACT are increased through learning. Several approaches could be taken to assess the influence of learning on program performance or program efficiency. By efficiency we mean efficiency with respect to execution time and memory used. An obvious way to measure program efficiency through learning in POLYFACT is to factor several polynomials with and without the use of learning and then measure the size of the possibility search tree (Section II) in each case. With this approach program efficiency is measured by comparing the size of the possibility search tree created using learning with that created without learning. This is characteristic of past approaches to evaluating learning [17]. While the above approach is valid for evaluating some programs, we do not believe that is is satisfactory for POLYFACT since it is only a gross measure of learning. Instead, we choose an approach that illustrates explicitly the effectiveness of POLYFACT in selecting both terms and possibilities.

Term Selection Results

The term selection tests consist of factoring several sequences of

randomly generated polynomials and determining for each polynomial: (1) the minimum number of possibilities in a term, (2) the maximum number of possibilities in a term, and (3) the number of possibilities in the term selected. We consider that POLYFACT selects "good" terms when it selects terms with a minimum number of possibilities. The term selection tests are also used to determine whether or not POLYFACT can adjust the order of the term selection heuristics for randomly generated polynomials with different ranges of values for size of coefficients, degree of variables, etc.

Table I contains the results of selecting terms for one sequence of random polynomials. The random polynomials for this sequence have the characteristics: coefficients [-64,64], [-400,400], [-2500,2500], degree of variables [0,8], [0,12], [0,12], and number of variables [2,4], [2,5], [2,5], respectively. The first six polynomials in this sequence are the training sequence. To conserve space, the random polynomials in this sequence are not included in Appendix A.

A summary of the results of the term selection tests is as follows:

1. The term selected to initiate factorization in 73 percent of the polynomials has the minimum number of possibilities (the percentage for the sequence in Table I, disregarding the training sequence, is about 67 percent).

2. POLYFACT orders the term selection heuristics depending on the characteristics of the polynomials (the order depends on the range of values for number of variables, degree of variables, etc.)

3. Term selection does increase the efficiency of POLYFACT considerably by allowing, in most cases, for as small a search space as possible.

For the sequence of polynomials in Table I, the number of variables is the most important feature for selecting a term with minimum number of possibilities with the degree of term and the size of coefficient of almost equal importance. Another sequence of polynomials could cause POLYFACT to order the term selection

TABLE I

TERM SELECTION FOR RANDOM POLYNOMIALS

Polynomial	Minimum Number of Possibilities In A Term	Maximum Number of Possibilities In A Term	Number of Possibilities In Term Selected
3	11	36	20
2	20	47	30
17	5	24	6
2	20	47	25
18	7	14	7
19	4	13	4
99	2	115	2
112	84	921	104
100	36	1139	36
135	25	160	25
179	14	38	14
180	2	201	2
181	14	1014	14
182	19	166	19
183	11	54	24
184	12	75	14
185	4	53	4
186	51	305	54
187	12	174	12
70	8	26	8
188	30	459	30
189	75	468	91
190	15	190	15
191	5	304	5
192	20	204	30
193	23	230	25
194	17	294	17
195	18	130	18
196	21	149	21
197	6	379	6
198	17	631	34
199	12	146	12
200	27	1140	42
201	6	241	12
202	32	489	32
203	24	398	24
204	30	645	30

heuristics differently. Learning allows POLYFACT to adjust the order of heuristics so that a term with the minimum number of possibilities is found in most cases. The selection of a term is very important because the analysis of variance results show that the number of possibilities in a term is one of the two main contributors to the factorization time for a polynomial.

Possibility Selection Results

We associate a value that we term <u>efficiency</u> with possibility selection. The efficiency value is an assessment of the capability of POLYFACT to learn to rank the possibilities. Before we define efficiency the following four definitions are needed:

> The <u>possibility search space</u> is the set of possibilities in the M x N possibility list.
>
> The <u>actual possibility search space</u> is the set of possibilities generated by the term selected to initiate the factorization process.
>
> Let $r_1, r_2, \ldots, r_n$ be the $n(n \geq 1)$ values for the ranks of the possibilities in the actual search space, and suppose the possibility selected from F_0 that leads to a successful factorization has rank $r_i (1 \leq i \leq n)$. Then the <u>reduced possibility search space</u> consists of all those possibilities with rank equal to or exceeding r_i.
>
> The size of a possibility search space is the number of possibilities in the search space and is denoted by $\| \text{ possibility search space } \|$.

The efficiency with respect to possibility selection is defined as:

$$\text{Efficiency} = \frac{\| \text{ actual search space } \| - \| \text{ reduced search space } \|}{\| \text{ actual search space } \|}$$

The efficiency values shown in Table II are given as a ratio to explicitly show the size of the actual search space (denominator) and the number of possibilities in the actual search space that are not members of the reduced search space (numerator). The efficiency value in decimal form allows the reader to ascertain that efficiency increases with experience. All

possibilities with the same rank have an equally likely chance of being selected as a term in a factor.

The possibility selection tests consist of factoring different sequences of randomly generated polynomials and determining the efficiency values for each polynomial. The polynomials in Tables II - V are in Appendix A. We show that POLYFACT learns to select the "good" possibilities by demonstrating that the efficiency values tend to increase for successive factorizations of randomly generated polynomials. An asterisk (*) to the right of the efficiency values in Table II indicates that only the highest ranked possibilities are in the reduced possibility search space.

Table II is one of three tables used in Claybrook [5] to gather the following results:

1. The size of the actual search space is decreased usually by 50 to 80 percent through learning.

2. As POLYFACT gains in experience it becomes more discriminant in the ranking of possibilities, i.e. some polynomials have several different ranks for the possibilities in their actual search spaces.

3. In most cases the reduced search space consists of only the highest ranked possibilities.

Result (3) and Table II show this factorization scheme could be implemented to consider a factorization attempt to fail when all of the highest ranked possibilities in F_0 have been used as the first term in the M-term factor, i.e. the possibilities with ranks lower than the highest rank are never considered. Then usually only 20 to 50 percent of the possibilities in F_0 would even be considered. But this consideration may cause a few reducible polynomials, e.g. polynomial 39 in Table II, to not be factored because a lower ranked possibility in F_0 led to a successful factorization. This idea requires that a proper training sequence of polynomials be input to prime the learning associated with possibility selection.

TABLE II

EFFICIENCY VALUES

SEQUENCE #1		SEQUENCE #2	
Polynomial	Efficiency	Polynomial	Efficiency
20	0(.00)	20	0(.00)
2	0(.00)	2	0(.00)
21	6/12(.50)	45	1/2(.50)*
22	2/8(.25)*	46	0/18(.00)
23	1/6(.17)	47	0/46(.00)
24	1/3(.33)*	48	2/8(.25)*
25	3/4(.75)*	49	17/36(.47)
26	6/14(.43)*	50	1/12(.08)*
27	5/14(.36)*	51	10/47(.21)
28	2/6(.33)	52	60/100(.60)*
29	11/16(.69)*	53	37/60(.62)*
30	5/17(.29)	54	3/11(.27)*
31	4/15(.27)*	55	17/25(.68)*
32	5/11(.45)*	56	71/99(.71)*
33	22/28(.79)*	57	34/45(.76)*
34	13/22(.59)*	58	81/100(.81)*
35	30/49(.61)*	59	5/7(.71)*
36	23/53(.43)	60	25/33(.75)*
37	3/6(.50)*	61	47/69(.68)*
38	6/10(.60)*	62	9/10(.90)*
39	4/8(.50)		
40	44/56(.79)*		
41	7/19(.37)*		
42	5/6(.83)*		
43	21/24(.87)*		
44	50/60(.83)*		

Recurrent Factorization Results

Another test that further demonstrates POLYFACT's learning and factorization capabilities is the recurrent factorization test. This test consists of factoring several sequences of polynomials, generated by the author, that have the same polynomial appearing more than once in a single sequence.

We maintain that learning occurs in POLYFACT when the factorization times decrease for subsequent factorizations of the same polynomial. The purpose of Tables III and IV is to show that POLYFACT learns from previous experience. Tables III and IV show that POLYFACT usually factors the second and third occurrence of the same polynomial in a single sequence in less time than for the initial factoring.

An analysis of Tables III and IV is summarized briefly:

1. the order of the polynomials in the sequences influence the factorization times, e.g. polynomials 2 and 9 in sequences 3 and 4,

2. after two or more factorizations of the same polynomial in a given sequence, further reduction in the factorization time is usually quite small,

3. POLYFACT is capable of factoring difficult polynomials, e.g. polynomials 6 and 8, and

4. subsequent factorizations of the same polynomial indicate the reduction in factorization time is markedly significant (on the order of 100 to 300 per cent).

Some Comparisons with Wang's Algorithm

We stated previously that Wang's algorithm [24] for polynomial factorization is an implementation of Berlekamp's algorithm with some variations. The Wang algorithm is implemented in LISP 1.5 on a PDP-10 at M.I.T. as part of the MACSYMA system [15]. We do not compare the algorithms themselves because they are significantly different approaches to polynomial factorization; and hence, there are few areas for comparison. However, we provide factorization results of some multivariable polynomials factored by POLYFACT and Wang's algorithm. Table V shows the factorization times for these polynomials.

TABLE III

RECURRENT FACTORIZATIONS

SEQUENCE #1		SEQUENCE #2	
Polynomial	Factorization Time (seconds)	Polynomial	Factorization Time (seconds)
1	2.48	3	2.32
2	16.36	7	16.34
8	864.72	7	11.88
6	781.26	7	11.88
2	8.02	9	222.74
8	208.34	9	221.74
6	381.20	9	174.66
2	7.94	4	53.94
8	208.00	4	29.12
6	380.08	4	28.00
		5	14.22
		5	4.48
		5	4.48

TABLE IV

RECURRENT FACTORIZATIONS

SEQUENCE #3		SEQUENCE #4	
Polynomial	Factorization Time (seconds)	Polynomial	Factorization Time (seconds)
3	2.32	3	2.32
63	169.44	2	12.50
9	554.84	64	112.52
63	95.16	65	9.94
9	174.62	2	11.14
63	95.16	64	102.82
9	174.34	65	11.86
64	180.64	2	6.86
2	24.28	64	49.42
16	17.14	65	11.86
64	126.30	5	16.98
2	19.96	7	32.16
16	10.06	9	222.68
64	50.02	5	15.62
2	14.08	7	24.72
16	10.06	9	221.46
		5	11.68
		7	24.72
		9	174.62

TABLE V

COMPARISON OF FACTORIZATION TIMES (IN SECONDS)+

Polynomial	POLYFACT (with learning)	Wang's Algorithm
9	174.66	**
12	6.85	6.74
16	10.06	**
67	149.26	**
68	160.03	**
69	172.16	**
70	1.97	1.85
71	25.38	23.83
72	67.49	76.37
73	129.01	476.11

+The reader should note that the 1108 is approximately three times as fast as the PDP-10.

The entries in Table V that contain asterisks (**) indicate that the memory allocation of the PDP-10 at M.I.T. has to be increased to do some of the more difficult polynomials given. This was not done, for obvious reasons, so no factorization times are available for Wang's algorithm. POLYFACT uses 53K (K=1024) words of memory on the UNIVAC 1108 for factoring most polynomials. This 53K words includes 28K words for the storage of approximately 230 subroutines and functions that comprise POLYFACT. The actual working space in POLYFACT is 25K words. Berlekamp's algorithm implemented by Wang has about 40K words (36 bits/word) available for working space.

One of the problems encountered by Wang's implementation is that large amounts of storage are required in factoring the more difficult polynomials (those with high degree and large numbers of terms). The part of POLYFACT that necessitates the largest use of memory is the M x N possibility list and the possibility search tree. The size of the M x N list is primarily a function

of the number of variables and the degree of the terms (more specifically the size of the M x N list is a function of the concentration of the variables in the terms and the degree of the variables). The size of the possibility search tree is dependent on the value of M and how well POLYFACT learns to select terms and possibilities.

Computing Costs

We have tried to develop a formulation of the computing costs for this factorization technique. However, the heuristic nature of the technique involving learning has prohibited us from formulating costs. All of the factors that dominate computing costs, e.g. M and the number of possibilities in the term T, are unknown until after factorization. The analysis of variance experiment is our best attempt to isolate the factors that tend to control the cost of factorization.

The encouraging result from the analysis of variance results is that the degree of the polynomial does not strongly influence performance as it does in Berlekamp-type algorithms. The analysis of variance tables (tables VI and VII) show that the factorization time is influenced somewhat by the number of possibilities in term T, and the number of possibilities in a term is influenced primarily by the number of variables in the term. Thus, our technique works particularly well, with respect to computing costs, for multivariable polynomials having at least one single variable term.

Usefulness of This Factorization Scheme

The author sees the usefulness of this factorization scheme in four areas. First, even though this technique is combinatorial in nature, it could in some

cases, obtain a solution more quickly than Hensel's approach. Polynomial 5 in Appendix A is an example of a situation where Hensel's Lemma would blow up a great deal. Wang's algorithm depends on substituting 0, 1, or -1 for all but one of the variables and then factoring the resulting univariate polynomial. Wang says that it is desirable to substitute as many zeros as possible because nonzero substitutions can cause some intermediate expression growth. Substitution of zeros for all but one variable can lead to the "bad zero" case mentioned earlier. Secondly, this technique could be combined with the Berlekamp-Hensel algorithm to handle some of the polynomials that cause problems for it (the author is currently working on a paper discussing this idea).

Thirdly, the idea presented through this technique deviates sharply from the current approaches to polynomial factorization by incorporating learning in it. Learning allows the algorithm to adjust to the sequence of polynomials factored with the result that factorization is performed more efficiently. We feel that new approaches to factorization should be presented, hopefully with the prospect that some useful ideas can be extracted. Finally, the scheme implemented in POLYFACT performs quite well despite the relatively inefficient implementation. We feel that the technique merits consideration; but we still consider the Berlekamp algorithms to be the most general factorization algorithms available.

Analysis of Variance Experiment

The performance of the mathematically oriented factoring algorithms, e.g. Musser's [19] and Berlekamp's [2] become less efficient as the degree of the polynomial increases. The effect of the number of variables and the size of the factors on the factorization times of these algorithms has not been published.

In order to determine what factors influence the factorization times for our scheme, we consider an experiment consisting of a factorial analysis of variance [1]. The analysis of variance separates the variations among all the observations into two parts, each part measuring variability attributable to some specific source. The level of significance is considered to be 10 per cent.

The factors considered in Table VI are:

1. M - the size of the M-term factor,
2. NVARS - the number of variables in the polynomial, and
3. NPOSS - the number of possibilities in the term T.

The factors in Table VII are:

1. M - the size of the M-term factor,
2. N - the size of the N-term factor,
3. NV - the number of variables in the term T, and
4. DEG - the degree of the term T.

Some of the important observations from the analysis of variance experiment are:

1. The factorization time is quite dependent on the value of M, since $F_{.90}(1,4)=4.54$.
2. The factorization time is somewhat dependent on the number of possibilities in the term T, since $F_{.90}(2,4)=4.32$.
3. The number of possibilities in T depends most heavily on the number of variables in T, followed by the degree of T, since $F_{.90}(1,2)=8.53$ and $F_{.90}(2,2)=9.0$.
4. The degree of the polynomial is not of major importance in determining the factorization time.
5. The effect of the number of variables in the polynomial on the factorization time is almost negligible.

TABLE VI

ANALYSIS OF VARIANCE
(FACTORIZATION TIME IS THE DEPENDENT VARIABLE)

Source	Degrees of Freedom	Sums of Squares	Mean Squares	F
M	1	142,702.62	142,702.62	6.53*
NVARS	2	2,553.06	1,276.06	0.05
NPOSS	2	158,432.96	79,216.48	3.62
M x NVARS	2	63.80	31.90	0.002
M x NPOSS	2	142,934.83	71,467.41	3.27
NVARS x NPOSS	4	83,081.77	20,770.44	0.95
RESIDUAL	4	87,423.08	21,855.77	
TOTAL	17	617,192.12		

*Significant at the 0.10 level in this and all following tables.

The reason that the number of variables in the polynomial and the degree of the polynomial are usually not heavy contributors to the factorization time is that this factorization scheme tries to select a term to initiate factorization that has a minimum number of variables in it and is of minimum degree. This tends to decrease the importance of these two factors.

We note from the analysis of variance experiment that the factorization time increases with the size of the M-term factor. This increase in factorization time is due partly to the fact that the larger the value of M, the more combinations of possibilities that can exist. However, this is not the only contributor to the value of M influencing the factorization time. The primary purpose of the initial implementation of this scheme in POLYFACT was to study learning in a complex problem environment and not necessarily to produce an efficient implementation. The importance of the size of M can be reduced considerably in POLYFACT by improving the technique for

TABLE VII

ANALYSIS OF VARIANCE
(NUMBER OF POSSIBILITIES IN T IS THE DEPENDENT VARIABLE)

Source	Degrees of Freedom	Sums of Squares	Mean Squares	F
M	1	425.04	425.04	2.15
N	1	0.38	0.38	0.002
NV	1	1,926.04	1,926.04	9.68*
DEG	2	3,077.08	1,538.54	7.74
M x N	1	477.04	477.04	2.40
M x NV	1	165.38	165.38	0.83
M x DEG	2	326.58	163.29	0.82
N x NV	1	715.04	715.04	3.60
N x DEG	2	704.25	352.13	1.77
NV x DEG	2	430.08	215.04	1.08
M x N x NV	1	3.37	3.37	0.017
M x N x DEG	2	165.08	82.54	0.42
M x NV x DEG	2	150.25	75.13	0.38
N x NV x DEG	2	847.58	423.79	2.13
RESIDUAL	2	397.75	198.88	
TOTAL	23	9,810.96		

searching the possibility search tree, and by providing a more efficient division algorithm. The division process in POLYFACT is a heuristic one and is not the usual polynomial division algorithms [7], [12].

Other Implementation Considerations and Possible Modifications

We feel that this factorization scheme can be made more efficient and perhaps more operable than the implementation in POLYFACT. We are currently investigating the possibility of incorporating some of the ideas in the Berlekamp-Hensel algorithm in this factorization scheme. Also the coefficient bounds formulation [27] can be used reduce the number of possibilities available in

the F_0 and F_1 sets. The replacement of the heuristic division algorithm by another polynomial division algorithm will improve the efficiency of operation considerably. By not requiring an explicit value for N, the number of combinations of M and N determined by the model-builder is greatly reduced, i.e. only the value of M will be specified.

The factorization scheme requires the value of N to form the M x N possibility list; however, in some instances (when terms have been combined) specifying N depends on predicting the number of terms in the polynomial prior to simplification. This places an unnecessary constraint on the model-builder. We propose the following as a possible alternative:

1. Perform factorization as described in this paper when the polynomial is considered unsimplifiable. This allows N to be explicitly defined, and the M x N possibility list to be constructed as outlined in Section IV.

2. When a polynomial is considered simplified then:

   a. Determine M and create the M x N possibility list by using $N=K\cdot\max(M,[NTERMS/M])$, where $1\leq K\leq I$ and I is chosen by the implementer and [ ] is the greatest integer function.

   b. Proceed with the factorization attempt as before; but use another more efficient division algorithm than the heuristic one mentioned above.

   c. If the attempt is unsuccessful, modify M and repeat the process.

We suggest that the initial value of K for a factorization attempt (with a fixed value of M) be greater than one. Then if factorization is unsuccessful, reduce K to one and try again. The reason for choosing an initial value of K greater than one is that the size of the possibility list decreases as N increases, and the possibilities for construction of the M-term factor might be in the smaller list. This could result in a saving of time and memory.

VI. SUMMARY

POLYFACT demonstrates that learning can be used to improve significantly the efficiency of a complex program in attempting to solve a difficult problem - the factorization of multivariable polynomials. The term selection tests show that POLYFACT selects a term with the minimum number of possibilities in approximately 73 percent of the polynomials factored. The possibility tests demonstrate that in most cases POLYFACT need consider only 20 to 50 percent of the possibilities in the actual search space.

Previous attempts at determining the symbolic factorization of multivariable polynomials using a purely heuristic approach have indicated very little success. The factorization tests described in this paper demonstrate that POLYFACT can factor many nontrivial polynomials. The degree of the polynomial and the number of variables in the polynomial influence the factorization times very little, as compared to the Berlekamp algorithms. The size of the M-term factor proves to influence the factorization times most. Surprisingly POLYFACT competes with Wang's implementation of Berlekamp's algorithm with respect to memory usage.

This heuristic factorization scheme appears to work quite well on polynomials that cause problems for Wang's algorithm, e.g. high degree polynomials, "bad zero" polynomials, etc. We feel that some of the ideas presented in this paper can be used to improve the performance of the more general Berlekamp algorithms, especially in the reconstruction of multivariable factors from the univariate factorization of the original polynomial. Also, some of the ideas in the Berlekamp algorithms can be used to definitely improve the performance of this technique implemented in POLYFACT.

ACKNOWLEDGEMENTS

The writer wishes to thank Jim Perry of the University of Connecticut for some critical observations of the initial draft of this paper; and also Richard Fateman and Paul Wang of Project MAC for their assistance in factoring some multivariable polynomials using Dr. Wang's implementation of Berlekamp's algorithm.

APPENDIX A

POLYNOMIALS

All of these polynomials were factored by POLYFACT as part of the research described in this paper.

1. $(8x-5y)(xy-8xz^2+z^2-5yz^2)$
2. $(5x^4+3x^5-6z^4)(7xz^3+4x^2y^2-9y^3z)$
3. $(yz+xz^3-x^3y^4)(-y^2z^2+xy+x^2z)$
4. $(x^3y^2)(1+xy^2z^4+x^2y^2z^4)(z^2+x^2y^5z^2+x^2z^5+z^5+x^2yz^3+xy^2z^4+x^2y^3z^5+xyz^4+x^3y)$
5. $(-y^{10}z^{14}+x^{17}y^4+x^9z^{18})(x^9y^3zw^4+x^6v^4z^8+w^7v^5)$
6. $(5x^2y^3-6x^5z^3+2xyz)(z^3w^4-xy^2z^2-x^3yw^2-x^5y^6w^4)(-z^6w^4-x^4y^2+y^2z^3-x^3yw^3-x^2y^2z^2w^2+x^5z)$
7. $(z^3w^4-xy^2z^2-x^3yw^2-x^5y^6w^4)(-x^6w^4-x^4y^2+y^2z^3-x^3yw^3-x^2y^2z^2w^2+x^5z)$
8. $(4y^2w^2+7x^2y^2zw^2+6x^3y+6x^3z^2w^2)(4w+4x^2z^2w+4y^3zw+3x^2y^2w^2+x^2yw^2+6y^2)$
9. $(x^2y^3-x^5z^3+yz)(z^3w^4-xy^2z^2-x^3yw^2-x^5y^6w^4)(-z^6w^4-x^4y^2+y^2z^3-x^3yw^3-x^2y^2z^2w^2+x^5z)$
10. $(z+y+x-3)^3$
11. $(z)(3xyzw-26y^3x^2-8xyzw^2+14x^2z)(-6x^3y^3z^3w^3+27x^2y^3z^3w^3-47xyzw-34x^2y^2w^2+5xy^2$
    $+47xw^3+6xyz^2w-39x^2y^2z^2w^3-21x^3y^3z^2w^2+3x^3yw^3+19x^3y^2zw^3-37y^2z^2w+26xyz^2+26x^3y^3z$
    $-47x^2y^2z^2w^2-47x^2yw^2)$
12. $(z+y+x-3)^3(z+y+x-2)^2$
13. $(x-y-z+w)(x-y+z-w)$
14. $(x^2y)(11y^2-22x^4+33x^5y^2+35x^3y)(22y^5+37x^4-18x^4y^5-38x^2y^4+29x^2y^3-41x^4y^2-26x^3y^4)$
15. $(70wu+x^2yw+62xyzu^2)(97xw+45xy^2zu^2-14z^2w^2-56y^2wu+96xy^2zwu-86xz^2w+2xy^2w^2u+81xz)$
16. $(29x^{12}y^{12}z^3w^4+3y^{20}w^{15}+21x^3z^2-15y^2z^{16})(x^{21}-y^{14}-z^{31}+w^2+y^{18}+x^2y^2-w^{12}z^{20})$
17. $(z-r-x^2y+w-xy^2)(-1-x+y-z^2w-z+zw^2+w)$
18. $(xy^2+21xz+zy^2+x^5z^8-9)^3$
19. $(x-y-z+w)(x^2-2xy+y^2-xz+yz+xw-yw+z^2-2zw+w^2)$
20. $(x^2z^2-y^3)(x-y+z)$
21. $(x^2y)(3y^5+5x^4-3x^4y^5-6x^2y^4)(y^2-3x^4+5x^5y^6+5x^3y)$

22. $(x^6y^4)(4y^2-2xy^2+3x^4)(-3+7y^2-x^2y)$

23. $(x)(2x-4y^3-5y^6-3x^3y^4)(-6-3xy-4x^4y^3+3xy^4)$

24. $(xy^2)(4x-xy^4-y^3+xy^3)(6+4x^4y^6+y-2x^6)$

25. $(x^2y^2)(5y+3x^5+x^5y)(y^4+4x^4y^4+x^4)$

26. $(x)(4x^2-y^5+7x^2y^3+7xy^2)(3x+7xy^4-6y^5-6x^4y^5)$

27. $(6xyz)(6x^5-x^4y^6-5y^3z^5)(x^3-x^4+y^4z^3)$

28. $(yx^3)(2z^2+5xy^5w^4+6x^2y^5z^2w^5)(-7-2xy^6zw^6+y^5zw^6+7x^2y^6z^5w^3-3x^4y^3z^5w^2)$

29. $(x^4y)(6x^2+7xy-3y^4-4xy^2)(-3x^2-x^3+4x^4y-2y^5-x^6y^4)$

30. $(xy)(2x+5y^4-2x^6y)(6x^4-6xy^3-7y^4+3x^3y^4-x^4y^3)$

31. $(y)(5y^3+3y^4-2x^4)(-3y^2+7x^6y^6-4x^3y^5-7x^6)$

32. $(2y^2z^3)(2z^2-3x^5y^2-3x^5y^4z^3)(3z^6+3xy^6-5x^2y^5z^2)$

33. $(2x^4y^5w^4)(4z^3+3y^3z^6w+y^2z-6x^2y^3w)(2xz+2x^2w^2+2x^4z^2w^2+z^4w)$

34. $(2x^4yz^2)(3x^2+3y^3z^5+x^2y^3z^3-2z^2)(-7xy-7x^3z^2+4y^4+3x^2y^3z^3)$

35. $(x^6y^2z)(7x-2x^2yz^3+2y^3z^3)(-6xz^3-7y^6z-6xy^2z^2+x^2y^5z^3-6x^3y^3)$

36. $(x^2y^2zw)(7z^5w^5-3x^3y^5w^2+3xy^6z^2w^2+7x^5y^6z)(2-2x^4y^3z^2-x^5y^3w^2+5xy^3z^3)$

37. $(x^4y^2z^4)(3x^2+3yz-4x^5y^2z^2-5x^2z^3)(-3x^2-4y^6+6x^2z^5-5x^3y^4z^4-5x^3y^6z^3)$

38. $(x^5y^3z^2)(4y+7xy^3z^5+x^2-5x^3y^4z^4)(-3y^3-3x^3y-z^4-7x^3z^5)$

39. $(x^2yz)(x^4-2y^4z^2+2x^4z)(3x+3yz^5+6x^2y^4z^5+x^4y^5z^4)$

40. $(x^5y^3z)(6z^4-7y^4z^2-2y^4)(7x^2z^2+3x^4y^3+7y^2z^3-2xy^2z^4-7x^3z)$

41. $(xz)(z^3-6x^3yz^4-3xy^4)(-7x^4-5x^2y^2z^3-4x^4z^2+7y^3z^4+4x^5z^5)$

42. $(y)(2x+x^6y^3z^6-y)(-3z^5-y^4z^6-x^3y^4-7xyz)$

43. $(zx+x^6y^3z^6-y-4x^2y^2)(6y^4-6x^2y^5+3x^3y+4y^6-6x6y^4)$

44. $(xy^2z)(2x^6+5xy^2z^3-4x^3yz^4+7y^4z^4)(3xz^5-5x^5y^4z^6+y^4z^5-x^5y^3-4x^3z^6)$

45. $(11y-22x^2+33x^2y^2+35x)(-41+10x^3y+22xy^3-18x^3y^3+67x^2y^2-41x^3y^2-26x^2y^3+5y^2+3x^2)$

46. $(xy)(47xy+4xy^2u+z^3w^3-13xyz^3w^2u^3)(8xy^2+43x^3y^3z^3u-23x^3y^2wu-20x^2y^2z^2w^2u^2$
$-46xwu^2+45x^3-48x^3y^2zwu-9y^3wu+36xy^3zwu^2+5x^2y^3wu+36xyz^3w^3u^2+36y^3w^2u$
$-29xy^3z^2w^3+14x^3y^3z^2w^3+46x^32^3u+31y^2w^3u^2-9y^2u^2)$

47. $(x^2yzwu^2)(y^3-32x^2zw^3u+29yz^2w^3u-44xw^2)(-26x^2u^2+48yz^2wu-x^2y^2zw^2u^2)$

48. $(y^2)(21x-32x^3z+7z)(19x+28x^2y^3+9x^2y-2x^2y^2+23x^3yz^3+32z^2+19xz^3-10x^2y^3z$
$+19x^2y^2z^2-8xz+22x^2z^2-13x^3z^3+28x^2z^3)$

49. $(z)(14x^2+3xyw-26y^3z-8xyw^2)(5xy^2-6x^3y^3z^3w^3+27x^2y^3z^3w^3-47xyzw-34x^2y^2w^2$
$+47xw^3+6xyz^2w-39x^2y^2z^2w^3-21x^3y^3z^2w^2+3x^3yw^3+19x^3y^2zw^3-37y^2z^2w+26x^3y^3z$
$-47x^2y^2z^3w^2-47x^2yw^2)$

50. $(xy)(45x-28x^2-46y^2)(-14+5xy^2-34y^2-15x^2)$

51. $(zw^2u)(42z^2+46x^3y^3u-7x^2y^3zw^2u^2)(15x^2z^3w-4y^3zw^2-37x^3y^3z^3w-6x^3y^3u$
$+42xzwu^2-2x^3ywu^3-6xz^2u^3+19x^3z^3w^2)$

52. $(x^3yz)(31x^2-22y^2zw-23xyz^2w^2+40xyw^3)(-24w^2+25xz^3w^2-42xy^2w^3+28z^3w^3-29xz^2)$

53. $(y^2)(14zw+63x^3z^2w^2-41x^3yw+46x^2yz)(33zw+49x^2y+9x^3z^2w^3-26x^2y^2z^2w)$

54. $(yz)(14y^2-38zw^3-41xz-46x^3z^2w^2)(-29y+9x^2z^2w-28x^2y^2z+43xz^3w^3+52x^3zw^3$
$+4xyz^2w^3+7x^2y^3z^2w-8xz^2-33x^2y^2zw^3+21y^2zw^2+15xyz^2-37xy^3z^3-21x^3yz^3$
$-40y^3+35xy^3+8y^2z^3w^2+20x^2zw^2)$

55. $(x)(6y^2w-14xy^3zw^2-34xz^3+27y^3z^2w^2)(-43y^2+20x^2y^2z+26z^2w^3+44x^2w^2+18xw^2$
$+17x^2y^3z^2w^3)$

56. $(yz)(44x^2z^2-41x^3z^2-26yz^3+24y^2w^2)(-12x^2z-38xzw+34x^3yzw^2-y^2w^3-6xy^2w^3)$

57. $(xy^2z^2)(19xy^2w^2+37x^3yz^3u^3-x^3z^3w^2u^2-16y^2zw^3)(33u^3-27x^2zw^3-16x^2yzw^3$
$+13yw^2u^3+16x^5y^2w^2u^2)$

58. $(x^2z^2u)(8x-43y^2zw-43yzw^2u)(-8x^2w^2u^2-16x^2yz^3u+49x^3yz^3wu^2+12z^3w^3$
$-5x^2yzw^2u^2-40xyzw^2u)$

59. $(29z^2+27y^2z^3-23x^3y^2-49x)(-22xy+28x^2yz-38x^2y^3z^3-43x^3y-18x^3z^2-78xy^3z^2$
$+35y^2z^3+44yz^3+y^3z^2+20xy^2z-42x^3yz^3-31x^2y^2+2y^3+25y^2z-5yz-46x^2y^2z)$

60. $(y^2z)(39yw-49xz+25x^3zw^3)(37yzw-48x^3zw^2+49x^2y^3w-9z^3w^3-10x^2y^2z+43xy^3z^2w^3$
$+41x^2zw^3-33x^2y^2zw)$

61. $(xz)(31yz^2+20xyw-44x^2y^2z^2w^3-39xz^2w^3)(-44yw-9xz^3w^3+30z^3w-37x^3y^2w$
$-3x^2yz^2w^3+21y^2z^3w-20xyz^3w^2-7yz^2w^2-24x^3y^3)$

62. $(yz)(11xw+39x^3y^2zw-3x^2z+yz^2w^3)(-19w^2-42xy^2w^2-38xyw-47yzw^2+2y^3zw$
$-46x^2y^2zw-27x^3y^3w+46y^2z^2w^3-14x^2y^3z)$

63. $(r^2v^3-rv^3-t^5)(z^3w^4-xy^2z^2-x^3yw^2-x^5y^6w^4)(-z^6w^4-x^4y^2+y^2z^3-x^3yw^3$
$-x^2y^2z^2w^2+x^5z)$

64. $(24yz+68xz^3-72x^3y^4)(-x^{10}z^{14}+x^{17}y^4+x^9z^{18})$

65. $(24yz+68xz^3-72x^3y^4)(-y^2z^2+xy+x^2z)$

66. $(x)(47y+4y^2u+z^2w^2-13xyz^2wu^2)(45x^2+43x^2y^2z^2u-23x^2yw+16xy^2zwu-46xu^2$
$-48x^2ywu+27y^2wu+5x^2y^2w+36xyz^2w^2u-29y^2zw^2+14x^2y^2zw^2+46x^2y^2u+8xy$
$+31yw^2u^2-9y^2u^2)$

67. $(xz^2u^4)(15z^2u+10xy^3wu^2+18xz^2w^3u^2+6y^3z^2w^2)(-12y^3z^2+48x^2y^3z^3u^2$
$+2x^2y^2wu^2-25yz^4w^3u^2-32y^4z^3w^4u^2-4xw^2+8x^3z^4wu-11x^3yw^2u-44xy^4z^4wu)$

68. $(6xy+40x^2w+31xzu^2+35y^2w^2)(24xy+9x^2wu^2+44z^2w^2u+37xzw^2+xy^2z^2w^2u^2$
$+29y^2w^2+31yz^2w^2+24x^2yz^2u^2+37y^2zu^2+23xyz+13x^2y^2w^2u+21xyzwu^2+12x^2yzw$
$+8z^2w^2u^2+22xy^2+22w^2u^2+12xy^2z^2u^2+43xyz^2w+43x^2yu+39xzwu+47zw^2u^2$
$+24xy^2z^2wu^2+27x^2uw^2u+41y^2z+42y^2w^2u^2)$

69. $(xy)(47xy+4xy^2u+z^3w^3-13xyz^3w^2u^3)(45x^3+43x^3y^3z^3u-23x^3y^2wu-20x^2y^2z^2w^2u^2$
$-46xwu^2-48x^3y^2zwu-9y^3wu+36xy^3zwu^2+5x^2y^3wu+36xyz^3w^3u^2+36y^3w^2u-29xy^3z^2w^3$
$+14x^3y^3z^2w^3+46x^3y^2u+8xy^2+31y^2w^3u^2-9y^2u^2)$

70. $(z+y+x-3)^3$

71. $(47xy+z^3w^2-w^2)(45x^3+3z^3-y^2-9y^2+2wz)$

72. $(35x^3y+33x^5y^6+11y^2-22x^4)(22y^5+37x^4-18x^4y^5-38x^2y^4+29x^2y^3-41x^4y^2-26x^3y^4)$

73. $(x^6y^3z^2)(29x+3xy^2+z^3w^2-12xyz^3w^2-w^2)(18x^3y+3z^3-y^2+14y^2w^2-8xy^2+2wz)$

REFERENCES

1. Bennett, Carl A. and Franklin, Norman A. Statistical Analysis in Chemistry and the Chemical Industry, Wiley, 1954, pp. 319-469.

2. Berlekamp, E. R. "Factoring Polynomials Over Large Finite Fields", Mathematics of Computation, Vol. 24, #111, July, 1970, pp. 713-735.

3. Berlekamp, E. R. "Factoring Polynomials Over Finite Fields", Bell System Technological Journal, Vol. 46, 1967, pp. 1853-1859.

4. Brown, W. S. "On Euclid's Algorithm and the Computation of Polynomial Greatest Common Divisors", JACM, Vol. 18, No. 4, October, 1971.

5. Claybrook, B. G. "POLYFACT: A Learning Program that Factors Multivariable Polynomials", Dissertation, Computer Science/Operations Research Center, Southern Methodist University, 1972, 194 pp.

6. Claybrook, B. G. and Nance, R. E. "The Dynamic Creation and Modification of Heuristics in a Learning Program", In Preparation.

7. Collins, G. E. "PM, A System for Polynomial Manipulation", CACM, Vol. 9, August, 1966, pp. 578-589.

8. Collins, G. E. "The SAC-1 System: An Introduction and Survey", Proceedings of the Second Symposium on Symbolic and Algebraic Manipulation, 1971, pp. 144-152.

9. Hearn, Anthony C. "REDUCE 2: A System and Language for Algebraic Manipulation", Proceedings of the Second Symposium on Algebraic and Symbolic Manipulation, 1971, pp. 128-133.

10. Hunt, Earl B., Marin, Janet, and Stone, Philip J. Experiments in Induction, Academic Press, New York, 1966, 247 pp.

11. Johnson, S. C. "A Factoring Algorithm for Polynomials Over an Arbitrary Galois Extension of the Rationals", Bell Laboratories Report, 1966, 38 pp.

12. Jordan, D. E., Kain, R. Y., and Clapp, L. C. "Symbolic Factoring of Polynomials in Several Variables", CACM, Vol. 9, August, 1966, pp. 555-569.

13. Knuth, D. E. The Art of Computer Programming, Vol. 2, Seminumerical Algorithms, Addison-Wesley, Reading, Massachusetts, 1969.

14. Manove, M., Bloom, S., and Engleman, C. "Rational Functions in MATHLAB", Symbol Manipulation Languages and Techniques, Daniel Bobrow (ed.), North-Holland, Amsterdam, 1968, pp. 86-102.

15. Martin, W. A. and Fateman, Richard J. "The MACSYMA System", Proceedings of the Second Symposium on Symbolic and Algebraic Manipulation, 1971, pp. 59-75.

16. Mendelson, Elliott. Introduction to Mathematical Logic, Van Nostrand Reinhold, New York, 1964, 300 pp.

17. Michie, Donald and Ross, Robert. "Experiements with the Adaptive Graph Traverser", Machine Intelligence 5, Meltzer, Bernard and Michie, Donald (eds.), American Elsevier, 1970, pp. 301-320.

18. Minsky, M. L. "Steps Toward Artificial Intelligence", Proceedings of the IRE 49, 1961, pp. 8-30.

19. Musser, David R. "Algorithms for Polynomial Factorization", Ph.D. Thesis, Computer Sciences Department, University of Wisconsin, 1971, 174 pp.

20. Newell, Allen, Shaw, J. C. and Simon, H. A. "A Variety of Intelligent Learning in a General Problem Solver", Self-Organizing Systems, Yovits, Marshall and Cameron, Scott (eds.), Pergamon Press, 1960, pp. 153-189.

21. Samuel, A. L. "Some Studies in Machine Learning Using the Game of Checkers", In Computers and Thought, Feigenbaum, E. and Feldman, J. (eds.), McGraw-Hill, 1963, pp. 71-105.

22. Slagle, J. R. and Farrell, C. D. "Experiments in Automatic Learning for a Multipurpose Heuristic Program", CACM, Vol. 14, February, 1971, pp. 91-99.

23. Van der Waerden, B. L. Modern Algebra, Vol. 1, Frederick Ungar Publishing Company, New York, 1953.

24. Wang, Paul S. and Rothschild, L. Preiss. "Factoring Multivariate Polynomials Over the Integers", SIGSAM Bulletin, No. 28, December 1973, pp. 21-29.

25. Waterman, D. A. "Generalization Learning Techniques for Automating the Learning of Heuristics", Artificial Intelligence 1, 1970, pp. 121-170.

26. Zassenhaus, H. "On Hensel Factorization I", Journal of Number Theory, Vol. 1, 1969, pp. 291-311.

27. Zimmer, Horst G. "Computers and Computations in Algebraic Number Theory", Proceedings of the Second Symposium on Symbolic and Algebraic Manipulation, March, 1971, pp. 172-179.