An Algorithm for the Chromatic Number of a Graph

CHUNG C. WANG

State University of New York at Buffalo, Amherst, New York

ABSTRACT. Christofides' algorithm for finding the chromatic number of a graph is improved both in speed and memory space by using a depth-first search rule to search for a shortest path in a reduced subgraph tree.

KEY WORDS AND PHRASES: chromatic number, color graph, scheduling, depth-first search, subgraph tree, independent set

CR CATEGORIES: 5.32

1. *Introduction*

Christofides [1] describes an algorithm for coloring the vertices of a graph using the minimum number of colors so that no two adjacent vertices are colored the same. His algorithm implies the construction of the maximum subgraph tree (see Section 3) of a graph from its maximal indepdendent sets (Harary [2]) and finding, by a breadth-first search rule, a shortest path between the root and the terminal nodes of this subgraph tree. The number of branches from each subgraph node in this subgraph tree is equal to the total number of maximal independent sets of the subgraph. This number is usually too big to make the algorithm feasible for its implementation on a computer, even for a graph with a small number of vertices.

This paper attacks the same problem. However, we show that the number of branches from each subgraph node in the subgraph tree can be reduced by considering only a subset of all maximal independent sets of the subgraph. Let p be the number of vertices in a graph. For a certain class of graphs, the number of terminal nodes is reduced by a factor of $(p/2)!$, which, we shall see, implies that the required number of steps to compute the chromatic number of a graph from its maximal independent sets is reduced by a factor of as much as $(p/2)!$ over Christofides' algorithm. The amount of storage is similarly reduced by a factor of $(p/2)!$ over Christofides' algorithm. By adopting a depth-first search rule to find a shortest path in a reduced subgraph tree, the required amount of storage can further be reduced by a ratio of $\sim p^2/8$ to $(p/2)!$.

At the end of this paper, we describe a depth-first search algorithm which is essentially the same as an experimental program we have written. Some results are reported.

We begin by proving a theorem which is used to justify our use of a reduced subgraph tree in our algorithm.

2. *Definitions and Underlying Theorem*

A set of vertices of a graph, G, is independent if no two of them are adjacent. An independent set of vertices, M, is maximal if no independent sets properly contain M in G. A

This work was supported in part by NSF grant No. 1451-A.

Author's present address: Department of Computer Science, University of Kentucky, 915 Office Tower, Lexington, KY 40506.

coloring of G is an assignment of colors to the vertices of G such that no two adjacent vertices have the same color. The set of vertices with any one color is independent and called a color class. An n-coloring of G uses n colors and therefore partitions vertices of G into n color classes. The minimum number of colors, n, required to color G is defined as the chromatic number of G. A color class in any coloring of G may or may not be a maximal independent set of G. The following lemma assures us that for any vertex of G, there exists a chromatic coloring of G such that the color class containing it is a maximal independent set of G.

LEMMA 1. *For every vertex x of G, let $C_1, C_2, \cdots, C_k$ be the maximal independent sets of G containing x; then there exists a chromatic coloring of G such that one of its color classes is C_i for some i, $1 \leq i \leq k$.*

PROOF. Let the chromatic number of G be n and $\{P_1, P_2, \cdots, P_n\}$ be a chromatic coloring of G. Without loss of generality, let $x \in P_1$. If $P_1 = C_i$, for some i, $1 \leq i \leq k$, then we are done. Otherwise, there exists an i, $1 \leq i \leq k$, such that $P_1 \subset C_i$. Let $P_2', P_3', \cdots, P_n'$ be the sets obtained by removing vertices of $C_i - P_1$ from the sets $P_2, P_3, \cdots, P_n$ respectively. Then the partition $\{C_i, P_2', P_3', \cdots, P_n'\}$ is the desired chromatic coloring of G. Q.E.D.

For any set S of vertices of G, we use $\langle S \rangle$ to denote the induced subgraph, i.e. the maximal subgraph of G with the vertex set S. Let V be the set of all vertices of G. The following lemma deals with the chromatic number of the induced subgraph, $\langle V - S \rangle$, when S is an independent set.

LEMMA 2. *If S is an independent set of G, the chromatic number of the induced subgraph, $\langle V - S \rangle$, is one less than the chromatic number of G if and only if S is a color class of some chromatic coloring of G.*

PROOF. Immediate from definitions.

From Lemmas 1 and 2, we have the following useful theorem, which, in fact, makes possible the improvements over the Christofides algorithm.

THEOREM. *For every vertex, x, of G, let $C_1, C_2, \cdots, C_k$ be the maximal independent sets containing x; then the chromatic number of $G = 1 + min_{1 \leq i \leq k}$ {chromatic number of $\langle V - C_i \rangle$}.*

PROOF. Let the chromatic number of G be n. The chromatic number of $\langle V - C_i \rangle$ is greater or equal to $n - 1$. Lemma 1 shows there exists one i, $1 \leq i \leq k$, such that C_i is a color class of a chromatic coloring of G. Lemma 2 shows that the chromatic number of $\langle V - C_i \rangle$ is $n - 1$, which proves the theorem.

We will use the formula implicit in the above theorem to compute recursively the chromatic number of G. We now introduce the concept of a subgraph tree used in the explanation and comparison of the algorithms.

3. *A Reduced Subgraph Tree*

The process of finding the chromatic number of G can be pictured as a search for a shortest path in a subgraph tree of G. The subgraph tree which we are about to construct is not unique. Even though we require that the number of branches from each subgraph node in our subgraph tree to be minimal, we have no guarantee that the subgraph tree has a minimum number of nodes. However, we shall see that the number of nodes in our subgraph tree, which we will call a reduced subgraph tree, is much less than that in the maximum subgraph tree, implied in the Christofides algorithm.

We construct a reduced subgraph tree of G as follows, assuming as Christofides does that the maximal independent sets of G have been found. The root of the subgraph tree is G itself and we say that the subgraph node, G, is in level 0. We look for a vertex which is contained in the least number of maximal independent sets of G. If more than one such vertices exist, we arbitrarily choose one of them as x. Let the maximal independent sets containing x be $C_1, C_2, \cdots, C_k$. The set of induced subgraphs of G, $\langle V - C_1 \rangle, \langle V - C_2 \rangle, \cdots, \langle V - C_k \rangle$, satisfies the condition of our theorem for computing the chromatic

number of G. This set of induced subgraphs forms the first level subgraph nodes of a reduced subgraph tree. For each first level subgraph node, say $G' \equiv \langle V - C_i \rangle$, where $1 \leq i \leq k$, we choose an arbitrary vertex y of G' such that y is contained in the least number of maximal independent sets of G'. Let the maximal independent sets of G' containing y be $D_1, D_2, \cdots, D_j$; then the set of induced subgraphs of G', $\langle V - (C_i \cup D_1) \rangle$, $\langle V - (C_i \cup D_2) \rangle$, $\cdots$, $\langle V - (C_i \cup D_j) \rangle$, again satisfies our theorem for computing the chromatic number of G'. This set of induced subgraphs forms the second level subgraph nodes branching from G' in a reduced subgraph tree. We continue this way to construct higher level nodes from lower level nodes until we reach the terminal nodes of a subgraph tree, which are null graphs.

We have called the subgraph tree of G implied in the Christofides algorithm the maximum subgraph tree for the following reason. Each maximal independent set of G creates a distinct subgraph node in the first level of the subgraph tree. If G' is a subgraph node in this subgraph tree, each maximal independent set of G' results in a distinct subgraph node from G'. This subgraph tree is obviously the maximum one which can be constructed from the maximal independent sets of a graph. The maximum subgraph tree is unique. On the other hand, the set of all first level subgraph nodes in a reduced subgraph tree is only a subset of the first level nodes in the maximum subgraph tree. A similar situation exists at every level for each of the subgraph nodes. Let G' be a subgraph node in the maximum subgraph tree. G' may or may not be a subgraph node in a reduced subgraph tree. If G' is, then the set of all nodes branching from G' in a reduced subgraph tree is again a subset of all nodes branching from G' in the maximum subgraph tree. Hence a reduced subgraph tree is a subtree of the maximum subgraph tree.

A path from the root to a terminal node of a subgraph tree, either the maximum one or a reduced one, may be interpreted as a coloring of G; let the path include subgraph nodes $G = G_0, G_1, \cdots, G_k$ where each G_i, for $1 \leq i \leq k$, is created by one maximal independent set of G_{i-1} and G_k is the null graph. The length of the path, k, is the number of colors used in the coloring. The color classes are those sets of vertices in $G - G_1$, $G_1 - G_2, \cdots, G_{k-1} - G_k$. It follows from the way we construct the subgraph trees that a shortest path must correspond to a chromatic coloring of G. We have to compare the length of all paths from the root to the terminal nodes of a subgraph tree to find a shortest path. The number of steps required to compute the chromatic number of G is therefore proportional to the total number of paths. Since the maximum subgraph tree contains a reduced subgraph tree, the number of steps required by the Christofides algorithm is always greater than that required by our algorithm, except in trivial cases. The question of how much greater is dealt with in Section 4.

4. *Quantitative Analysis*

In this section we give a quantitative comparison of the sizes of the maximum subgraph tree and a reduced subgraph tree for two special classes of graphs. First, consider a graph which consists of m disjoint copies of the complete graph with p/m vertices. Let G be such a graph. G obviously has p vertices. We find a typical maximal independent set of G by choosing one vertex from each of the m complete subgraphs of G. Hence there are $(p/m)^m$ maximal independent sets of G. Let G' be an ith-level subgraph node in a subgraph tree of G. G' must be in the form of m disjoint copies of the complete graph with $(p/m - i)$ vertices. The number of maximal independent sets of G' is, similarly, $(p/m - i)^m$. The number of subgraph nodes branching from G' in the maximum subgraph tree is therefore $(p/m - i)^m$. On the other hand, a fixed vertex of G' is contained in exactly $(p/m - i)^{m-1}$ maximal independent sets of G', which implies that there are merely $(p/m - i)^{m-1}$ subgraph nodes branching from G' in a reduced subgraph tree. All paths from the root to the terminal nodes in a subgraph tree are at equal length. The number of shortest paths is equal to the number of terminal nodes in a subgraph tree of G. There are $((p/m)!)^m$ terminal nodes in the maximum subgraph tree as opposed to

$((p/m)!)^{m-1}$ terminal nodes in a reduced subgraph tree. The number of steps required by our algorithm is therefore $(p/m)!$ times less than that required by the Christofides algorithm. This improvement factor is $(p/2)!$ when $m = 2$.

For graphs which are connected and can be converted to the graph G above by deleting or adding few edges, we expect roughly the same amount of improvement using a reduced subgraph tree over the maximum subgraph tree.

Next, let us compare the number of first level nodes in the maximum subgraph tree to those in a reduced subgraph tree of a random graph. Random graph is defined by Erdös and Rényi [3–6]. By a random graph G_{pq} we shall mean a graph on p vertices where each of the $p(p-1)/2$ edges occurs with probability q. Matula [7] shows that the expected number of cliques of G_{pq} is $\sum_{d=1}^{p} \binom{p}{d} q^{d(d-1)/2}(1-q^d)^{p-d}$. The expected number of first level nodes in the maximum subgraph tree of G_{pq} is equal to the expected number of the maximal independent sets of G_{pq}, which can be obtained by substituting $1-q$ for q in the above expression. This expected number is $\sum_{d=1}^{p} \binom{p}{d}(1-q)^{d(d-1)/2}(1-(1-q)^d)^{p-d} = E_2$. The expected number of first level nodes in a reduced subgraph tree is in general less than the expected number of maximal independent sets containing a fixed point of G_{pq}, which can be shown to be $\sum_{d=1}^{p} \binom{p-1}{d-1}(1-q)^{d(d-1)/2}(1-(1-q)^d)^{p-d} = E_1$. The expected numbers of the first level nodes in the maximum subgraph tree and a reduced subgraph tree of G_{pq} for $p = 10, 20, 30, 40, 50$ and $q = .1, .2, \cdots, .9$ are compared using expressions E_1 and E_2 in Table I.

5. *A Depth-First Search Algorithm*

A shortest path in a reduced subgraph tree can be found by using one of the two search methods, i.e. breadth-first search and depth-first search. In the breadth-first search, the algorithm scans through all possible paths in a given level for a terminal node; if a terminal node is encountered, then a shortest path is found and the algorithm stops; otherwise the algorithm continues to scan all nodes in the next level for a terminal node. On the other hand, in a depth-first search algorithm, we use the following rule to select the next node for scanning: select a node which is not yet scanned before and is branched from a node most recently being scanned. The first path from the root to a terminal node scanned by the algorithm serves as an initial guide line when other paths are scanned. Whenever a shorter path is encountered, this shorter path will be used as a guide line when the remaining paths are scanned. The algorithm scans down a path only up to the level which is equal to the length of the path used as a guide line. The path, which is last used as a guide line when all paths have been scanned, is one of the shortest paths of a subgraph tree desired.

TABLE I

p =	10		20		30		40		50	
	E_1	E_2	E_1	E_2	E_1	E_2	E_1	E_2	E_1	E_2
q =										
.1	3.88	6.07	39.36	78.11	377.51	890.04	3096.18	8374.68	21701.58	65856.81
.2	4.66	9.26	35.96	98.49	209.58	715.24	970.85	3924.56	3776.08	17527.55
.3	4.40	10.57	24.19	84.28	96.68	431.16	311.48	1673.29	861.85	5379.24
.4	3.83	10.84	15.70	66.91	47.58	264.27	119.51	808.52	264.16	2092.52
.5	3.22	10.61	10.30	52.60	25.21	170.00	52.63	435.90	98.78	963.50
.6	2.66	10.16	6.87	41.54	14.11	114.22	25.41	254.09	42.00	496.74
.7	2.17	9.57	4.59	32.98	8.15	78.63	13.02	156.73	19.32	277.09
.8	1.71	8.76	3.09	26.77	4.69	55.63	6.62	97.38	8.95	155.93
.9	1.25	8.15	1.89	19.68	2.63	38.22	3.40	63.51	4.17	94.83

Notes. E_1 = the expected number of maximal independent sets containing a fixed vertex of G_{pq}; E_2 = the expected number of maximal independent sets of G_{pq}.

Since almost all nodes at the levels which are smaller than the length of the shortest paths have to be scanned in both search methods, the number of nodes scanned in the depth-first search method is in general greater than that in the breadth-first search method. However, the amount of storage required by the depth-first search method is much less than that required by the breadth-first search method. When the ith-level nodes are scanned, all possible nodes at levels from 0 to $i - 1$ must be kept in storage with the breadth-first search method, whereas with the depth-first search method, the nodes at the jth level, for $1 \leq j \leq i$, required to be kept in storage are merely all those branched from a common node at $(j - 1)$-st level. For example, consider again the graph which consists of m disjoint copies of the complete graph with p/m vertices. The maximum number of nodes in a reduced subgraph tree which are required to be kept in storage is approximately $\sum_{j=0}^{p/m-1} (p/m - j)^{m-1}$ for the depth-first search method and $\prod_{j=0}^{p/m-1} (p/m - j)^{m-1}$ for the breadth-first search method. This ratio is $\sim p^2/8$ to $(p/2)!$ when $m = 2$. We expect from this example that the saving in memory space in using the depth-first search over the breadth-first search is substantial for finding the chromatic number of a general graph.

We have programmed a depth-first search algorithm in PASCAL [8] and run it on a CDC 6400. Since the running time depends highly on the structure of a graph, an estimate of the running time is difficult to derive. Table II shows the frequency with respect to running time for 50 pseudorandom graphs of 20 vertices at edge densities 0.25, 0.50, and 0.75. The running times recorded in Table II include the time taken by Bierstone's algorithm [9] to generate the maximal independent sets for each graph. We describe the depth-first search algorithm.

It is assumed here that the maximal independent sets of a given graph G are computed before the algorithm begins. For example, one may use Bierstone's algorithm for this purpose. The following notation is used in the algorithm:

V	the set of vertices of graph G,
M_j	the maximal independent set (MIS) of G, where $1 \leq j \leq m$ and m is the total number of MIS's of G,
g	the would-be chromatic number of G,
c_j	the would-be jth color class of a chromatic coloring of G,
n	the current search level of the subgraph tree of G,
T	the set of vertices of G not in the current subgraph at level n,
b_n	the MIS of $\langle V - T \rangle$ at level n in processing,
e_n	the number of unprocessed MIS's of $\langle V - T \rangle$ at level n,
STACK	store all unprocessed MIS's of $\langle V - T \rangle$ at level 1 up to level n.

Step 1. Set $T = \varnothing$, $n = 0$, $b_0 = \varnothing$, and $c_i = \{i\}$ for $i = 1, 2, \cdots, g$. ($g = \mid V(G) \mid$.)

Step 2. If $n \geq g$ then set $T = T - b_n$, go to step 6; otherwise continue to step 3.

Step 3. If $V - T = \varnothing$ then set $g = n$, $T = T - b_n$, $c_i = b_i$ for $i = 1, 2, \cdots, g$ and go to step 6; otherwise continue to step 4.

TABLE II

	$G_{20,.25}$	$G_{20,.50}$	$G_{20,.75}$
0–5 sec	3*	18	47
5–10 sec	9	17	3
10–15 sec	9	11	0
15–20 sec	14	2	0
20–25 sec	7	1	0
25–30 sec	3	1	0
>30 sec	5	0	0

* The table should be read that 3 $G_{20,.25}$ graphs out of 50 samples run to completion during the period from 0 to 5 sec.

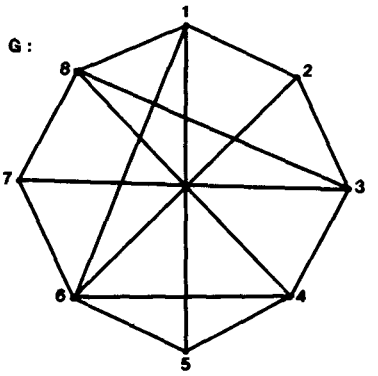

FIG. 1. The vertex set $V = \{1,2,3,4,5,6,7,8\}$. The maximal independent sets of G are $\{2,5,8\}$, $\{1,4,7\}$, $\{2,5,7\}$, $\{6,8\}$, $\{2,4,7\}$, $\{3,5\}$, $\{3,6\}$, and $\{1,3\}$. The chromatic coloring found by the algorithm is $\{\{1,4,7\}, \{2,5,8\}, \{3,6\}\}$.

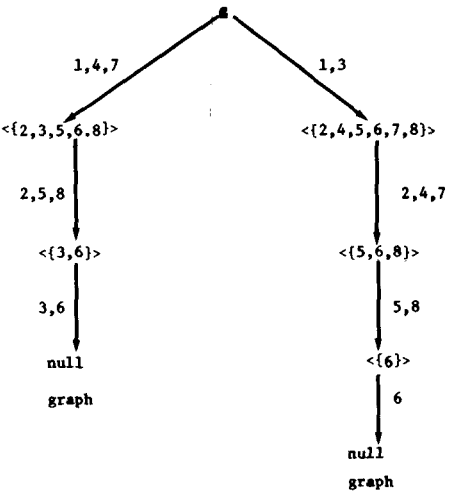

FIG. 2. A reduced subgraph tree of the graph G in Figure 1

Step 4. Compute the maximal independent sets of $\langle V - T\rangle$ from $M_1 - T, M_2 - T, \cdots, M_m - T$ and let them be denoted as $S_1, S_2, \cdots, S_t$.

Step 5. Choose $u \in V - T$ such that u is contained in the least number of maximal independent sets of $\langle V - T\rangle$. Let them be denoted as $S_{u_1}, S_{u_2}, \cdots, S_{u_r}$. Put S_{u_i} on STACK for $i = 1, 2, \cdots, r$. Set $n = n + 1$, $e_n = r$.

Step 6. If $n = 0$ then stop; otherwise continue to step 7.

Step 7. If $e_n = 0$ then set $n = n - 1$, $T = T - b_n$ and go to step 6; otherwise continue to step 8.

Step 8. Move the top item from STACK to b_n. Set $e_n = e_n - 1$, $T = T \cup b_n$, and go to step 2.

6. *Example*

Consider an 8-vertex graph[1] G shown in Figure 1 and the subgraph tree of G shown in Figure 2. A chromatic coloring of G is $(\{1,4,7\}, \{2,5,8\}, \{3,6\})$ as indicated in the following trace of the algorithm.

Step 1. $T = \varnothing$, $n = 0$, $b_0 = \varnothing$, $g = 8$.

Steps 2–4. MIS's of $\langle V\rangle$ are $\{2,5,8\}$, $\{1,4,7\}$, $\{2,5,7\}$, $\{6,8\}$, $\{2,4,7\}$, $\{3,5\}$, $\{3,6\}$, $\{1,3\}$.

Step 5. $u = 1$, $n = 1$, $e_1 = 2$, STACK = $(\{1,3\}, \{1,4,7\})$.

Steps 6–8. $e_1 = 1$, $b_1 = \{1,4,7\}$, $T = \{1,4,7\}$, STACK = $(\{1,3\})$.

Steps 2–4. MIS's of $\langle\{2,3,5,6,8\}\rangle$ are $\{2,5,8\}$, $\{6,8\}$, $\{3,5\}$, $\{3,6\}$.

Step 5. $u = 2$, $n = 2$, $e_2 = 1$, STACK = $(\{1,3\}, \{2,5,8\})$.

Steps 6–8. $e_2 = 0$, $b_2 = \{2,5,8\}$, $T = \{1,2,4,5,7,8\}$.

Steps 2–4. MIS's of $\langle\{3,6\}\rangle$ is $\{3,6\}$.

Steps 5–8. $u = 3$, $n = 3$, $e_3 = 0$, $b_3 = \{3,6\}$, $T = V$, STACK = $(\{1,3\})$.

Steps 2–3. $g = 3$, $T = \{1,2,4,5,7,8\}$, $c = (\{1,4,7\}, \{2,5,8\}, \{3,6\})$.

Steps 6–7, 6–7, 6–8. $e_1 = 0$, $n = 1$, $b_1 = \{1,3\}$, $T = \{1,3\}$, STACK = ( ).

[1] The reader will find the above graph minus the edge joining 4 and 6 possesses a more interesting subgraph tree. The trace for this graph is too long to be included here.

Steps 2–4. MIS's of $\langle\{2,4,5,6,7,8\}\rangle$ are $\{2,5,8\}$, $\{2,5,7\}$, $\{6,8\}$, $\{2,4,7\}$.

Step 5. $u = 4$, $n = 2$, $e_2 = 1$, STACK = $(\{2,4,7\})$.

Steps 6–8. $e_2 = 0$, $b_2 = \{2,4,7\}$, $T = \{1,2,3,4,7\}$, STACK = ( ).

Steps 2–4. MIS's of $\langle\{5,6,8\}\rangle$ are $\{5,8\}$, $\{6,8\}$.

Step 5. $u = 5$, $n = 3$, $e_3 = 1$, STACK = $(\{5,8\})$.

Steps 6–8. $e_3 = 0$, $b_3 = \{5,8\}$, $T = \{1,2,3,4,5,7,8\}$, STACK = ( ).

Steps 2, 6–7, 6–7, 6–7, 6. Stop. $n = 0$, $T = \varnothing$, STACK = ( ), $g = 3$, $c = (\{1,4,7\}, \{2,5,8\}, \{3,6\})$.

ACKNOWLEDGMENTS. The author wishes to thank Professor Patricia Eberlein for her advice and encouragement in the preparation of this paper.

REFERENCES

1. CHRISTOFIDES, N. An algorithm for the chromatic number of a graph. *Computer J. 14*, 1 (Feb. 1971), 38–39.
2. HARARY, F. *Graph Theory*. Addison-Wesley, Reading, Mass., 1969.
3. ERDÖS, P., AND RÉNYI, A. On random graphs. I. *Publicationes Mathematicae (Debrecen) 6* (1959), 290–297.
4. ERDÖS, P., AND RÉNYI, A. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci. 5A* (1960), 17–61.
5. ERDÖS, P., AND RÉNYI, A. On the strength of connectedness of a random graph. *Acta Math. Acad. Sci. Hung. 12* (1961), 261–267.
6. ERDÖS, P., AND RÉNYI, A. On the existence of a factor of degree one of a connected random graph. *Acta Math. Acad. Sci. Hung. 17* (1966), 359–368.
7. MATULA, D. On the complete subgraphs of a random graph. Proc. of the Second Chapel Hill Conference on Combinatorial Mathematics and Its Application, U. of North Carolina, 1970, pp. 356–369.
8. WIRTH, N. *The Programming Language Pascal*. Berichte der Fachgruppe Computer-Wissenschaften, Eidgenossische Technische Hochschule, Zurich, Switzerland, 1970.
9. MULLIGAN, G. D., AND CORNEIL, D. G. Corrections to Bierstone's algorithm for generating cliques. *J. ACM 19*, 2 (April 1972), 244–247.

RECEIVED JULY 1972; REVISED APRIL 1973