01 Aug 1988

Composite Graph Coloring Algorithms and Applications

Stephen Hong Seng Yek

Billy E. Gillett
Missouri University of Science and Technology

COMPOSITE GRAPH COLORING ALGORITHMS AND APPLICATIONS

S. H. S. Yek* and B. E. Gillett

CSc-88-7

Department of Computer Science

University of Missouri-Rolla

Rolla, Missouri 65401 (314)341-4491

*This report is substantially the M.S. thesis of the first author, completed August 1988.

ABSTRACT

A vertex-composite graph is a graph that can have unequal chromaticities on its vertices. Vertex-composite graph coloring or composite graph coloring involves coloring each vertex of a composite graph with consecutive colors according to the vertex's chromaticity with no two vertices adjacent to one another having the same color(s).

New heuristic algorithms including the use of the saturation degree method have been developed in this research. All eleven heuristic algorithms including Clementson and Elphick algorithms were then tested using random composite graphs with five different chromaticity distributions. The best algorithm which uses the least average colors from the experiment is the MLF1I algorithm follow very closely by the MLF2I algorithm.

Four applications, Timetabling, Job Shop Scheduling, CPU Scheduling and Network Assignment Problem, have been formulated as composite graph coloring problems and solved using heuristic composite graph coloring algorithms.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

I. INTRODUCTION

Graph coloring is a classical problem of graph theory and has been around for a long time. It has applications in many areas such as code design, circuit troubleshooting, distribution of computer memory, design of integrated circuits, timetabling, storage or transportation of goods, and scheduling of jobs.

In general, graph coloring is a coloring of a graph in which a color is assigned to each vertex of the graph in such a way that no two adjacent vertices connected by an edge have the same color. For a given graph, a vertex that can be assigned any one color is said to have a chromaticity of one. Standard graph coloring involves coloring non-composite graphs or simply graphs having vertices of equal chromaticities. A graph is r-colorable if it can be colored with r or less colors. Figure 1 shows a graph with arbitrary assignment of colors to its vertices.

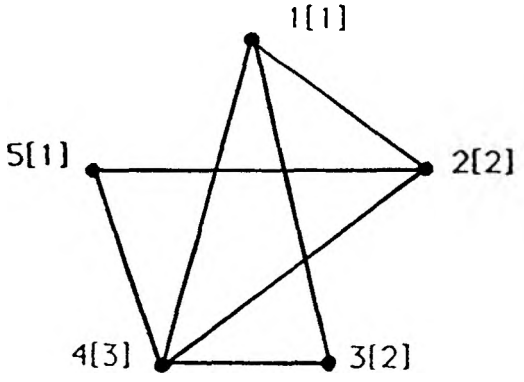

Figure 1. A coloring of a non-composite graph.

For any graph, G, there exists a minimum coloring (number) for that graph denoted by the chromatic number of the graph, $\chi(G)$. An optimal coloring of a graph, G, is one which uses exactly $\chi(G)$ colors. However, trying to determine the χ of any graph is, unfortunately, NP-complete, which means there is no known algorithm that can color any graph optimally in a time bounded by a polynomial function (1). The problem of determining the χ is classified as the standard graph coloring problem or graph coloring problem as cited in the literature. Algorithms computing χ exactly are applicable only to rather small graphs and because of their limitations, many heuristic (approximation) algorithms have been developed for solving larger graph coloring problems.

A number of search papers have been published in the area of the standard graph coloring problem. Results range from better algorithms to new applications using the graph coloring technique. An example from the algorithm side is a paper published by Peemoller (1) in 1986 which discussed the various algorithms and the results from the numerical experiments conducted . From the application side, Larus and Hilfinger (2) used the graph coloring method for storage (register) allocation in the LISP compiler.

A few years back as more efficient algorithms and new applications were introduced, Clementson and Elphick (3) proposed the composite graph coloring problem. The meaning composite relates to the color or chromaticity associated with each vertex of the graph. Each vertex of a graph can be allocated one or more colors. A graph is said to be composite

if its vertices' chromaticities are not all equal. The standard graph coloring algorithm only solves problems of non-composite graphs or simply graphs. As an illustration, Figure 2 shows an arbitrary coloring of a composite graph with vertices having unequal chromaticities.

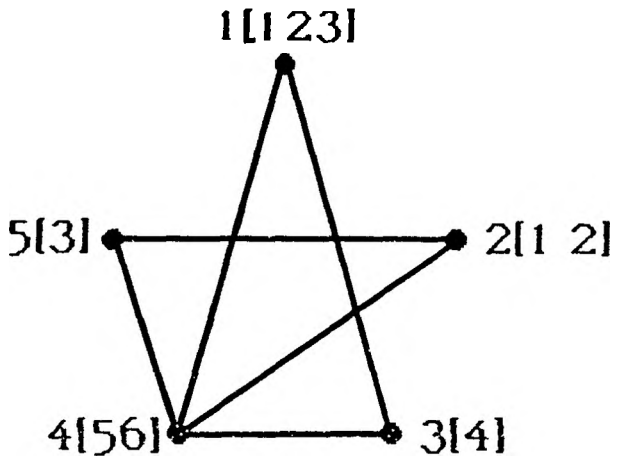

Figure 2. A simple coloring of a composite graph.

It is also known that finding the exact chromatic number of a composite graph is NP-complete. The composite graph coloring problem suffered the same fate as the standard graph coloring problem (4), inheriting the NP-completeness property. For this reason, attention has been focused on the development of heuristic algorithms which will hopefully produce good colorings for any composite graphs in a reasonable amount of time.

In the case of composite graph coloring, Clementson and Elphick (3) in 1983 presented four heuristic algorithms:

1) LF1 - Largest first by chromaticity,

2) LF1I - Largest first by chromaticity with interchange,

3) LF2 - Largest first by chromatic degree, and

4) LF2I - Largest first by chromatic degree with interchange.

All of these algorithms are the vertex-sequential type coloring algorithms which will be discussed in greater detail later in the literature review section.

These are not the only algorithms which have been published. In 1987, Johnnie Roberts (5) pursued on the same track and produced another eight heuristic algorithms. The eight algorithms are:

1) LFPH - Largest-first-by-pigeonhole-measure,

2) LFPHI - Largest-first-by-pigeonhole-measure-with-interchange,

3) LFCD - Largest-first-by-chromaticity-times-degree,

4) LFCDI - Largest-first-by-chromaticity-times-degree-with-interchange,

5) RLF1 - Recursive largest first coloring algorithm using chromatic degree,

6) RLFD1 - Recursive largest first coloring algorithm using degree,

7) DYNPH - Dynamic-pigeonhole-measure, and

8) DYNFPH - Dynamic-floating-point-pigeonhole-measure.

All the algorithms presented order the vertices by the following measures : (1) the chromaticities of the vertices, (2) the chromatic degrees of the vertices, and (3) the degree of the vertices. The LFPH algorithm (largest-first-by-pigeonhole-measure) orders the vertices of the graph by the pigeonhole principal. On the other hand, the LFCD algorithm (largest -first-by-chromaticity-times-degree) orders the vertices in decreasing order according to the product of the chromaticity and the degree of a vertex.

Both the LFPHI and LFCDI algorithms employ the same interchange technique used by Clementson and Elphick.

The RLF1 and the RLFD1 algorithms are the recursive-largest-first algorithms, generalizations of the RLF algorithm from the standard graph coloring problem presented by Leighton (6). The last two algorithms, the DYNPH (dynamic-pigeonhole-measure) and the DYNFPH (dynamic-floating -point-pigeonhole-measure) use the measures based upon the pigeonhole measure to determine the next vertex to be colored.

The perspective thrust of this research is to explore further possibilities of "better" algorithms and potential application areas in which composite graph coloring can be applied. The algorithms to be presented employ similiar strategies as listed by Roberts including a new one, the saturation degree method. The saturation degree method was first introduced by Brelaz (7) in the standard graph coloring problem. He defined the saturation degree of a vertex as the number of different colors used for vertices adjacent to it. It was shown that the saturation degree algorithm is exact for a group of graphs called bipartite graphs. As was noted by Spinrad and Vijayan (8), this algorithm has its own worst case behavior and may use n colors on 3-colorable graphs having O(n) vertices. However the algorithm performed much better than other heuristics on randomly chosen graphs (7,9).

Two algorithms using Brelaz's method have been developed. The algorithms, DS1I and DS2I, preorder the vertices and use the modified saturation degree methods to select the uncolored vertices. Whenever the algorithms try to color the vertices with higher color(s) than the maximum current color, the interchange procedure used by Clementson and Elphick is invoked to attempt reducing the number of colors used. Three algorithms using the largest-first strategy have also been developed. The LF3 algorithm, closely resembles the LF1I and LF2I algorihms, orders the vertices in decreasing chromaticities and sub-orders in decreasing degrees. The next two algorithms which are dynamic in nature used the modified saturation degree methods similiar to the DS1I algorithm and DS2I algorithm. MLF1 algorithm and MLF2 algorithm preorder the vertices in decreasing chromaticities and sub-orders with the modified saturation degree methods.

A more elaborate description of each of the new algorithms is presented in Chapter IV. The developed algorithms are tested by coloring groups of 25 random composite graphs with vertices, n = 100 and n = 200. Various edge density distributions including the truncated Poisson were used. The results of the experimentation are then presented later in Chapter VI.

Since the composite graph coloring concept is still relatively new to the world of graph coloring, there have been no reported applications in the literature. Four possible composite graph coloring applications were investigated. The first application which has been formulated as a standard

graph coloring problem is the Timetabling Problem. The Timetabling Problem involves scheduling of classes with no conflict of timeframes or scheduling of examination timetabling for students. The second application is the Job Shop Scheduling Problem. Jobs which require several operations to be processed without taking precedence of which operations are to be performed first can be formulated as a composite graph coloring problem. The third application is called the CPU Scheduling Problem and it involves scheduling the quickest way to run jobs on different processors. The last application is a network problem, called appropriately, the Network Assignment Problem. Given a center (location) disseminating vital information, the problem is to find the quickest way to send the information. The problem then can be constructed as a composite graph coloring problem.

A more detailed description of each applications is presented in Chapter V including examples of the applications solved by the composite graph coloring strategy.

II GRAPH DEFINITIONS

An ordered pair consists of two objects in a given fixed order and is not a set consisting of two elements (10, p.122). The ordering of two objects is important and the objects may not be distinct. An ordered pair is denoted by $<x,y>$. A familiar example of an ordered pair is the representation of a point in a two-dimensional plane in cartesian coordinates. Thus the notion of an ordered pair can be extended to define an ordered triple and more generally, an n-tuple.

A. GENERAL DEFINITIONS

A graph is an ordered triple (3-tuple), $G=<V,E,\phi>$ where V is a nonempty set called the set of vertices, nodes or points of the graph, E is said to be the set of edges of the graph and ϕ is a mapping from the set of edges E to a set of ordered or unordered pairs of elements of V (10, p.469). An edge $e \in E$ is associated with an ordered pair $<u,v>$ or an unordered pair (u,v) where $u,v \in V$. Thus the edge e connects or joins the vertices u and v, and the endpoints of the edge e are the vertices u and v. Vertices connected by an edge in a graph are called adjacent vertices. In a graph a vertex which is not adjacent to any other vertex is called an isolated vertex. A graph containing only isolated vertices is called a null graph. A graph in which every edge is undirected (connected by unordered pair of vertices) is called an undirected graph. If every edge of a graph is directed (connected by ordered pair of vertices) it is called a directed graph. The degree of a vertex v in a graph G, denoted by $d_G(v)$, is the number of edges connected

to the vertex v. A graph is said to be a complete graph if every vertex is adjacent to all other vertices in the graph. Figure 3 shows an undirected graph with V = $\{v_1,v_2,v_3,v_4,v_5\}$ and E = $\{e_1,e_2,e_3,e_4,e_5\}$. The mapping of vertices to edges are $\phi(e_1) = (v_1,v_2)$, $\phi(e_2) = (v_2,v_3)$, $\phi(e_3) = (v_1,v_3)$ and etc. The vertex v_5 is an isolated vertex. As an illustration, the vertices v_1 and v_2 are adjacent vertices of the edge e_1. The degree of the vertex v_1 is equal to three, i.e., $d_G(v_1) = 3$ and $d_G(v_4) = 2$.

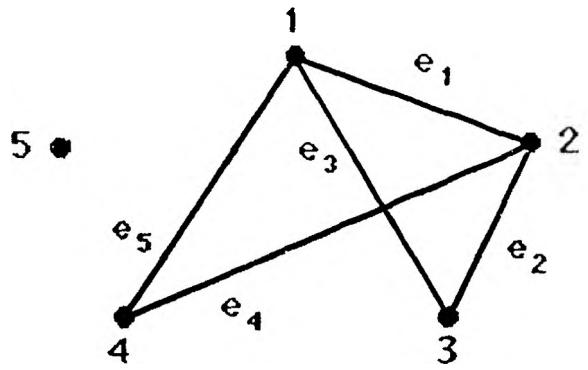

Figure 3. An undirected graph.

Figure 4 shows a complete graph. Note that all vertices are adjacent to all other vertices in the graph. A complete graph is also known to be a graph with an edge density of one. If not all the vertices are adjacent to all other vertices in the graph then the graph is said to have a density less than one depending on the percentage of the connectivity of the vertices.

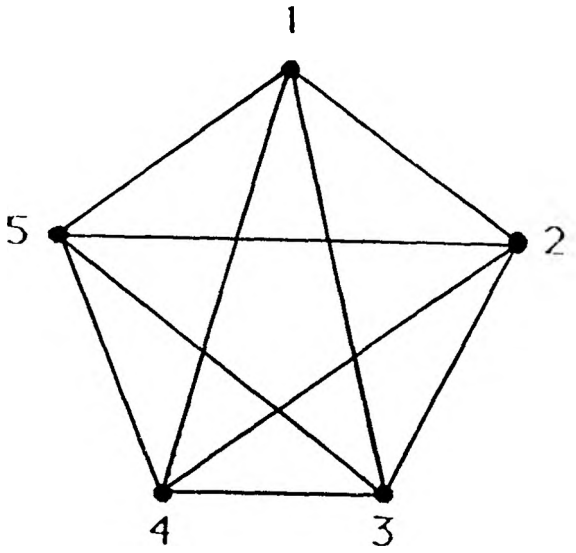

Figure 4. A complete graph.

The composite graphs can be defined in two ways: a) an edge-composite graph (e-composite graph) or b) a vertex-composite graph (v-composite graph) (3).

B. COMPOSITE GRAPH DEFINITIONS

An e-composite graph G is a triple <V,E,C(E)> where V is a non-empty finite set of vertices, E is a finite set of edges and C(E) is a finite set of positive integers called edge chromaticities. Thus the graph can have multiple edges with any pair of vertices connected by more than one edge. Associated with the edge set $\{e_i\}$ are the chromaticities, c^e_j's associated with each edge. The graph G is said to be K-edge colorable if to each of its edges e_j there may be assigned c^e_j of integers the integers (1,...,K) such that the integers assigned to each edge are distinct and consecutive, and such

that no two adjacent edges have an integer in common. If G is K-edge colorable, but not (K-1)-edge colorable, then the edge chromatic number of G denoted by $\chi(G)$ is K.

A v-composite graph is also a triple <V,E,C(V)> where V is a non-empty finite set of vertices, E is a finite set of edges and C(V) is a finite set of positive integers called vertex chromaticities. Thus associated with each vertex v_i is the vertex chromaticity c^{v_i}. The adjacency matrix $A = [a_{ij}]$ of G with n vertices is defined as a n X n matrix in which $a_{ij} = 1$ if v_i is adjacent to v_j and $a_{ij} = 0$ if otherwise. The graph G is K-colorable if the coloring to the vertices v_j with integers 1,...,K are distinct and consecutive and such that no two adjacent vertices have an integer in common. If graph G is K-colorable but not (K-1)-colorable, then the chromatic number of G denoted by $\chi(G)$ is K which is the exact coloring of the graph G. The vertex chromatic degree of vertex v_i , denoted by f_i, is given by

$$f_i = \sum_{j=1}^{n} a_{ij} c^{v_j} + c^{v_i}.$$

The total vertex chromaticity of G, denoted by c^v (G) , is given by

$$c^v(G) = \sum_{i=1}^{n} c^{v_i}.$$

III. LITERATURE REVIEW

Considerable literature in the field of graph theory has dealt with the coloring of graphs as can be read from Ore's extensive book "The Four-Color Problem" (11). The coloring problems at the time were concerned with coloring planar graphs. Graphs which can be drawn in the plane without any two of its edges crossing each other are called planar graphs. In the five-color conjecture it was proven that every planar graph can be colored with five colors so that no two adjacent vertices are of the same color. In 1976, Appel and Haken (12) proved the famous four-color conjecture to be true by using computer generated planar graphs to show that planar graphs are 4-colorable. It has been debated since then because the conjecture could not be proven without the use of a computer.

A. THE STANDARD GRAPH COLORING PROBLEM

A number of problems can be formulated as a graph coloring problem, for example :

i) the construction of examination timetables by Wood (13),

ii) the construction of school timetables by Williams (14),

iii) the assignment of frequencies for radio stations by Hale (15),

iv) the storing of parser tables by Dencker (16).

There are more but these examples will suffice to show the importance of the graph coloring problem and provide the foundation to the understanding of the composite graph coloring problem. Over the years many heuristic standard graph coloring algorithms have been developed to produce better

colorings. In 1967, Welsh and Powell (20) proposed an algorithm called the largest-first (LF) in which the vertices of the graph are arranged in non-increasing order of their degrees and are colored sequentially. This algorithm in a slightly different but equivalent form is mentioned by Wood (13), Christofides (19), and Matula et al (17). Today, the algorithm is called the vertex-sequential coloring algorithm. There are also two other types of algorithms, the vertex-sequential with interchange and the color-sequential coloring algorithms (5).

1. Vertex-sequential coloring algorithms. Algorithms belonging to this type arrange or preorder the vertices of the graph in some order and then assign a color to each of the vertices in the preordered sequence with the least possible color. There are essentially two algorithms in the literature which use this approach. They are the largest-first (LF) and the smallest-last (SL) algorithms (17). The LF algorithm as described earlier, orders the vertices in the order of decreasing degrees, i.e., $d(v_i) \geq d(v_{i+1})$ for $1 \leq i < n$ where $V = \{v_1, ..., v_n\}$. The SL algorithm is similiar to LF in strategy but it recursively orders the smallest degree vertices last. That is, it orders the vertices in an order in which each vertex has the smallest degree in the induced subgraph on the set of vertices preceding and including the vertex. More formally, SL ordering selects $d(v_n) = \min_{w \in V} d(w)$ and for $n-1 \geq i > 1$, $d_v(v_i) = \min_{w \in U} d_v(w)$ where $U = V - \{v_n, ..., v_{i+1}\}$.

Both the LF and SL algorithms tend to order the high degree vertices first before the low degree vertices. As noted by Leighton (6), from

computational experience, this is generally a good strategy compared with algorithms which color the higher degree vertices last.

2. Vertex-sequential with interchange coloring algorithms. An algorithm of this type colors graphs using the same method as the vertex-sequential algorithms with the addition of the interchange module. The interchange module is invoked each time the algorithm tries to use a color that has not been assigned earlier in the coloring process. The purpose of the interchange process then is an attempt to prevent the introduction of any further new color in the graph by trying to use the available colors previously assigned.

Given any graph G = (V,E) and color function f such that f(w) $\in$ {i,j} for all w $\in$ V, then (i,j)-interchange on G is a redefinition of f such that if f(w) = i previously, f(w) is now assigned j and vice versa for all w $\in$ V. The interchange module has been shown to yield good results when used in conjunction with LF and SL algorithms. The following gives an illustration of how the interchange process works. Let v_m be the current vertex and K be the current largest color used, denoted by $K = \max_{i<m} f(v_i)$. For $1 \leq i < j \leq K$, let G_{ij} be the subgraph of G induced by the vertices of G previously colored i or j. If possible, select i and j such that no connected component of G_{ij} contains two differently colored vertices both adjacent to v_m. If such a G_{ij} exists, then perform an (i,j)-interchange on each connected component of G_{ij} which contains an i-colored vertex adjacent to v_m in G. Now it is possible to assign the color i to v_m and thus the addition of a new color, K+1, has

been avoided. However, if no G_{ij} exists, then the interchange will fail and v_m will be assigned the color K+1.

In the literature two interchange methods were introduced one by Matula, Marble and Issacson (17) and an extended version by Johnson (5,6,18). Leighton (6) reported that based upon a limited amount of computational experience, the extended version seemed to produce slightly better results than the other interchange method.

3. Color-sequential coloring algorithms. The color-sequential coloring algorithms color a graph by coloring all the possible vertices with color k first before proceding to color other vertices with color k+1. One color-sequential coloring algorithm may start out by assigning color 1 to all vertices in the graph that can be assigned color 1. These vertices form an independent set (19), not necessarily a maximum independent set (AMIS), of the subgraph. Upon completing the coloring of all possible vertices with color 1, the next set of vertices (next independent set) are colored with color 2 and so on until all the vertices in the graph are colored. Note, each remaining uncolored vertex is always adjacent to at least one vertex that has been assigned the color i for each i $\in$ I [1,K]. There were four color-sequential coloring algorithms in the literature proposed by Welsh and Powell (20), Williams (14), Johnson (18) and Leighton (6). A more detailed description of the various algorithms can be read in Roberts' (5) dissertation.

B. THE COMPOSITE GRAPH COLORING PROBLEM

1. Vertex-sequential coloring algorithms. A vertex-sequential coloring algorithm for coloring the composite graphs preorders the vertices of the graph to be colored in some order and then colors the vertices in accordance with the pre-determined ordering. The algorithm is similiar in strategy to the standard vertex-sequential coloring algorithms discussed earlier in the standard graph coloring section of this chapter except when coloring the individual vertices due to the unequal chromaticities of the composite graph. Using the notation $< v_1,..., v_j >$ as the subgraph induced by the vertex set $\{v_1,..., v_j\}$ and an ordering $v_1,..., v_n$ of the vertices of a graph G, the algorithm colors the graph as follows:

a) v_1 is assigned colors 1,2,3,..., c^{v_1} since v_1 has a chromaticity of c^{v_1}.

b) if $< v_1,..., v_{i-1} >$ has been j-colored, then $v_1,..., v_{i-1}$ have the same colors in $< v_1,..., v_i >$ and v_i is assigned colors $(m+1),...,(m+ c^{v_i})$, where $m \leq$ j is the minimum integer such that none of colors $(m+1),..,(m+ c^{v_i})$ have been assigned to vertices in $< v_1,..., v_{i-1}$ which are adjacent to v_i.

Clementson and Elphick (3) described two algorithms, LF1 and LF2, which use this strategy. Along the same line of development, Roberts (5) described two more algorithms, LFPH and LFCD. Both the LF1 and LF2 algorithms require a preordering of vertices using the largest vertex first ordering. The largest vertex first ordering unlike the original LF algorithm does not order vertices according to the highest degree rule. In the LF1 algorithm, the vertices are ordered in decreasing chromaticity order and

vertices having equal chromaticities are sub-ordered in decreasing chromatic degree order. The chromatic degree of a vertex v_i is the sum of chromaticities of the vertices adjacent to it plus the chromaticity of vertex (v_i). In the LF2 algorithm, the vertices are first ordered in decreasing chromatic degree and if ties exist, the vertices in question are sub-ordered in decreasing chromaticity order.

The largest-first-by-pigeonhole-measure (LFPH) algorithm orders the vertices in decreasing static pigeonhole measure order. The static pigeonhole measure of a vertex is the pigeonhole measure of the vertex for conditions prior to coloring any vertices of the graph. Further information about the pigeonhole principal can be obtained from Roberts' (5) dissertation. The next algorithm, LFCD, is called the largest-first-by -chromaticity-times-degree algorithm and it arranges the vertices in decreasing order by the value of the product of a vertex's chromaticity and its degree.

2. Vertex-sequential with interchange coloring algorithms. The algorithms are similiar to the standard graph coloring vertex-sequential coloring algorithms. Vertices of a composite graph are ordered according to some strategy and are colored with the lowest possible sequence of colors. The interchange module is invoked whenever the algorithms tries to color a vertex using color(s) that has (have) not been assigned previously. In the standard graph coloring, two interchange methods were discussed; in the composite graph coloring the only interchange method currently

available involves changing the colors of two vertices, the current vertex and an adjacent vertex in such a way both vertices will be assigned the sequence of colors that have been used currently so that no new color(s) will be used.

The following is a description of how the interchange method works. Let vertex v_i, with chromaticity c^{v_i} , be the next vertex to be colored and let the current number of colors in use be denoted by K. Assume the sequential-vertex coloring algorithm assigns the colors j,(j+1),...,(j+ c^{v_i} -1) to vertex v_i, then

a) if $(j + c^{v_i} - 1) \leq K$, then these colors will be assigned to v_i and the algorithm proceeds to color v_{i+1}.

b) if $(j + c^{v_i} - 1) > K$, then the interchange module is invoked and an attempt is made to reduce the number of colors currently being used in the coloring. The attempt is made as follows:

   i) Try assigning each of the initial colors 1,...,(j-1) to v_i. A subset P = $\{P_k\}$, k = 1,...,r of the integers 1,...,(j-1) is constructed such that, if v_i is assigned the initial color P_k , then only one of the vertices $v_1,\ldots, v_{i-1}$ adjacent to v_i is colored with one or more colors $P_k,\ldots,(P_k + c^{v_i} - 1)$. Therefore, locate a vertex v_k with chromaticity c^{v_k} which is adjacent to v_i being colored with one or more colors $P_k,\ldots,(P_k + c^{v_i} - 1)$ for each corresponding element P_k of P.

   ii) If P = ϕ, v_i is colored j,...,(j+ $c^{v_i} - 1$) and the algorithm proceeds to color vertex v_{i+1}. Denote the set of permissible initial colors for v_k by Q = $\{Q_k\}$, k = 1,...,r. A subset R of P, $P_k \in$ R is constructed only if

$Q_k \leq (j + c^{v_i} - c^{v_k} - 1)$ to ensure that the last color used by v_k is not greater than (j+ c^{v_i} -1).

iii) If R = ϕ, no color reduction can be produced and v_i is colored j,...,(j+ $c^{v_i} - 1$) and the algorithm proceeds to color vertex v_{i+1}. Otherwise (R $\neq \phi$) , the total number of colors used up to v_i can be reduced by changing colors with one of the vertices in $v_1,\ldots, v_{i-1}$. Define k_0 as follows:

$$\max(P_{k_0} + c^{v_i}, Q_{k_0} + c^{v_{k_0}}) = \min_k \max(P_k + c^{v_i}, Q_k + c^{v_k}).$$

Vertex v_i is assigned the initial color P_{k_0} and vertex v_k is assigned the initial color Q_{k_0}, and the algorithm proceeds to color vertex v_{i+1}. Essentially when R $\neq \phi$, there exists sets of sequences of colors for vertex v_i and the adjacent vertex v_k which are less than $(j + c^{v_i} - 1)$. The above expression simply states to select the minimum set of sequence of colors for v_i and v_k such that the two sets of sequences of colors selected are the least from the sets of sequences of colors avaiable.

Four algorithms, LF1I, LF2I, LFPHI and LFCDI, belong to the vertex-sequential with interchange algorithms. The first two were the product of Clementson and Elphick (3), and the last two the product of Roberts (5).

3. Color-sequential coloring algorithms. As with the standard graph coloring problem, Roberts proposed two new color-sequential coloring algorithms for composite graphs. The nature of the color-sequential coloring algorithms for composite graphs is similiar to the color-sequential coloring algorithms found in the standard graph coloring. The RLF1 and the RLFD1 coloring algorithms described by Roberts (5) are generalizations of the RLF coloring algorithm presented by Leighton (6). Both algorithms color vertices by the order of the chromaticity of a vertex, similiar in strategy as the LF1 coloring algorithm. In the two algorithms, the next vertex to be colored is a vertex with the highest chromaticity from the set of all uncolored vertices that are not adjacent to a vertex that has been assigned the current color. If ties occur with equal chromaticity, both algorithms prescribed a set of criterion to break the ties.

4. Other algorithms. Roberts (5) introduced a new concept in the stategy of coloring the vertices of a graph. The concept, called the pigeonhole measures, is based on the pigeonhole principle. Low integer values are assigned to vertices that are easy to color and higher values to vertices that are likely to use new colors in the coloration. Two algorithms, the dynamic-pigeonhole-measure (DYNPH) and the dynamic-floating-point-pigeonhole-measure (DYNFPH) use the pigeonhole measures to determine the order in which the vertices of a composite graph are colored.

5. Applications of the coloring problem. The composite graph coloring problem was proposed initially by Punter (21) to overcome the limitation of the standard graph coloring problem formulation which causes models to be inflexible. An example is the school timetabling problem which require some multiple period lessons and these multiple periods pose a problem in the assignment of consecutive colors to the vertices which represent the multiple periods. According to Clementson et al (3), the composite graph coloring problem is applicable to scheduling problems and store economy problem. The store economy deals with reduction of memory size required by a program.

IV. NEW HEURISTIC ALGORITHMS

Roberts (5) reported that there is no single heuristic algorithm which can color all composite graphs with a minimum number of colors. He observed that the largest-first algorithms tend to use more colors as the number of vertices of the graph increase and the opposite is true for the recursive largest-first algorithms. One observation from the results is that an algorithm which uses the largest-first strategy up to now can produce good sub-optimal colorings. Another observation is, for lower edge density, 20% and less, the largest-first algorithms produced better colorings than any other algorithms known.

In realistic application of graph colorings, Leighton (6) and Lotfi (22) used the standard graph coloring algorithms to solve the examination timetabling and course scheduling application using graphs having edge densities approximately 18% and 3.93%. Since such applications exist for the non-composite (standard) graph coloring problem then similiar applications should exist for the composite graph coloring problem.

Armed with the above observations, then further investigation into new heuristic algorithms would be necessary to see if there are other feasible algorithms that can be included into the set of good algorithms.

A. DEGREE SATURATION ALGORITHMS

The degree saturation algorithms originated from the standard graph coloring algorithm described by Brelaz (7). The algorithm for the standard graph coloring called the Dsatur algorithm, colors vertices of a non-composite graph relying upon the comparison of the degrees and structure of a graph. It uses the saturation degree of vertices to determine which vertex to color first. The saturation degree of a vertex is defined to be the number of different colors to which it is adjacent to colored vertices. In Brelaz's report the Dsatur algorithm was proven to be an exact algorithm for bipartite graphs and has a running polynomial time of $O(n^2)$. A bipartite graph is a graph whose vertices can be divided into two disjoint groups with each edge having one vertex in each group. As an illustration, Figure 5 shows a bipartite graph with 7 vertices. Vertices 1, 2 and 3 belong to one group while vertices 4, 5, 6 and 7 belong to another group.

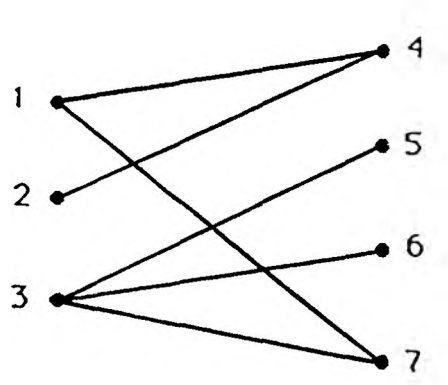

Figure 5. A bipartite graph.

The algorithm is also an important part of heuristic procedures to find maximal cliques in general graphs.

Two composite graph coloring algorithms using this measure have been developed. The saturation degree of a composite graph vertex (an uncolored vertex) is defined as the number of colors in the sequences of colors of vertices adjacent to it. As an illustration, if vertex 1 is colored 1,2 and 3, vertex 2 is colored 2,3,4 and 5, and vertex 3 is an uncolored vertex adjacent to vertices 1 and 2, then the saturation degree of vertex 3 is 5. The first algorithm is called the DS1I algorithm uses the same procedure as the Dsatur algorithm from the standard graph coloring except when there is a tie in selecting the maximal saturation degree, the vertex that has the largest chromatic degree will be selected to color first. If there is an equality in the chromatic degree selection, the first vertex with the equal chromatic degree is selected for coloration. Furthermore, if the vertex to be colored uses higher colors than the current maximum color the interchange procedure is invoked to attempt reducing the number new colors. An outline of the DS1I algorithm is given as follows:

- DS1I Algorithm -

1) Select a vertex with the largest (maximal) degree.
2) Color the vertex of maximal degree according to the vertex's chromaticity, i.e., 1,..., c^{v_i}.
3) Select a vertex with a maximal saturation degree. If there is an equality, select any vertex of maximal chromatic degree in the uncolored subgraph.

4) Color the chosen vertex with the least possible (lowest numbered) sequence of color(s). If the colors used exceed the current maximum color, invoke the interchange procedure.
5) If all the vertices are colored, stop. Otherwise, return to step 3.

The second algorithm is called the DS2I algorithm. It is similiar to DS1I except in step 3, chromatic degree is replaced by chromaticity. a vertex which has the largest chromaticity to be colored first. An outline of the DS2I algorithm is as follows:

- DS2I Algorithm -

1) Select a vertex with the largest (maximal) degree.
2) Color the vertex of maximal degree according to the vertex's chromaticity, i.e., 1,..., c^{v_i}.
3) Select a vertex with a maximal saturation degree. If there is an equality, select any vertex of maximal chromaticity in the uncolored subgraph.
4) Color the chosen vertex with the least possible (lowest numbered) sequence of color(s). If the colors used exceed the current maximum color, invoke the interchange procedure.
5) If all the vertices are colored, stop. Otherwise, return to step 3.

When the two composite coloring algorithms start coloring a graph, there is a possibility that there may be more than one vertex having the maximal degree. In both cases, the first vertex having equal maximal degree will be colored first.

An exact algorithm for coloring composite bipartite graphs equivalent to the Dsatur algorithm can be constructed by modifying step 3 and step 4 of either the DS1I algorithm or the DS2I algorithm. In step 3, when there is an equality in selecting the maximal saturation degree, the tie is broken by selecting any vertex of maximal degree. In step 4, the only modification is the removal of the interchange procedure. The algorithm with the modified steps, DS0, is exact for composite bipartite graphs. The DS0 algorithm was compared with DS1I algorithm and DS2I algorithm in a set of trial random composite graphs. The results indicated DS1I and DS2I were using less colors on the average compared to DS0. However, when the three algorithms were run using a set of composite bipartite graphs, all three algorithms produced the same colorings. As a result, only the DS1I algorithm and the DS2I algorithm were used in the actual experimentation with other heuristic algorithms.

B. VERTEX-SEQUENTIAL ALGORITHMS

Three algorithms based on the vertex-sequential method have been developed. The first algorithm called LF3I is in parallel with LF1I and LF2I algorithm in terms of strategy. The LF3I algorithm starts by arranging the vertices in an order before sequentially coloring the vertices. Whenever the algorithm tries to use color(s) higher than the current maximum color, the interchange procedure is invoked.

Given a composite graph, the LF3I algorithm sorts the vertices using heap sort to order the vertices in decreasing chromaticity. If more than one vertex has the same chromaticity, the vertices in question are then sub-order in decreasing degree. The algorithm then colors the vertices sequentially according to the sorted order. As mentioned above, the interchange procedure is invoked when the algorithm uses new color(s). Below is an outline of the LF3I algorithm.

- LF3I algorithm -

1) Order the vertices of the composite graph by decreasing chromaticity and sub-order in decreasing degree.
2) Color the ordered vertex starting with the highest chromaticity first.
3) Check for new color(s) used. If so, call the interchange procedure.
4) If all vertices have been colored, stop. Otherwise go back to step 2.

The other two algorithms, MLF1 and MLF2, incorporate the saturation degree method while using the largest first strategy. The vertices are ordered in decreasing chromatic degree - the same initial ordering of vertices as the LF1 algorithm. When the algorithms find that there are two or more vertices having equal chromaticity, one of the vertices is selected for coloration based on the maximal saturation degree rule. Unlike the DS1I algorithm and DS2I algorithm, the vertices to be colored sequentially are fixed within the chromaticity ordering. Vertices with equal chromaticity in the ranking of the chromaticity ordering are the only ones subjected to the ordering by saturation degree method. The difference between MLF1 algorithm and MLF2 algorithm lies in the saturation degree method and is

shown in step 2 of the outline of each algorithm. The following is an outline of the MLF1 algorithm. The outline of MLF2 algorithm follows immediately after the outline of MLF1 algorithm.

- MLF1 <MLF1I> Algorithm -

1) Order the vertices of the composite graph by decreasing chromaticity and sub-order in decreasing chromatic degree.
2) Check to see if there is more than one vertex with the same chromaticity. If none, color the vertex with the minimum sequence of color(s). Otherwise find the vertex with the maximal saturation degree and color it with the minimum sequence of color(s). If a tie occurs in the selection of maximal saturation degree, select the vertex with the largest degree.
3) <Interchange>: Check for new color(s) used. If so, invoke the interchange procedure. Otherwise go on to the next step.
4) If all vertices have been colored, stop. Otherwise go back to step 2.

- MLF2 <MLF2I> Algorithm -

1) Order the vertices of the composite graph by decreasing chromaticity and sub-order in decreasing chromatic degree.
2) Check to see if there is more than one vertex with the same chromaticity. If none, color the vertex with the minimum sequence of color(s). Otherwise find the vertex with the maximal saturation degree and color it with the minimum sequence of color(s). If a tie occurs in the selection of maximal saturation degree, select the vertex with the largest chromatic degree.

3) <Interchange>: Check for new color(s) used. If so, invoke the interchange procedure. Otherwise go on to the next step.

4) If all vertices have been colored, stop. Otherwise go back to step 2.

The interchange procedure included in the outline of both of the algorithms is used to enhance the original algorithms. Using the interchange procedure then produces two enhanced algorithms, MLF1I and MLF2I respectively. All seven new heuristic algorithms in this chapter were used in the experimentation to be described in Chapter VI.

V. APPLICATIONS OF COMPOSITE GRAPH COLORING

A. TIMETABLING PROBLEM

The timetabling problem has been a major topic in the application of the standard graph coloring problem. Wood (13) and Welsh and Powell (20) have shown the basic examination scheduling problem is a standard graph coloring problem. Each vertex of the graph is represented by each course examination and each undirected edge is introduced between two vertices (courses) if the two courses cannot have the examinations at the same time frame. By minimizing the number of colors of the graph, the equivalent formulation then represents the objective of the examination scheduling which is to minimize the number of time frames used for the overall scheduling problem.

Recent timetabling applications still use the standard graph coloring method - see Lotfi and Sarin (22) and Mehta (23). Punter (21) observed that most schools require some multiple time frames for certain courses and these courses are represented by multiple vertices in the standard (non-composite) graph. In coloring such vertices, there is no guarantee that the colors (time frames) assigned to these vertices will be consecutive. The composite graph concept is introduced to overcome this limitation.

One of the composite graph coloring applications presented now is the timetabling problem. The objective is to assign instructors to classrooms in such a way that there is no conflict of classrooms for instructors. Note the timetabling problem could also be extended to examination scheduling

where there are unequal durations of examination time. As an example problem,

Prof. 1 uses classroom #202 for 4 courses.
T. Asst. 1 uses classroom #213 for 1 course.
T. Asst. 2 uses classroom #213 for 1 course.
T. Asst. 3 uses classroom #213 for 1 course.
T. Asst. 4 uses classroom #203 for 1 course.
Prof. 2 uses classroom #206 for 2 courses.
Prof. 3 uses classroom #206 for 1 course.
T. Asst. 5 uses classroom #202 for 1 course.
Prof. 4 uses classroom #203 for 2 courses.
Prof. 5 uses classroom #216 for 1 course.
T. Asst. 6 uses classroom #216 for 1 course.
(Note: Room 213 is a personal computer laboratory.)

The problem then is to schedule the classrooms (rooms) so that no two instructors will use the same room at the same time and the number of time frames used in the scheduling is minimum. Below is a procedure to formulate the problem as a composite graph coloring problem.

1) Assign the duration of class time to each room, i.e., all courses take 50-minutes except for the PC laboratory courses which take 1 hour and 50-minutes.
2) Label all rooms and let the same room number be under the same label. This step is essentially to simplify the identification of the room numbers.

Label the room numbers as follows: 1 1 1 1 2 2 2 3 4 4 4 1 3 3 4 4 eg., 1 represents room #202, 2 represents room #213 and etc.

3) Relabel the rooms to represent the number of vertices of the graph and join edges with vertices having the same room. This relabeling process is necessary to make the vertex numbers of the composite graph unique.

Label each room to correspond to each vertex of the graph:
Room: 1 1 1 1 2 2 2 3 4 4 4 1 3 3 4 4
Vertex: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

4) Use a composite graph coloring algorithm (MLF1I) to solve the problem. (The solution to the timetabling is shown in Figure 6.)

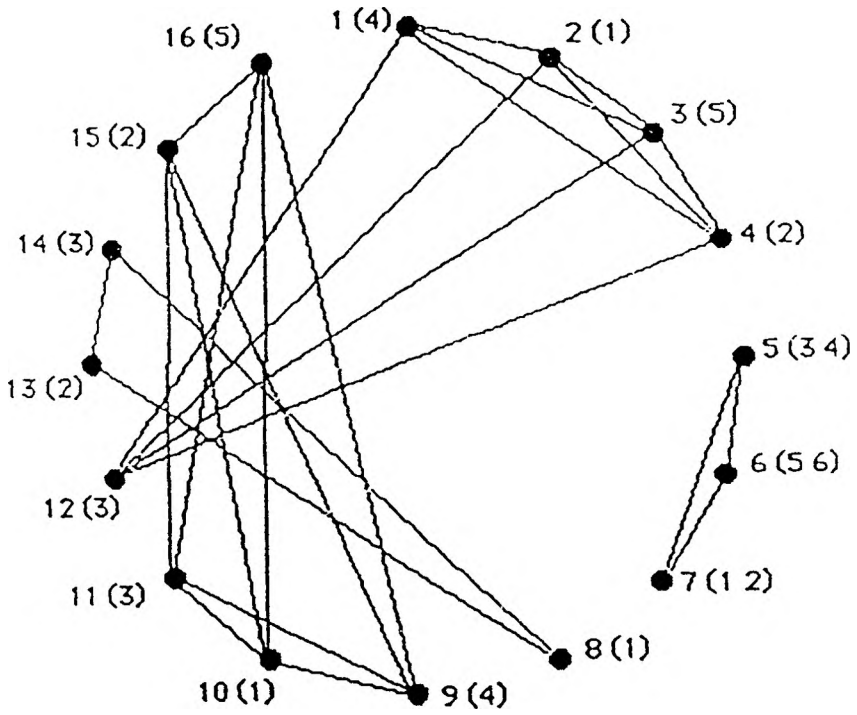

Figure 6. A graphical solution to the timetabling problem.

The number(s) in the parentheses represents the time period(s) the rooms corresponding to the vertices will be utilized. By mapping the vertex number back to the room number, the various instructors' time schedules can be found. If color 1 denotes the timeframe from 8:30 am to 9:20 am then

the sequence of colors on upwards represent the increments of color 1's time frame. For instance, vertices V1, V2, V3 and V4 represent room 202 traced to Professor 1. V2 and V4 colorings interpret to the professor's 8:30 am to 10:20 am class schedule. Similiar interpretations result in various instructors' class schedules as listed below:-

Solution to timetabling
Prof. 1 .... 8:30 am - 10:20 am, 11:30 am - 12:20 pm (Rm. 202)
T.A. 1 .... 10:30 am - 12:20 pm (Rm. 213)
T.A. 2 .... 12:30 pm - 2:20 pm (Rm. 213)
T.A. 3 .... 8:30 am - 10:20 am (Rm. 213)
T.A. 4 .... 8:30 am - 9:20 am (Rm. 203)
Prof. 2 .... 8:30 am - 9:20 am, 11:30 am - 12:20 pm (Rm. 216)
Prof. 3 .... 10:30 am - 11:20 am (Rm. 216)
T.A. 5 .... 10:30 am - 11:20 am (Rm. 202)
Prof. 4 .... 9:30 am - 11:20 am (Rm. 203)
Prof. 5 .... 9:30 am - 10:20 am (Rm. 216)
T.A. 6 .... 12:30 pm - 1:20 pm (Rm. 216)

B. JOB SHOP SCHEDULING PROBLEM

In a job shop setting, jobs which may be having several different operations are sent to work centers for processing. A job can have more than one operation depending on the job's requirement. A work center on the other hand may consist of a single machine or several machines working to perform a type of operation. The job shop scheduling activity is divided into two steps, allocating appropriate jobs to the right work centers (machine loading) and sequencing the order of jobs to be processed (job sequencing) at the work centers.

The job shop scheduling problem which will be solved using composite graph coloring is the problem of minimizing the total completion time of jobs at the work centers. The problem is complicated by the fact that there may be hundreds or thousands of individual jobs competing for time on limited work centers. Given a job shop scheduling problem, the problem can be represented by a composite graph. The vertices of the composite graph will be the jobs or the operations of the jobs. An edge is joined to the two vertices (operations) if both operations require the use of the same machine or are part of a job. A composite graph coloring algorithm can then be used to solve the shortest completion time for all jobs. An example problem is now given as follows:

Job A a(2), b(1); Job B a(2), c(3), d(2)
Job C d(2), e(1); Job D a(1)

The above information is interpreted as jobs with its given operations' processing time units in parentheses. For instance, job A involves two operations - operation 'a' takes 2 processing time units and operation 'b' takes 1 processing time unit. In reality such operations might be drilling operations taking 2 minutes on one drilling operation type and another 1 minute for the other drilling operation type. To set up the problem as a composite graph, let each operation represents a vertex in the composite graph. There are seven operations in the example problem and therefore seven vertices are needed in the composite graph. In Job A, operation 'a' is represented by vertex 1 and operation 'b' is represented by vertex 2. The operations of the rest of the jobs are represented in similiar fashion. As explained earlier, edges are then connected to the vertices if the operations

represented by those vertices require the same machine for processing or the operations are part of a job. The algorithm used for coloring this problem is the LF3I algorithm. Figure 7 shows the composite graph equivalent of the example problem already solved by the LF3I algorithm. The numbers in brackets beside the vertex numbers represent the operations' time and schedule for processing. Referring to Figure 7, vertex 1 or operation 'a' of Job A takes the first and second time units and vertex 2 or operation 'b' of Job A takes the third time unit. This means that operation 'a' will be processed first follow by operation 'b'. The minimum completion time for all the jobs is 7 time units. Similiar interpretations produce the solution on the next page.

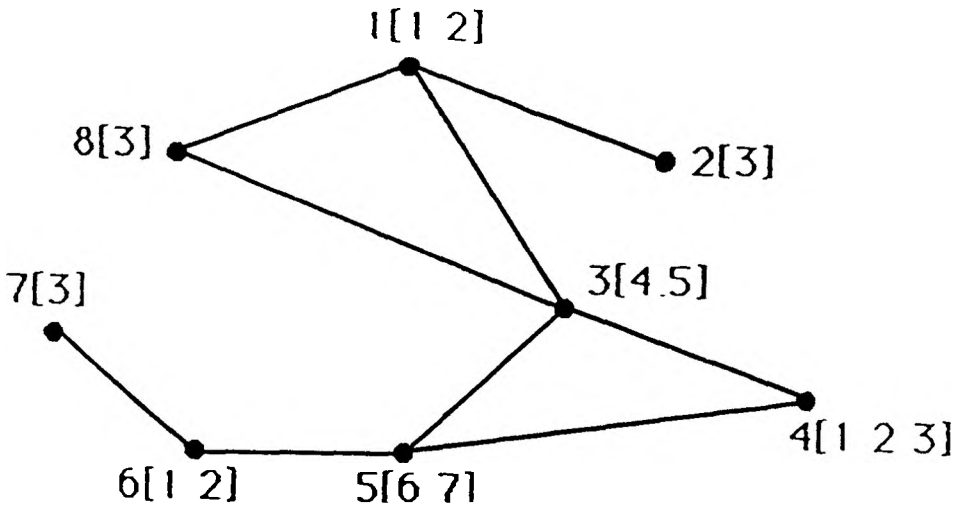

Figure 7. Composite graph of the job shop scheduling problem.

Solution:	Job	Operations
	A	a(1 2), b(3)
	B	a(4 5), c(1 2 3), d(6 7)
	C	d(2 3), e(1)
	D	a(3)

At the moment, minimizing the completion time of jobs at the work centers does not seem to pose any problems. However in another objective which is to minimize the completion time of each operation, the problem becomes difficult to solve. Finding the minimum completion time of jobs does not guarantee minimum completion time of each operation in the jobs as illustrated in Figure 8 of the same example problem.

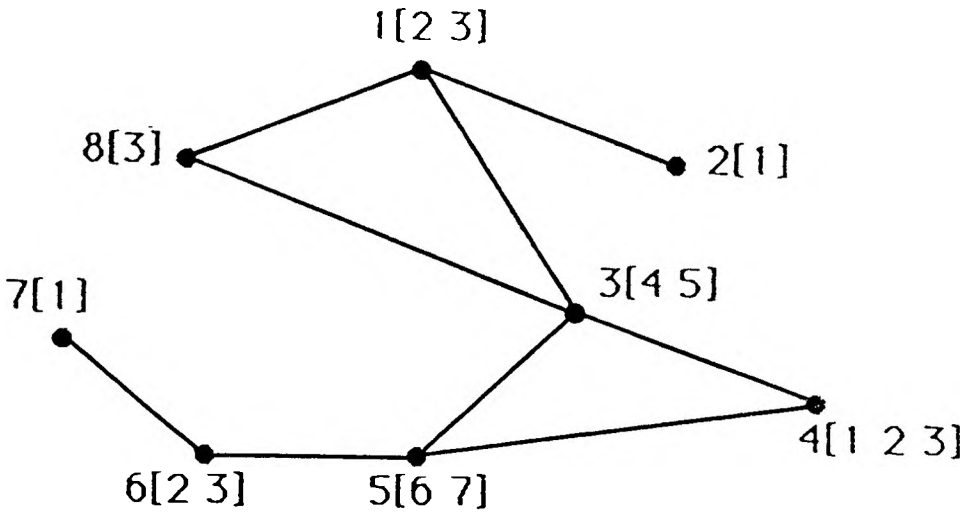

Figure 8. Graph with minimum completion time of each operation.

To calculate the minimum completion time of each operation add up the completion time of each operation of all jobs. In Figure 8 the minimum completion time of each operation is 24 time units whereas in figure 7 the minimum completion time of each operation is 28 time units. An algorithm was developed to handle the minimum completion time of each operation after a coloring algorithm solves the minimum completion time for all jobs. Unfortunately, the algorithm could not solve complex combination of jobs and related operations. Another problem from the composite graph formulation is that operations can not be prioritized due to the simplistic nature of the composite graph. A proposed modified composite graph is discussed later in Chapter VIII.

C. CPU SCHEDULING PROBLEM.

The CPU scheduling problem involves scheduling of jobs (programs or tasks) to be executed by processors. The objective of the problem is to find the shortest execution time for all processors and the minimum completion time for each job. To illustrate how the scheduling problem can be implemented as a composite graph coloring problem, an example problem is given:

Jobs in queue	CPU	Time factor
1	P1	3
2	P2	2
3	P3	4
4	P1	1
5	P2	3
6	P1	5
7	P3	2

Associated with each job is a time factor which is the execution time of each job. In this problem there are three processors P1, P2 and P3 being used by the various jobs coming in a queue waiting for execution. The CPU scheduling problem might be the problem of a job scheduler which controls the routing of the various jobs to the appropriate processors. In the example problem after collecting enough jobs in the queue, the job scheduler then queues the jobs in an order for the appropriate processors. Assuming there is no parallel processing involved in the various jobs, then all jobs are run sequentially.

In constructing the composite graph an important observation is made. Since each processor handles the jobs assigned to it sequentially, the composite graph represented by that processor is a complete composite graph. By including other processors, the overall composite graph is made up of complete composite sub-graphs represented by each of the processors. The CPU scheduling problem then can handle a single processor or multiple processor scheduling problem. Also since each processor (in the example problem) is represented by a complete composite sub-graph, the shortest execution time of each processor is the sum of all the execution times of jobs belonging to that processor - which solved the objective of finding the shortest completion time for all processors. To solve the shortest execution time for each job is the problem of finding the minimum completion time for each job. By observation, if the vertex (job) with the least time units is colored first then when totaling the completion time of the jobs, the optimal completion time for the jobs can be found. For

instance in the example problem, if job 2 is colored (4 5) and job 5 is colored (1 2 3), then the minimum completion time for the individual jobs adds up to 8 (5+3). However if job 2 is colored (1 2) and job 5 is colored (3 4 5), the completion time for the individual jobs adds up to only 7 (2+5). So in order to minimize the completion time of each job, job 2 will have to be run first and then follow by job 5 for processor P2. Note that since the jobs are processed sequentially, the overall shortest execution time of the two jobs for processor P2 is 5 which is the optimal time since the 2 jobs form the complete graph. An algorithm which colors the smallest chromaticity first was developed to solve this particular composite graph coloring application. Figure 9 shows the colored composite graph of the example problem solved by using the algorithm described above.

The vertices of the composite graph directly represent the jobs in the queue. In order to find the shortest completion time for all the processors and jobs the scheduler will have to order the jobs in the following manner:

CPU	Job	Time factor
P1	4	1
	1	2 3 4
	6	5 6 7 8 9
P2	2	1 2
	5	3 4 5
P3	7	1 2
	3	3 4 5 6

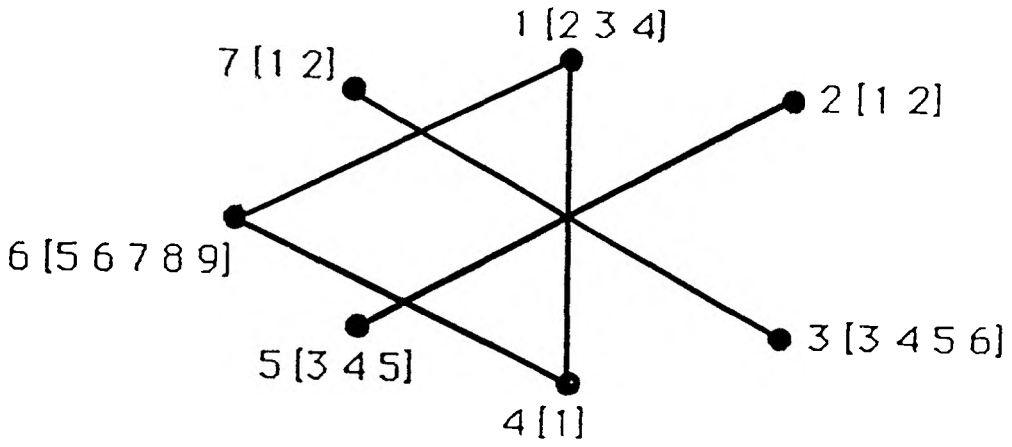

Figure 9. Composite graph of the CPU scheduling application problem.

D. NETWORK ASSIGNMENT PROBLEM.

The composite graph coloring method may have applications in the network problem. A case problem might be to determine the locations in a network for quickest dispersal of information through a network. Given, say, a company's central information center, the problem is to find locations that can transmit information relatively fast and at the same time reduce the number of transmissions handled by the information center. To construct the graph equivalent to the network, let the vertices be the company's locations and the edges joining the locations be the connection between the

locations. The importance of a location can be determined by locality of the location (or remoteness of the location) and the transmission factors of the location. The transmission factors are the chromaticities of the vertices (locations) and are evaluated from the efficiency and the effectiveness of the communication facility at that location. Consider a 12 location example:

Location 1 connected to location 2
Location 2 connected to locations 1, 3 and 4
Location 3 connected to location 2
Location 4 connected to locations 2, 3, 5 and 8
Location 5 connected to locations 4, 6 and 7
Location 6 connected to location 5
Location 7 connected to location 5
Location 8 connected to locations 4, 9, 10, 11 and 12
Location 9 connected to location 8
Location 10 connected to location 8
Location 11 connected to location 8
Location 12 connected to location 8

An algorithm which colors the largest degree first and sub-orders by highest chromaticity produces the coloring shown in brackets in Figure 10. To select the locations for receiving information from the information center, simply choose locations starting with the lowest colors beginning with color 1. In the example the chosen locations are:

a) Location 2

b) Location 5

c) Location 8

Note that this application has not been studied as a composite graph coloring application. The application is presented to motivate interest in the network problems.

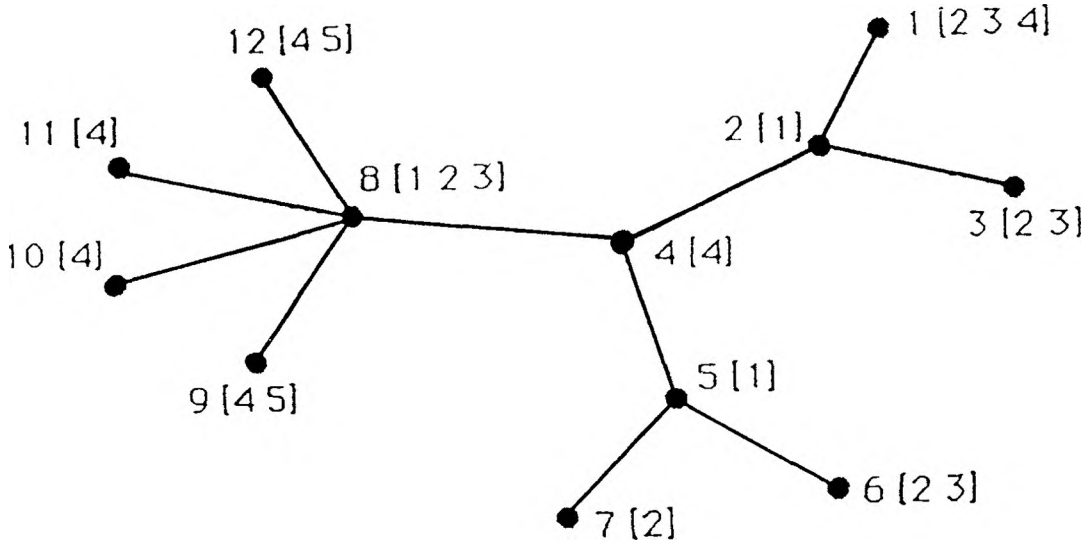

Figure 10. Composite graph representing the network example problem.

VI. RESULTS

A total of eleven heuristic algorithms were investigated using random composite graphs. A random composite graph $G_{<n,p,d>}$ is defined as a graph having n vertices with n(n-1)/2 possible edges occuring with probability $0<p<1$ (called the edge density), and the chromaticities of vertices determined by the chromaticity distribution, d. The objectives of the experiment were :

1) to corroborate and compare the results with Clementson et al (3) and Roberts (5) experiments.
2) to investigate the saturation degree methods and combination of chromaticity, chromatic degree and degree measures with the saturation degree methods.
3) to investigate the effects of varying the chromaticities, edge densities and increasing the number of vertices.

In the Clementson et al (3) experiments four (heuristic) algorithms, LF1, LF1I, LF2 and LF2I were compared with random composite graphs of 100 vertices using a truncated Poisson distribution with parameter $q = 1$ as the chromaticity distribution and with edge densities $p = 0.2$, 0.3 and 0.4. In this research, five chromaticity distributions were used. The five chromaticity distributions are:

1) truncated Poisson distribution - TP as abbreviation.
2) uniform distribution - UF as abbreviation.
3) 'down ramp' distribution - DR as abbreviation.
4) shifted binomial distribution - BN as abbreviation.
5) 'up ramp' distribution - UR as abbreviation.

Each of these distributions are described in detail by Roberts (5). The range of chromaticity values used in the chromaticity distributions in this experiment is between 1 and 10. A brief introduction to one of the chromaticity distributions is presented. The truncated Poisson expression is given by:

$$P(c^{v_i} = k) = q^k/(e^q - 1)k! \ \ where \ \ k = 1,2,...$$

In the experiment the parameter q is set to 1.0 and each integer increment of chromaticity, k, is calculated for k = 1,...,10. Rather than describing each of the chromaticity distributions further, a calculated and ready to use table for assigning the chromaticities to vertices is provided. Table I shows the cumulative values calculated for the five chromaticity distributions. To assign chromaticity to a vertex, a random number is generated and a comparison is made against the type of distribution. If the random number falls between the range of specified values then the chromaticity belonging to the range of specified values is assigned to the vertex. For instance if a random composite graph uses the TP distribution (truncated Poisson distribution), the chromaticity of a vertex is determined by the random number conversion. Suppose the random number has a value of 0.653, then comparing the cumulative values of Table I, chromaticity of k = 2 is chosen and assigned to that vertex since $0.582 \geq 0.653 > 0.873$.

Chromaticity Distributions					
k	TP	UF	DR	BN	UR
1	0.58198	0.1	0.190	0.002	0.010
2	0.87297	0.2	0.360	0.020	0.040
3	0.96997	0.3	0.510	0.090	0.090
4	0.99422	0.4	0.640	0.254	0.160
5	0.99907	0.5	0.750	0.500	0.250
6	0.99988	0.6	0.840	0.746	0.360
7	1.00000	0.7	0.910	0.910	0.490
8	1.00000	0.8	0.960	0.980	0.640
9	1.00000	0.9	0.990	0.998	0.810
10	1.00000	1.0	1.000	1.000	1.000

Table I. CUMULATIVE DISTRIBUTIONS OF THE CHROMATICITY DISTRIBUTIONS.

As part of the experiment, the random composite graphs were generated under the $G_{<n,p,d>}$ groupings. Graphs with vertices n = 100 and 200, p = 0.1,...,0.5 edge density together with chromaticity distribution of TP, UF, DR, BN and UR combined were tested using the 11 heuristic algorithms. The random composite graphs used were based on the $G_{<n,p,d>}$ groupings and to show how the the random composite graphs were generated in the groupings, the following pseudo code is provided:

```
for n = 100 and 200 do
    for p = 0.1 to 0.5 by 0.1 do
        for d = TP, UF, DR, BN, UR do
            for i = 1 to 25 by 1 do
                generate G<n,p,d>
```

In each $G_{<n,p,d>}$ grouping, 25 random composite graphs were used by the eleven heuristic coloring algorithms namely, LF1, LF1I, LF2, LF2I, LF3I, DS1I, DS2I, MLF1, MLF1I, MLF2 and MLF2I. The results of the colorings by the eleven algorithms are placed in the Table II to Table XI. To interpret the results in the tables, each algorithm's results are read column wise. The first row of the algorithm shows the number of wins coloring the 25 random composite graphs with respect to other algorithms. The second row shows the number of draws coloring the 25 random composite graphs with respect to other algorithms. The last row or third row represents the average colors used by the algorithm in coloring the 25 random composite graphs.

After examining Table II to Table XI for algorithms using the least colors, four algorithms, LF1I, LF3I, MLF1I and MLF2I, consistently were using less colors than other algorithms. To find out more about the nature of those algorithms, plots on the four algorithms were generated based on different chromaticity distributions, edge density variations and vertices expansion. The plots are shown on Figure 11 to Figure 36.

Table II. RESULTS OF EXPERIMENT IN COLORING RANDOM COMPOSITE GRAPHS.

(Vertices = 100, Edge Density = 10%)

	LF1	LF1I	LF2	LF2I	LF3I	DS1I	DS2I	MLF1	MLF1I	MLF2	MLF2I
a)	0	0	0	0	0	1	1	0	0	0	0
TP b)	5	14	0	2	15	6	4	8	21	8	21
c)	11.72	10.84	13.44	11.80	10.80	11.68	11.88	11.40	10.56	11.36	10.60
a)	0	1	0	0	4	2	4	0	2	0	0
UF b)	1	5	0	2	4	3	3	0	5	1	9
c)	39.72	36.52	43.76	38.44	36.36	37.48	37.64	39.60	36.60	39.64	36.28
a)	0	0	0	2	3	1	3	0	2	0	0
DR b)	2	9	0	0	9	1	4	2	10	1	10
c)	33.68	29.72	35.44	31.44	29.96	31.84	30.28	34.00	29.48	33.72	29.56
a)	0	3	0	2	3	3	4	0	1	0	0
BN b)	0	2	0	2	0	2	5	0	6	0	7
c)	43.36	38.56	44.24	39.24	38.40	38.64	37.96	42.92	38.44	42.68	38.60
a)	0	0	0	1	5	0	7	0	1	0	1
UR b)	0	3	0	0	5	4	3	0	5	0	3
c)	55.08	49.72	60.44	52.24	49.04	50.32	49.36	54.68	49.80	54.96	49.96

a) Row representing number of win(s).
b) Row representing number of draw(s).
c) Row representing average colors used in the 25 random composite graphs.

Table III. RESULTS OF EXPERIMENT IN COLORING RANDOM COMPOSITE GRAPHS.

(Vertices = 100, Edge Density = 20%)

	LF1	LF1I	LF2	LF2I	LF3I	DS1I	DS2I	MLF1	MLF1I	MLF2	MLF2I
a)	0	3	0	0	0	0	0	0	4	0	0
TP b)	0	7	0	1	9	0	1	3	16	2	15
c)	17.20	15.64	20.24	17.92	15.76	18.04	17.84	16.28	15.08	16.44	15.40
a)	0	6	0	0	4	0	0	0	2	0	4
UF b)	0	5	0	0	5	0	0	0	2	0	4
c)	58.04	53.04	66.04	57.92	53.12	58.72	57.56	58.44	53.28	58.16	53.20
a)	0	3	0	0	6	0	0	0	1	0	1
DR b)	0	6	0	1	8	0	0	0	12	1	10
c)	46.60	42.64	54.00	47.48	42.36	48.44	48.24	45.88	42.28	46.16	42.44
a)	0	4	0	1	4	0	4	0	3	0	1
BN b)	0	3	0	2	2	2	2	0	5	0	5
c)	63.60	57.28	67.72	59.28	56.96	59.56	58.52	61.88	57.00	61.80	57.36
a)	0	5	0	0	4	0	2	0	3	0	3
UR b)	0	1	0	0	3	1	1	1	6	1	5
c)	82.56	74.36	89.92	78.92	74.12	77.88	77.04	80.36	73.76	79.64	73.28

a) Row representing number of win(s).
b) Row representing number of draw(s).
c) Row representing average colors used in the 25 random composite graphs.

Table IV. RESULTS OF EXPERIMENT IN COLORING RANDOM COMPOSITE GRAPHS.

(Vertices = 100, Edge Density = 30%)

	LF1	LF1I	LF2	LF2I	LF3I	DS1I	DS2I	MLF1	MLF1I	MLF2	MLF2I
a)	0	0	0	0	0	0	0	0	2	0	0
TP b)	0	10	0	0	12	0	2	3	19	3	20
c)	22.96	21.08	26.28	23.80	21.00	23.72	24.12	21.84	20.52	21.80	20.52
a)	0	7	0	0	2	0	0	0	3	0	5
UF b)	0	3	0	0	3	0	0	0	7	0	5
c)	85.44	78.36	99.56	88.04	79.60	89.56	87.00	83.92	78.36	84.48	78.08
a)	0	5	0	0	2	0	0	1	2	0	1
DR b)	0	5	0	0	7	0	0	0	8	0	11
c)	60.04	54.48	70.00	60.84	54.48	62.76	61.40	58.48	54.00	58.56	53.76
a)	0	4	0	0	3	2	1	0	4	0	1
BN b)	0	1	0	0	1	4	1	0	9	0	7
c)	84.08	77.92	92.52	80.88	77.80	78.04	80.76	81.92	76.40	81.72	77.00
a)	0	4	0	0	5	0	0	0	8	0	1
UR b)	0	2	0	0	1	0	0	1	5	0	5
c)	105.96	97.28	119.12	103.40	98.36	103.76	102.64	104.84	94.96	103.44	97.04

a) Row representing number of win(s).
b) Row representing number of draw(s).
c) Row representing average colors used in the 25 random composite graphs.

Table V. RESULTS OF EXPERIMENT IN COLORING RANDOM COMPOSITE GRAPHS.

(Vertices = 100, Edge Density = 40%)

	LF1	LF1I	LF2	LF2I	LF3I	DS1I	DS2I	MLF1	MLF1I	MLF2	MLF2I
a)	0	0	0	0	0	0	0	0	3	0	7
TP b)	0	6	0	0	3	0	0	5	13	3	14
c)	27.80	25.68	32.16	29.36	26.36	30.00	29.72	26.12	25.12	26.16	24.84
a)	0	7	0	0	7	0	0	0	7	0	0
UF b)	1	1	0	0	1	0	0	0	4	0	2
c)	106.20	98.80	123.56	108.76	99.76	110.84	109.24	105.12	98.00	104.92	99.36
a)	1	5	0	0	6	0	1	0	5	0	3
DR b)	0	1	0	1	2	0	0	0	2	0	3
c)	74.72	68.96	86.92	78.48	69.24	77 84	79.16	73.76	68.92	74.00	68.80
a)	0	3	0	2	3	0	0	0	2	0	4
BN b)	0	4	0	0	3	0	1	0	7	0	10
c)	102.16	95.48	113.36	98.92	96.52	101.32	98.28	100.04	94.80	100.88	94.48
a)	0	8	0	0	4	0	0	0	5	0	2
UR b)	0	2	0	0	1	0	0	0	5	1	5
c)	133.00	122.80	148.92	131.64	125.04	132.24	130.60	130.48	122.20	130.68	122.96

a) Row representing number of win(s).
b) Row representing number of draw(s).
c) Row representing average colors used in the 25 random composite graphs.

Table VI. RESULTS OF EXPERIMENT IN COLORING RANDOM COMPOSITE GRAPHS.

(Vertices = 100, Edge Density = 50%)

	LF1	LF1I	LF2	LF2I	LF3I	DS1I	DS2I	MLF1	MLF1I	MLF2	MLF2I
a)	0	0	0	0	1	0	0	1	6	0	5
TP b)	0	7	0	0	0	0	0	3	7	1	11
c)	33.76	31.12	38.44	34.36	32.00	36.08	35.40	31.72	30.44	31.92	30.40
a)	0	8	0	0	3	0	0	0	4	0	6
UF b)	0	2	0	0	0	0	0	0	3	0	3
c)	129.44	119.44	148.68	132.76	121.36	133.20	134.96	128.32	120.28	128.52	120.16
a)	0	4	0	0	1	0	0	0	6	0	4
DR b)	0	5	0	0	2	0	0	1	8	0	8
c)	88.96	82.64	104.28	94.52	84.44	95.60	94.28	88.28	81.44	88.56	82.44
a)	0	8	0	0	6	1	0	0	2	1	3
BN b)	0	1	0	0	0	0	0	1	2	1	3
c)	124.32	116.04	137.64	122.28	117.60	122.04	122.32	120.56	116.92	120.80	116.36
a)	0	6	0	0	5	0	0	0	4	1	8
UR b)	0	1	0	0	0	0	0	0	1	0	1
c)	161.28	150.76	180.12	161.48	150.84	160.88	160.80	155.80	150.76	157.52	149.00

a) Row representing number of win(s).
b) Row representing number of draw(s).
c) Row representing average colors used in the 25 random composite graphs.

Table VII. RESULTS OF EXPERIMENT IN COLORING RANDOM COMPOSITE GRAPHS.

(Vertices = 200, Edge Density = 10%)

	LF1	LF1I	LF2	LF2I	LF3I	DS1I	DS2I	MLF1	MLF1I	MLF2	MLF2I
a)	0	1	0	0	0	0	0	0	3	0	4
TP b)	0	9	0	0	6	0	0	2	15	0	16
c)	17.16	15.68	20.60	18.52	15.92	18.88	18.16	16.44	15.36	16.60	15.28
a)	0	4	0	0	5	0	0	0	1	0	5
UF b)	0	4	0	0	3	0	0	0	7	0	7
c)	67.04	59.96	76.44	66.12	59.56	65.48	65.56	66.08	59.92	66.04	59.88
a)	0	2	0	0	3	0	0	0	2	0	3
DR b)	1	6	0	0	4	0	0	0	12	0	13
c)	46.80	43.20	56.24	48.20	43.56	49.44	48.60	47.24	42.80	47.32	42.36
a)	0	5	0	0	0	1	2	0	4	0	5
BN b)	0	3	0	0	1	3	2	0	7	0	6
c)	64.40	57.56	69.88	61.00	60.20	59.12	59.04	63.40	57.32	62.76	57.40
a)	0	6	0	0	2	0	3	0	4	0	2
UR b)	0	6	0	0	4	0	0	0	4	0	4
c)	82.04	74.68	92.24	81.32	76.12	78.72	77.40	80.76	74.48	80.04	74.56

a) Row representing number of win(s).
b) Row representing number of draw(s).
c) Row representing average colors used in the 25 random composite graphs.

Table VIII. RESULTS OF EXPERIMENT IN COLORING RANDOM COMPOSITE GRAPHS.

(Vertices = 200, Edge Density = 20%)

	LF1	LF1I	LF2	LF2I	LF3I	DS1I	DS2I	MLF1	MLF1I	MLF2	MLF2I
a)	0	0	0	0	0	0	0	1	3	0	3
TP b)	0	6	0	0	5	0	0	3	18	1	15
c)	25.96	23.92	31.40	28.40	24.20	28.72	29.28	24.16	23.12	24.36	23.24
a)	0	8	0	0	3	0	0	0	4	0	7
UF b)	0	2	0	0	0	0	0	0	1	0	3
c)	102.92	93.28	120.08	107.32	94.32	106.56	106.88	99.60	93.52	99.56	92.92
a)	0	3	0	0	4	0	0	0	3	0	2
DR b)	0	5	0	0	3	0	0	0	11	0	11
c)	71.88	65.36	86.36	75.04	66.16	77.20	76.36	69.68	64.72	70.00	64.80
a)	0	3	0	0	3	0	1	0	2	0	5
BN b)	0	3	0	0	1	0	1	1	7	0	9
c)	99.64	92.48	110.92	97.88	94.08	97.56	95.88	96.40	91.64	96.00	91.00
a)	0	7	0	0	3	0	0	0	7	0	7
UR b)	0	1	0	0	0	0	0	0	1	0	1
c)	128.28	118.28	149.04	131.24	120.48	129.80	129.00	126.20	117.32	125.88	117.52

a) Row representing number of win(s).
b) Row representing number of draw(s).
c) Row representing average colors used in the 25 random composite graphs.

Table IX. RESULTS OF EXPERIMENT IN COLORING RANDOM COMPOSITE GRAPHS.

(Vertices = 200, Edge Density = 30%)

	LF1	LF1I	LF2	LF2I	LF3I	DS1I	DS2I	MLF1	MLF1I	MLF2	MLF2I
a)	0	0	0	0	0	0	0	1	5	0	5
TP b)	0	0	0	0	4	0	0	6	14	4	13
c)	35.20	32.80	41.88	37.52	32.56	39.36	38.80	32.72	31.52	32.44	31.56
a)	0	0	0	0	3	0	0	0	7	0	7
UF b)	0	2	0	0	5	0	0	0	4	0	5
c)	135.00	126.60	160.92	144.28	126.16	143.56	144.00	130.76	124.72	132.32	125.28
a)	0	6	0	0	2	0	0	0	5	0	7
DR b)	0	3	0	0	1	0	0	0	3	0	3
c)	95.84	88.44	144.12	102.60	90.08	104.64	105.80	94.96	88.32	94.60	87.64
a)	0	6	0	0	1	0	1	0	5	1	5
BN b)	0	2	0	0	0	0	2	1	6	1	3
c)	133.32	124.68	150.36	135.12	125.88	133.52	133.08	127.88	123.64	127.64	124.32
a)	0	6	0	0	4	0	0	0	4	0	6
UR b)	0	2	0	0	1	0	0	0	4	0	3
c)	173.96	160.28	198.64	180.88	162.20	177.80	176.52	168.72	160.36	167.36	160.60

a) Row representing number of win(s).
b) Row representing number of draw(s).
c) Row representing average colors used in the 25 random composite graphs.

Table X. RESULTS OF EXPERIMENT IN COLORING RANDOM COMPOSITE GRAPHS.

(Vertices = 200, Edge Density = 40%)

	LF1	LF1I	LF2	LF2I	LF3I	DS1I	DS2I	MLF1	MLF1I	MLF2	MLF2I
a)	0	0	0	0	0	0	0	0	1	0	4
TP b)	0	5	0	0	4	0	0	3	15	6	18
c)	44.48	41.72	52.52	47.56	42.12	49.32	48.04	41.88	40.72	41.44	40.36
a)	0	7	0	0	2	0	0	0	6	0	8
UF b)	0	0	0	0	2	0	0	0	1	0	1
c)	171.72	161.60	206.88	184.68	162.24	184.64	185.08	168.24	160.76	167.88	160.32
a)	0	4	0	0	3	0	0	0	8	0	6
DR b)	0	1	0	0	1	0	0	0	3	0	3
c)	120.24	111.60	144.68	129.28	112.56	131.72	131.40	117.20	110.44	117.48	110.60
a)	0	7	0	0	7	0	0	0	5	0	2
BN b)	0	3	0	0	1	0	0	0	2	1	3
c)	168.12	157.88	189.44	172.04	158.28	171.08	171.00	162.20	157.88	161.88	159.48
a)	0	3	0	0	3	0	0	0	6	0	11
UR b)	0	0	0	0	2	0	0	0	1	0	1
c)	215.96	203.64	253.60	227.04	202.16	227.68	223.60	210.20	202.88	211.48	201.32

a) Row representing number of win(s).
b) Row representing number of draw(s).
c) Row representing average colors used in the 25 random composite graphs.

Table XI. RESULTS OF EXPERIMENT IN COLORING RANDOM COMPOSITE GRAPHS.

(Vertices = 200, Edge Density = 50%)

	LF1	LF1I	LF2	LF2I	LF3I	DS1I	DS2I	MLF1	MLF1I	MLF2	MLF2I
a)	0	1	0	0	0	0	0	0	7	0	8
TP b)	0	1	0	0	0	0	0	2	7	3	8
c)	54.84	51.24	64.92	58.56	52.44	59.64	59.92	51.72	50.12	51.36	50.08
a)	0	4	0	0	2	0	0	0	9	0	8
UF b)	0	1	0	0	0	0	0	0	2	0	1
c)	210.64	200.32	249.52	226.52	200.08	227.48	227.76	206.68	197.40	207.00	197.96
a)	0	5	0	0	3	0	0	0	7	0	5
DR b)	0	2	0	0	0	0	0	0	4	0	4
c)	145.88	135.56	174.64	158.56	137.60	163.68	160.28	142.60	135.32	143.68	135.20
a)	0	7	0	0	1	0	0	2	5	0	6
BN b)	0	3	0	0	1	0	0	0	1	1	2
c)	205.00	194.08	235.72	210.76	197.56	209.80	209.48	199.80	194.08	200.04	195.68
a)	0	3	0	0	5	0	0	1	6	0	9
UR b)	0	0	0	0	0	0	0	0	1	0	1
c)	264.76	250.84	310.88	274.32	250.12	277.72	275.84	257.88	248.88	259.00	248.32

a) Row representing number of win(s).
b) Row representing number of draw(s).
c) Row representing average colors used in the 25 random composite graphs.

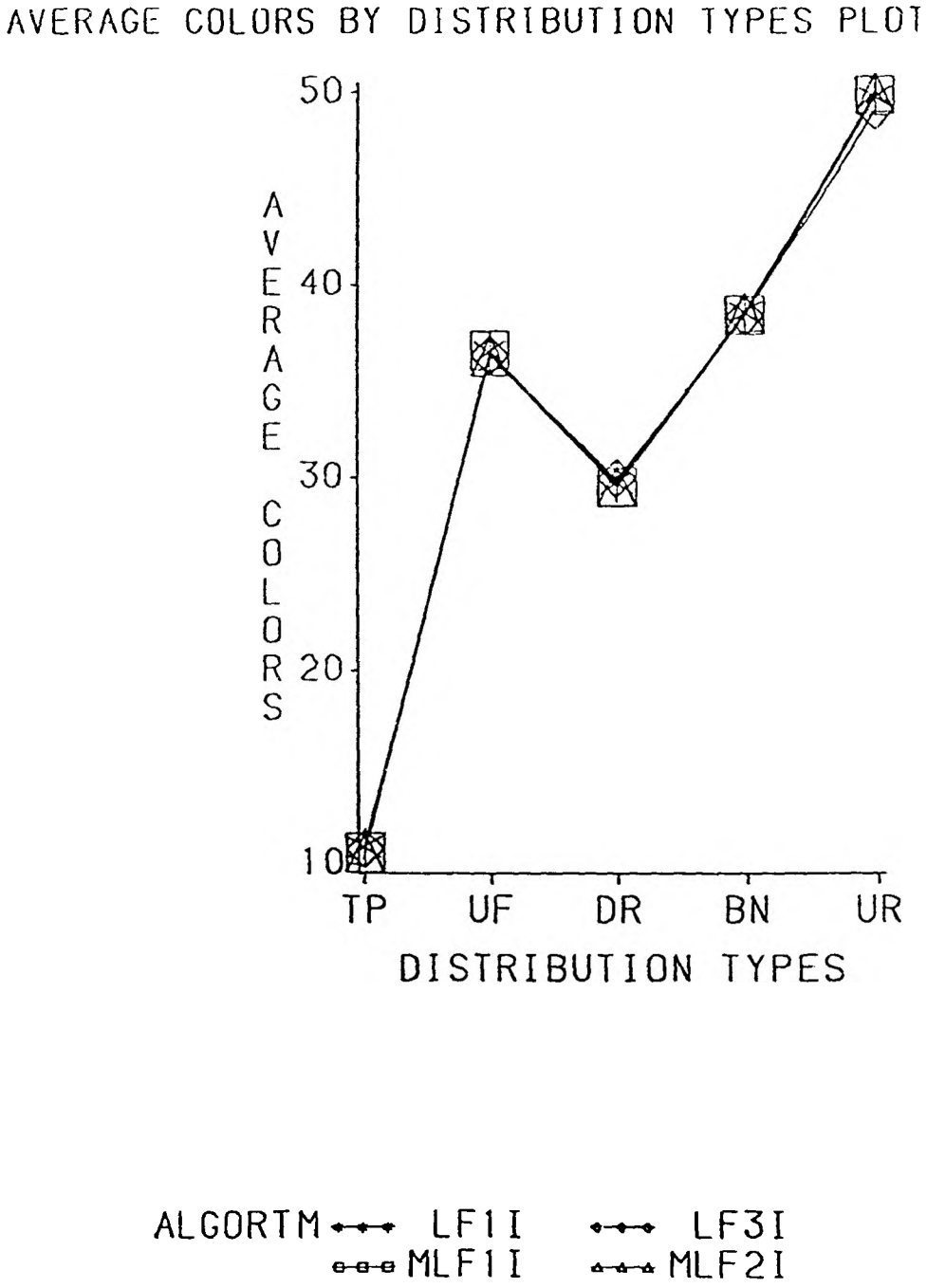

Figure 11. Graph with vertices = 100 and edge density = 10%.

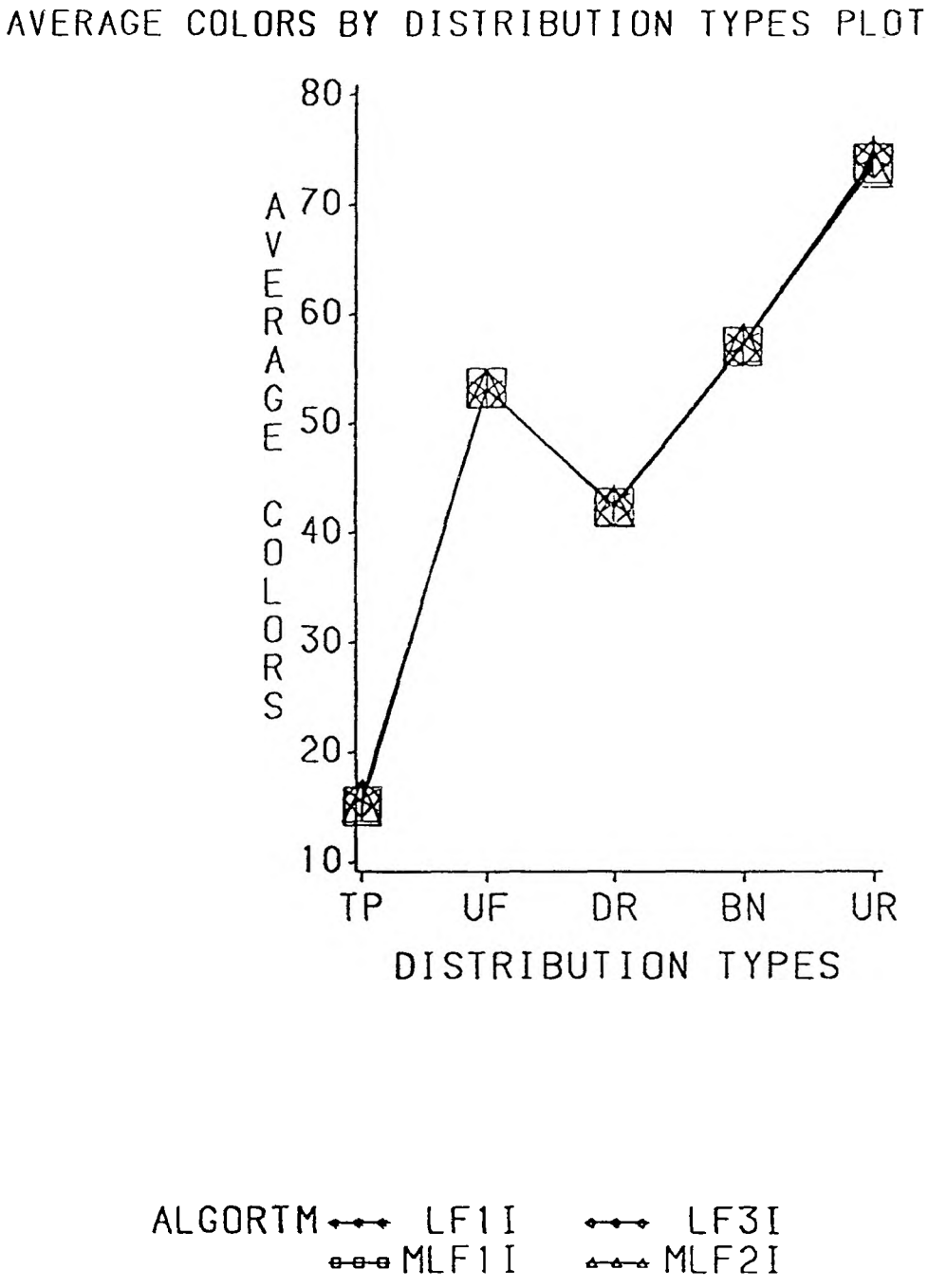

Figure 12. Graph with vertices = 100 and edge density = 20%.

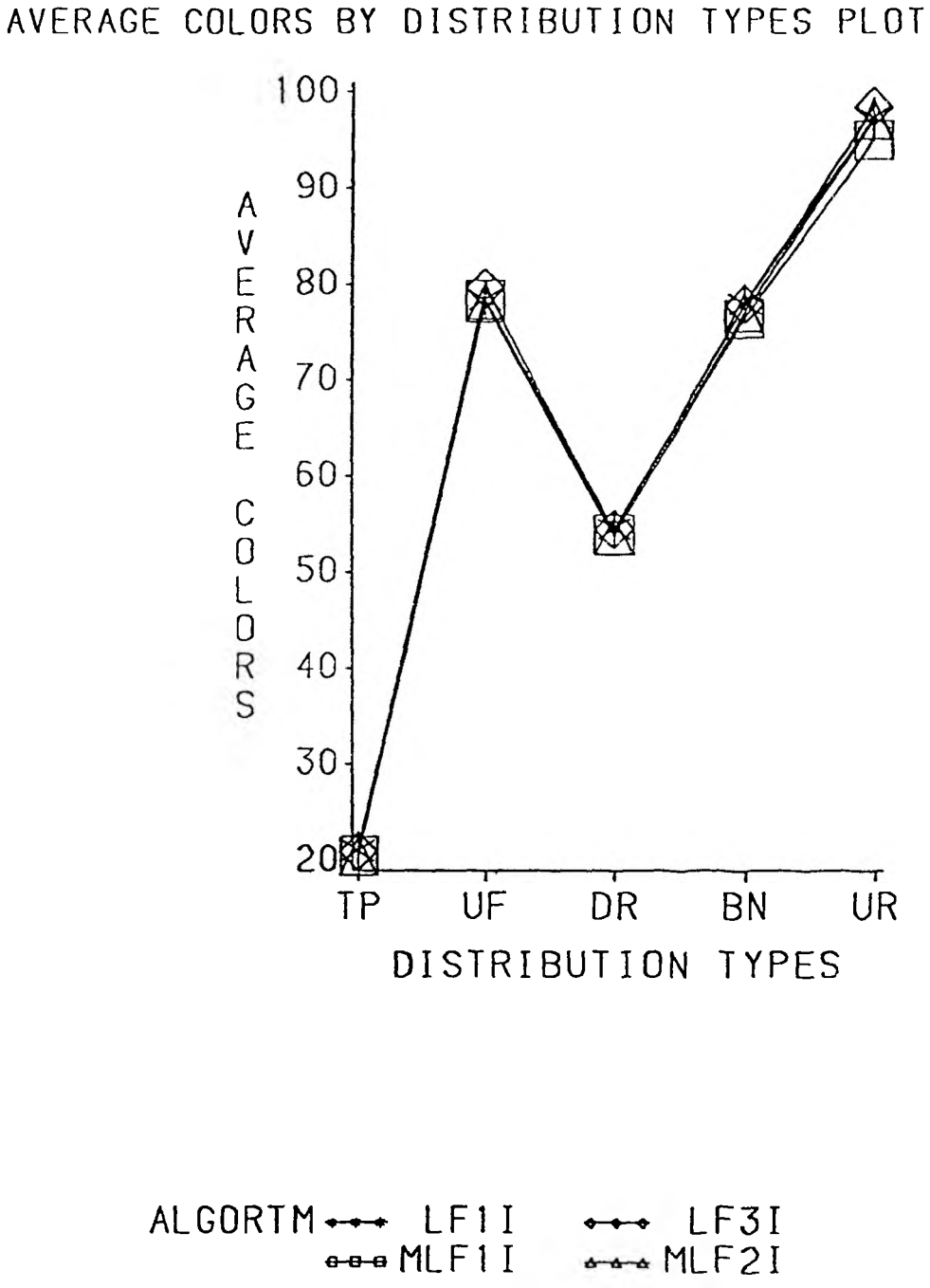

Figure 13. Graph with vertices = 100 and edge density = 30%.

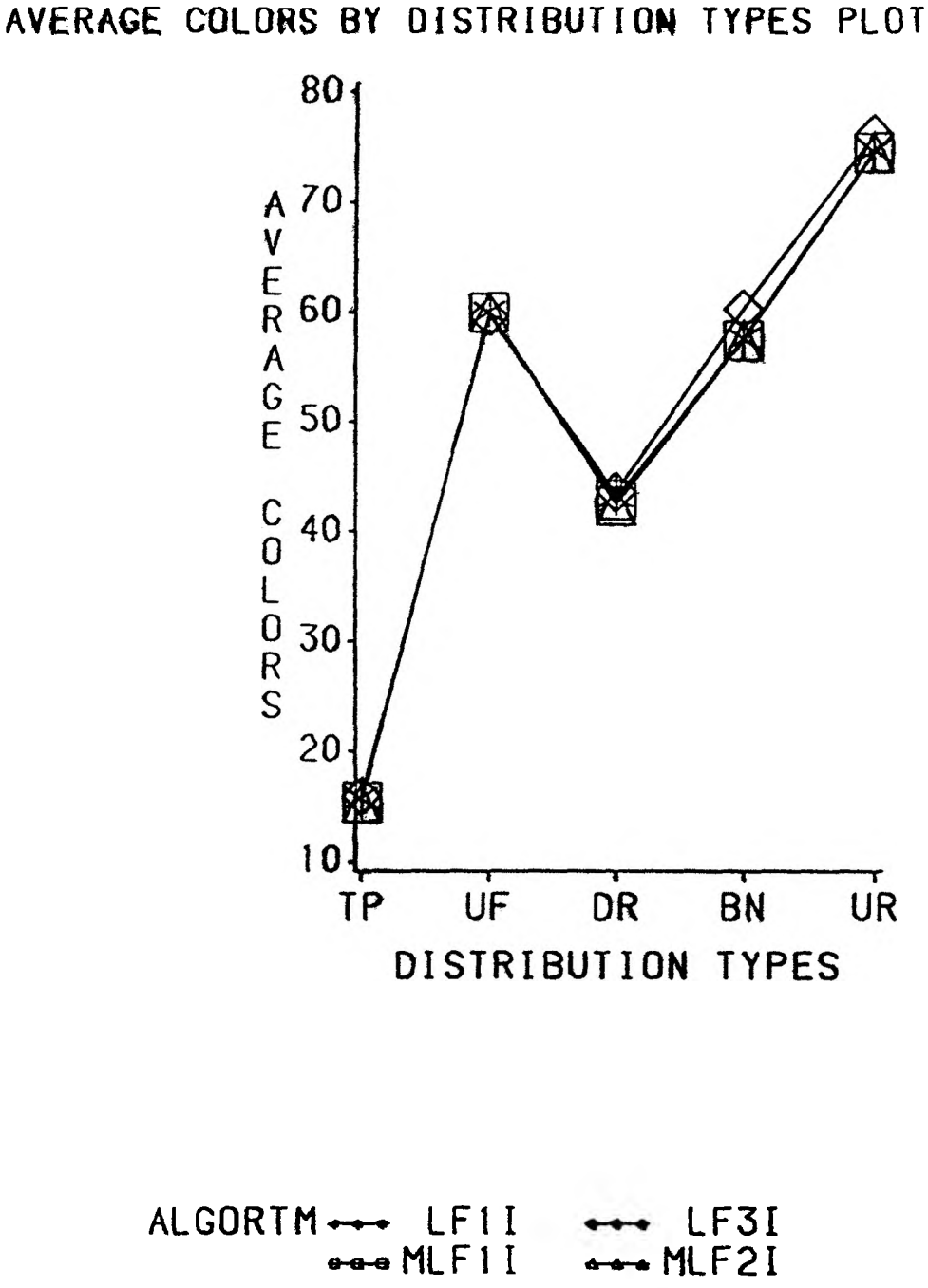

Figure 14. Graph with vertices = 200 and edge density = 10%.

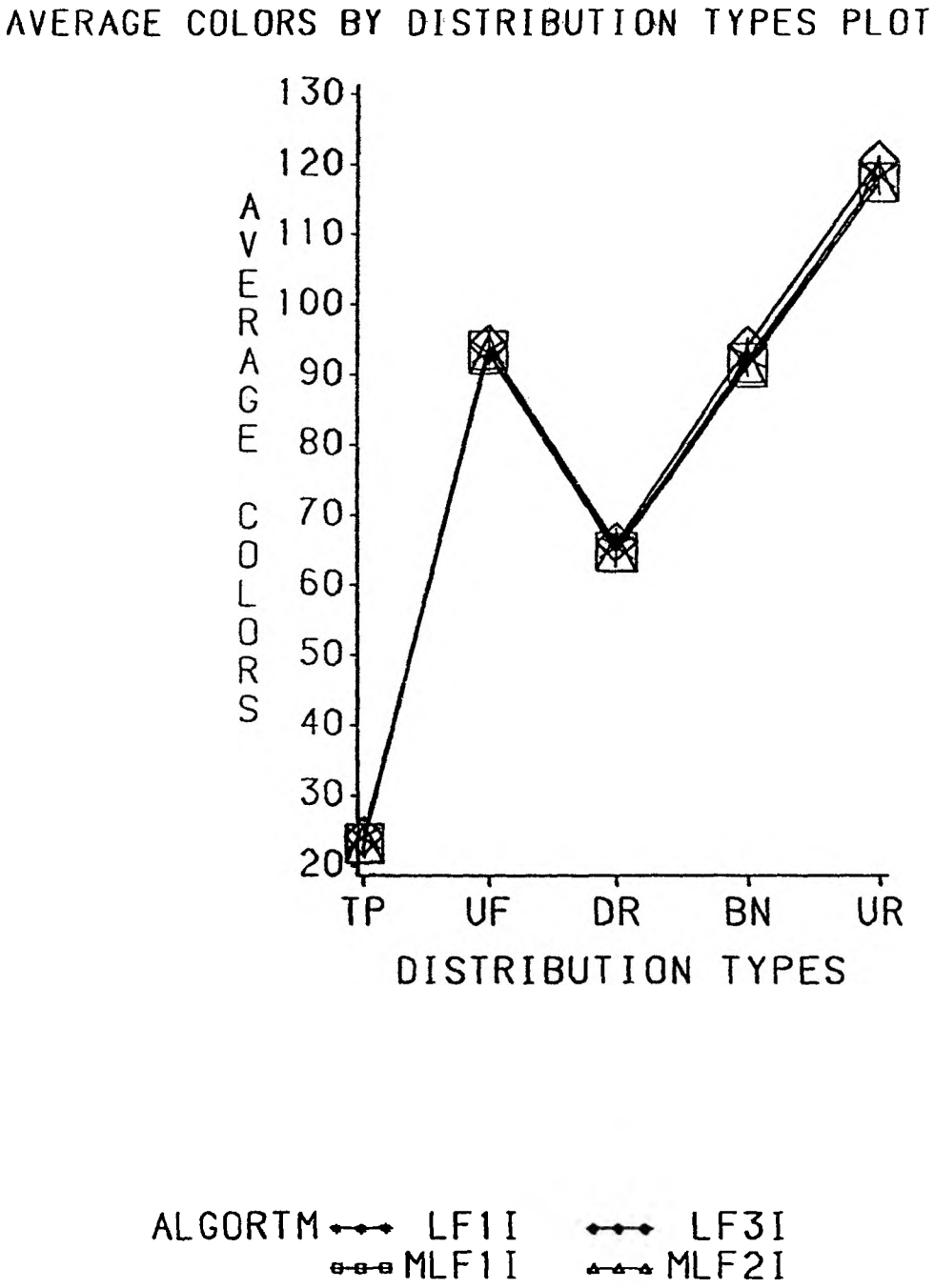

Figure 15. Graph with vertices = 200 and edge density = 20%.

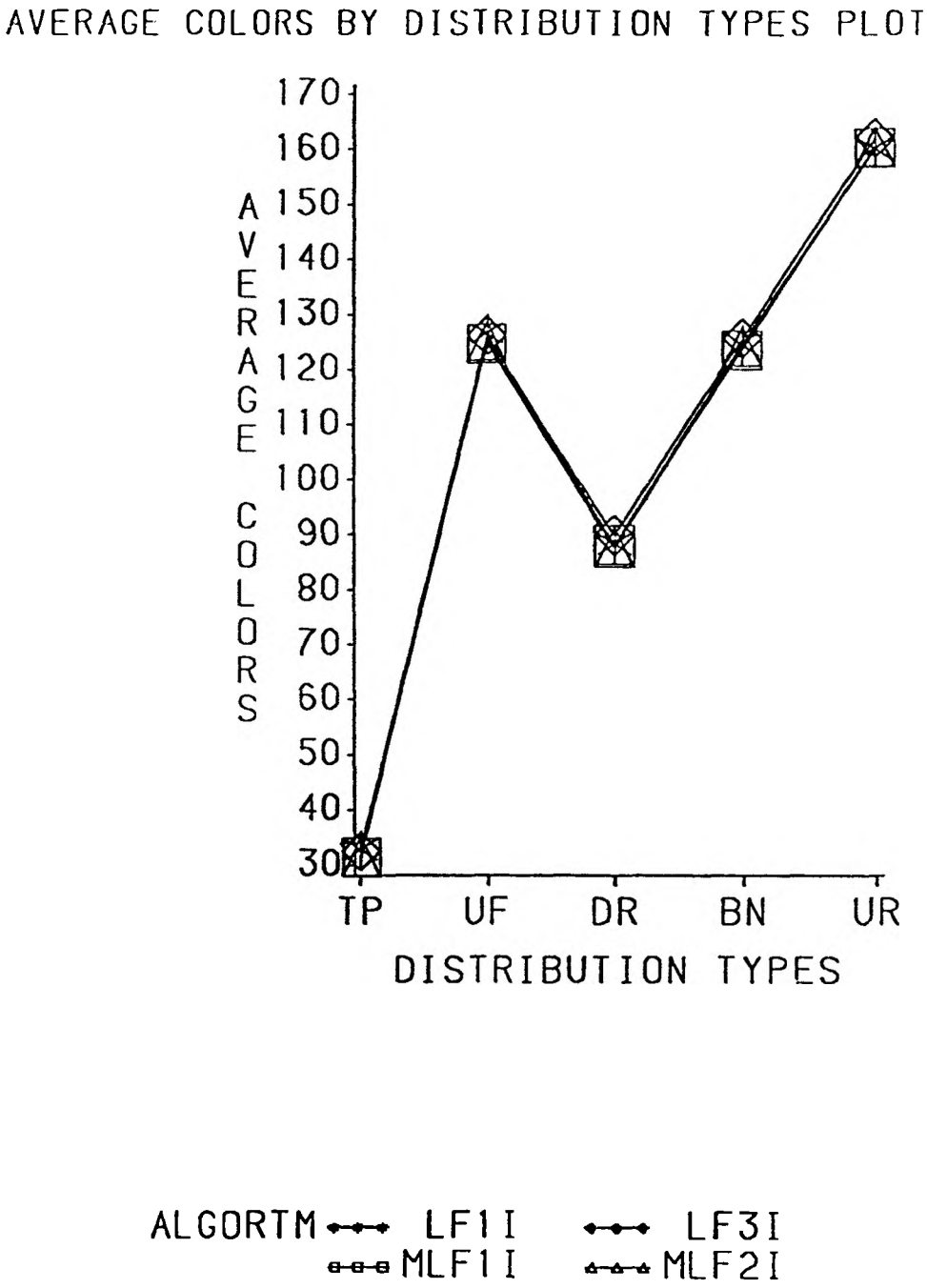

Figure 16. Graph with vertices = 200 and edge density = 30%.

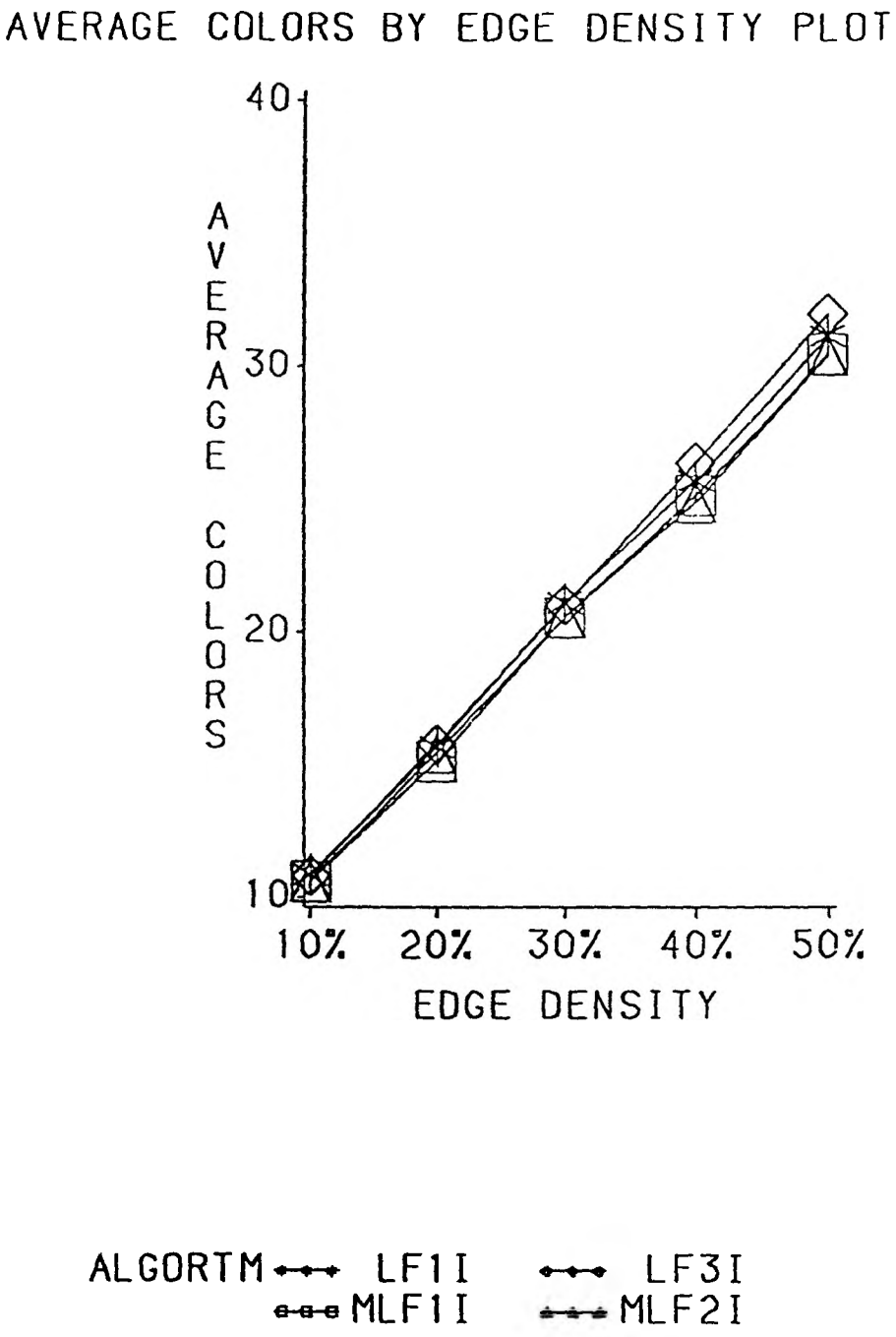

Figure 17. Graph with vertices = 100 and TP distribution type.

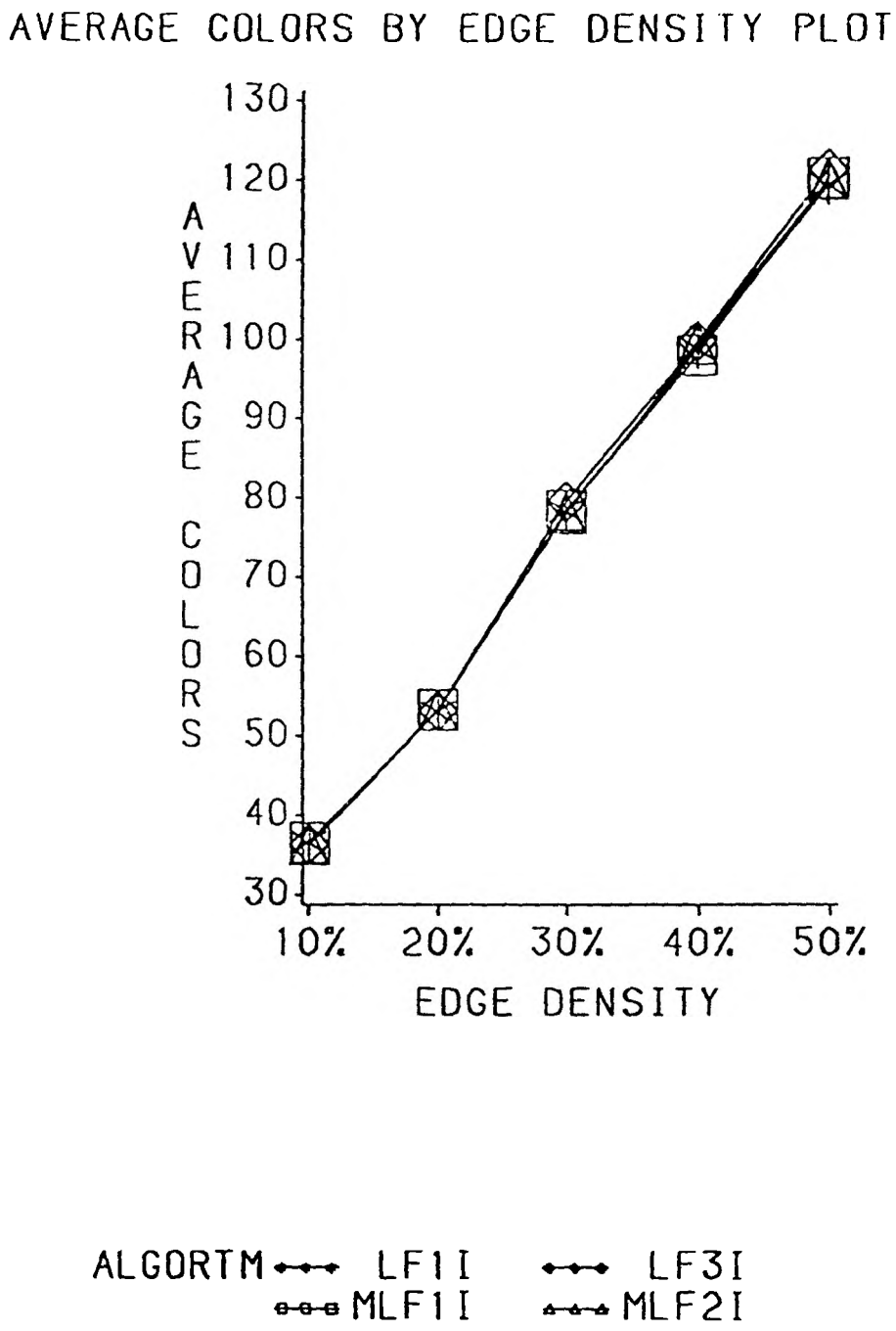

Figure 18. Graph with vertices = 100 and UF distribution type.

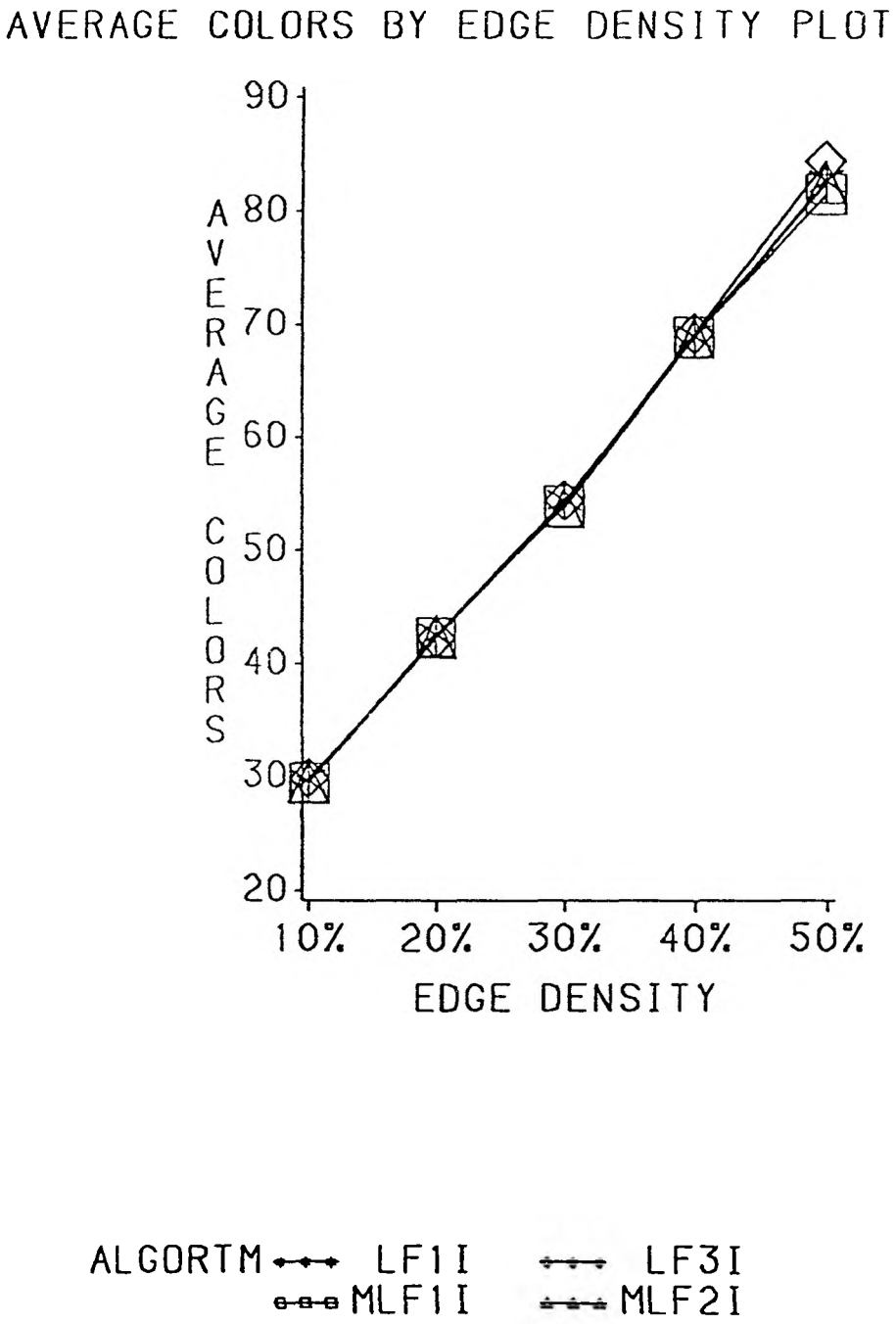

Figure 19. Graph with vertices = 100 and DR distribution type.

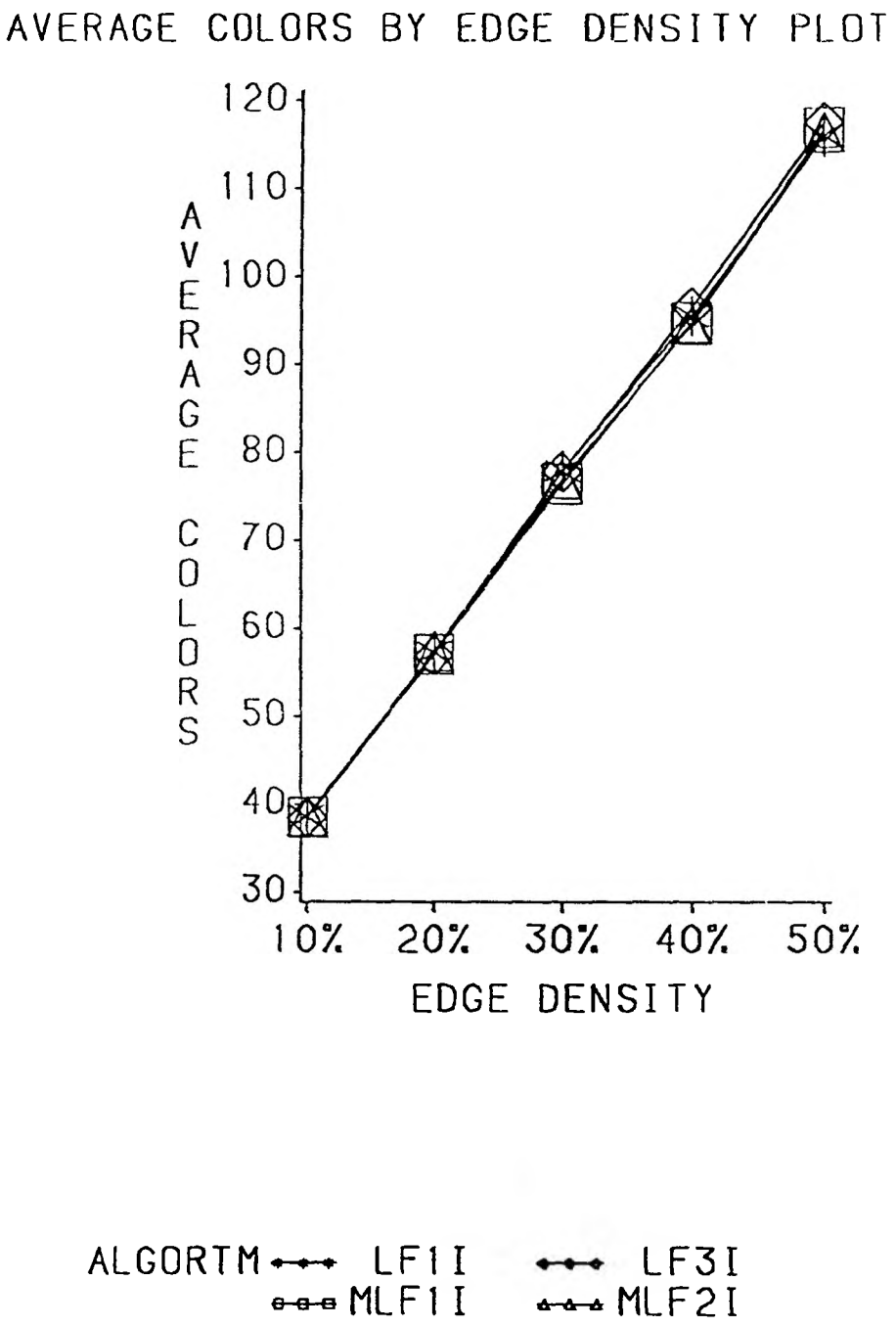

Figure 20. Graph with vertices = 100 and BN distribution type.

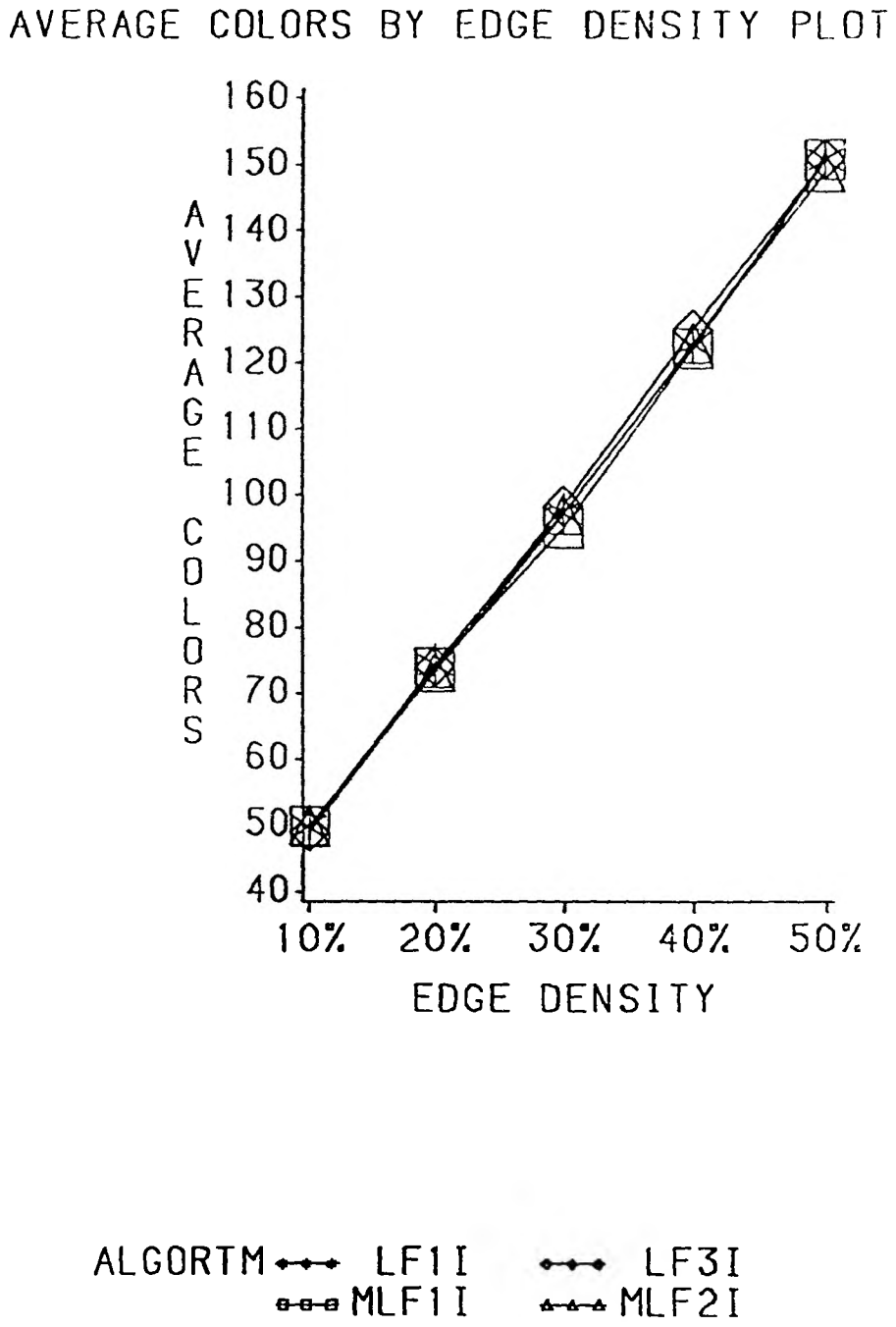

Figure 21. Graph with vertices = 100 and UR distribution type.

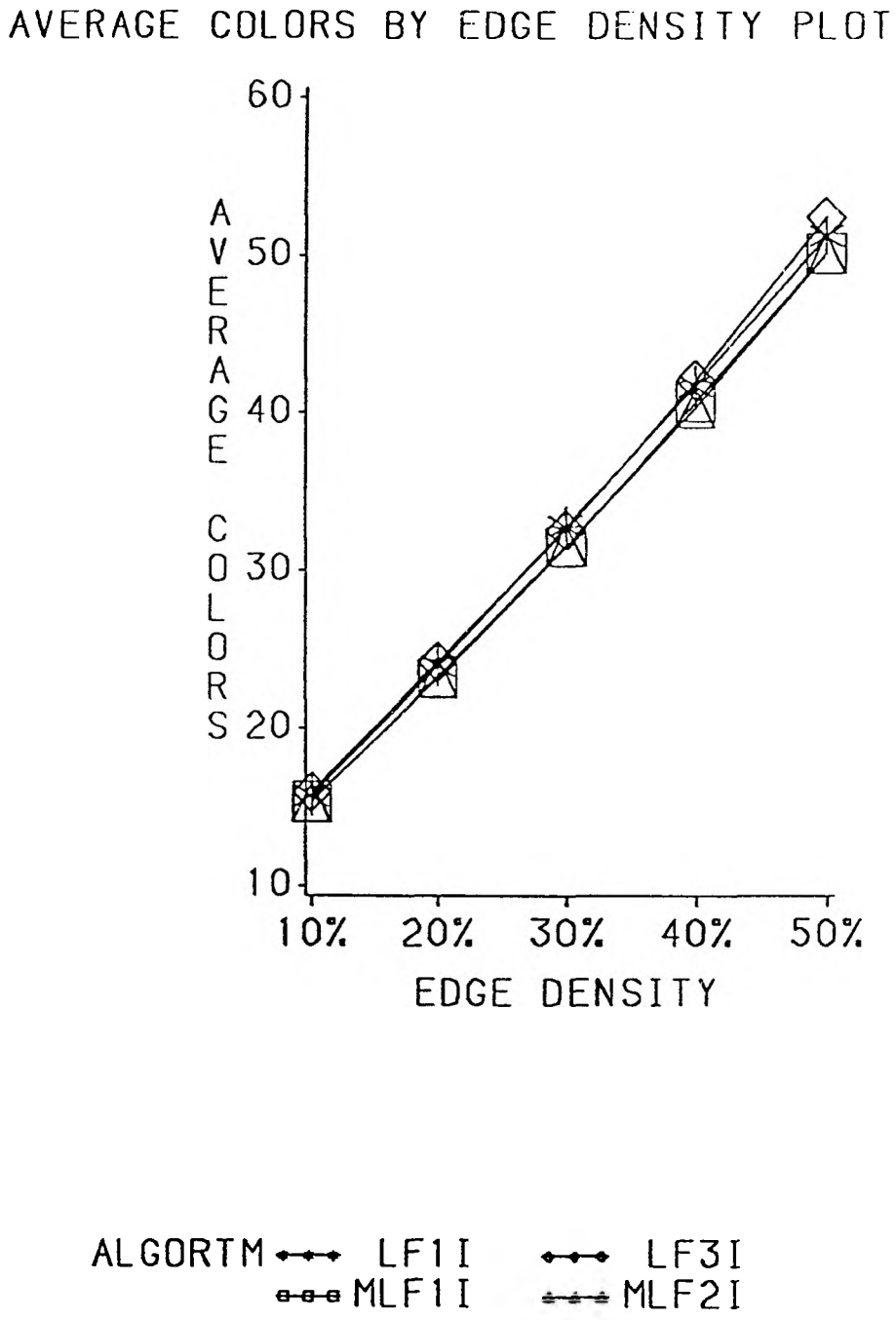

Figure 22. Graph with vertices = 200 and TP distribution type.

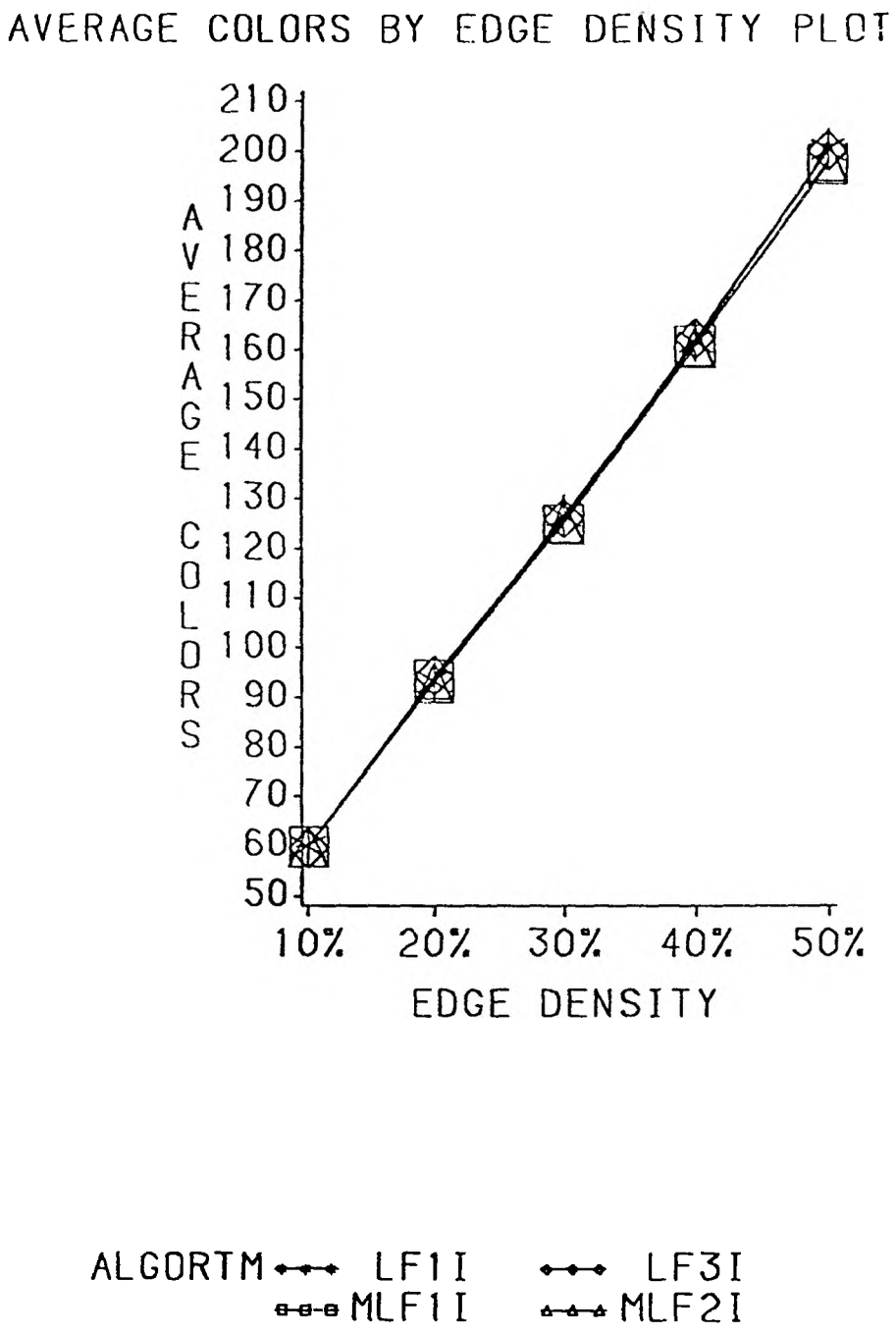

Figure 23. Graph with vertices = 200 and UF distribution type.

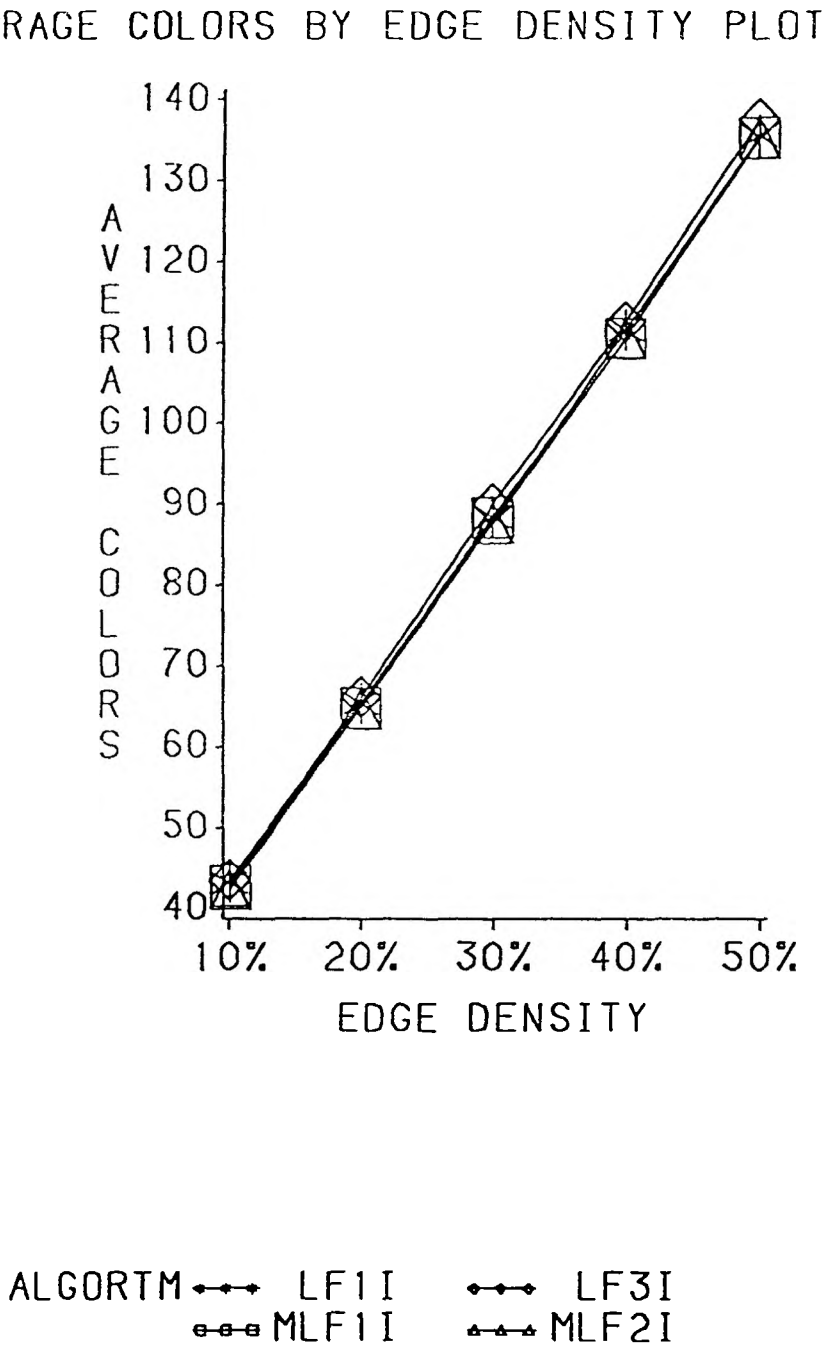

Figure 24. Graph with vertices = 200 and DR distribution type.

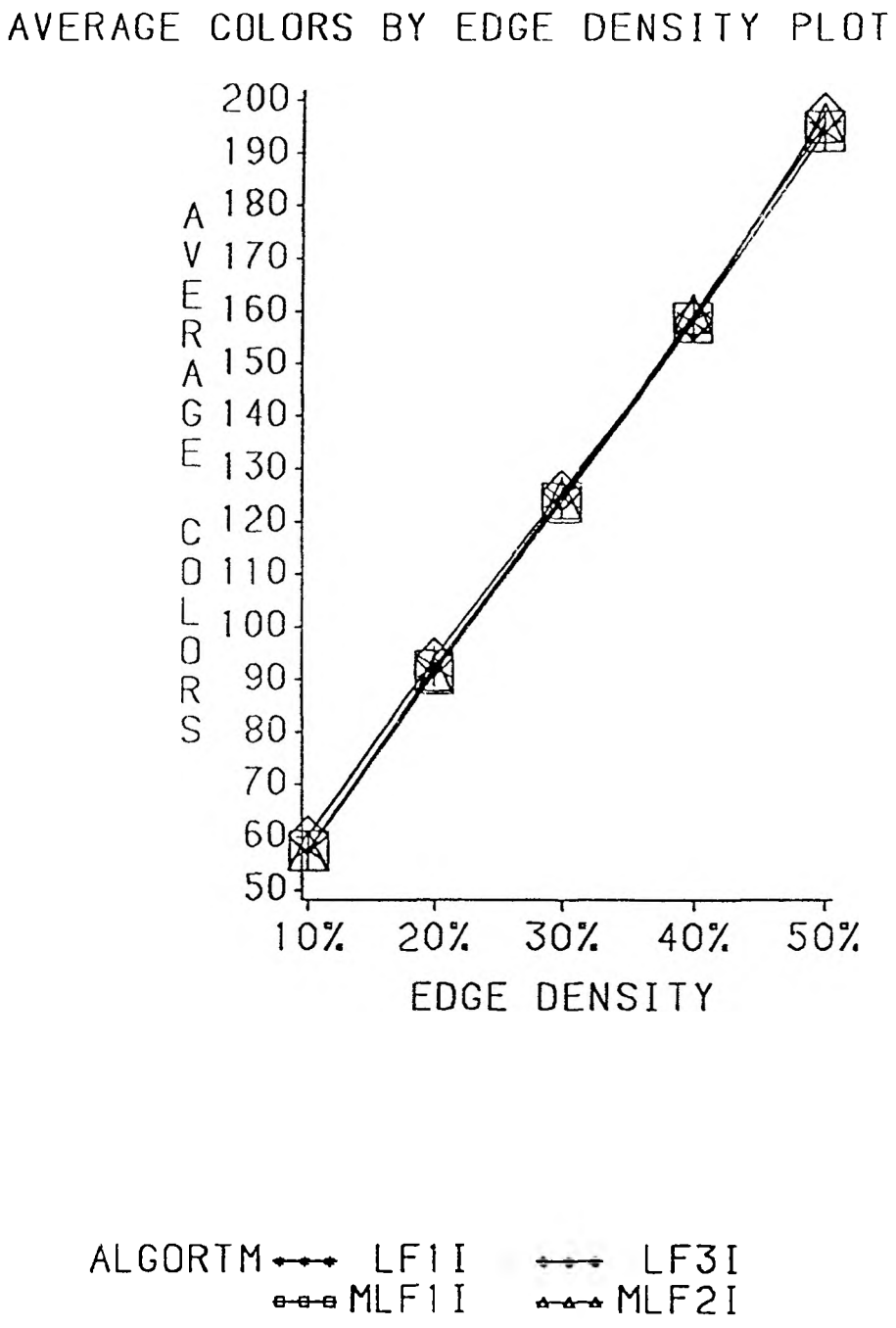

Figure 25. Graph with vertices = 200 and BN distribution type.

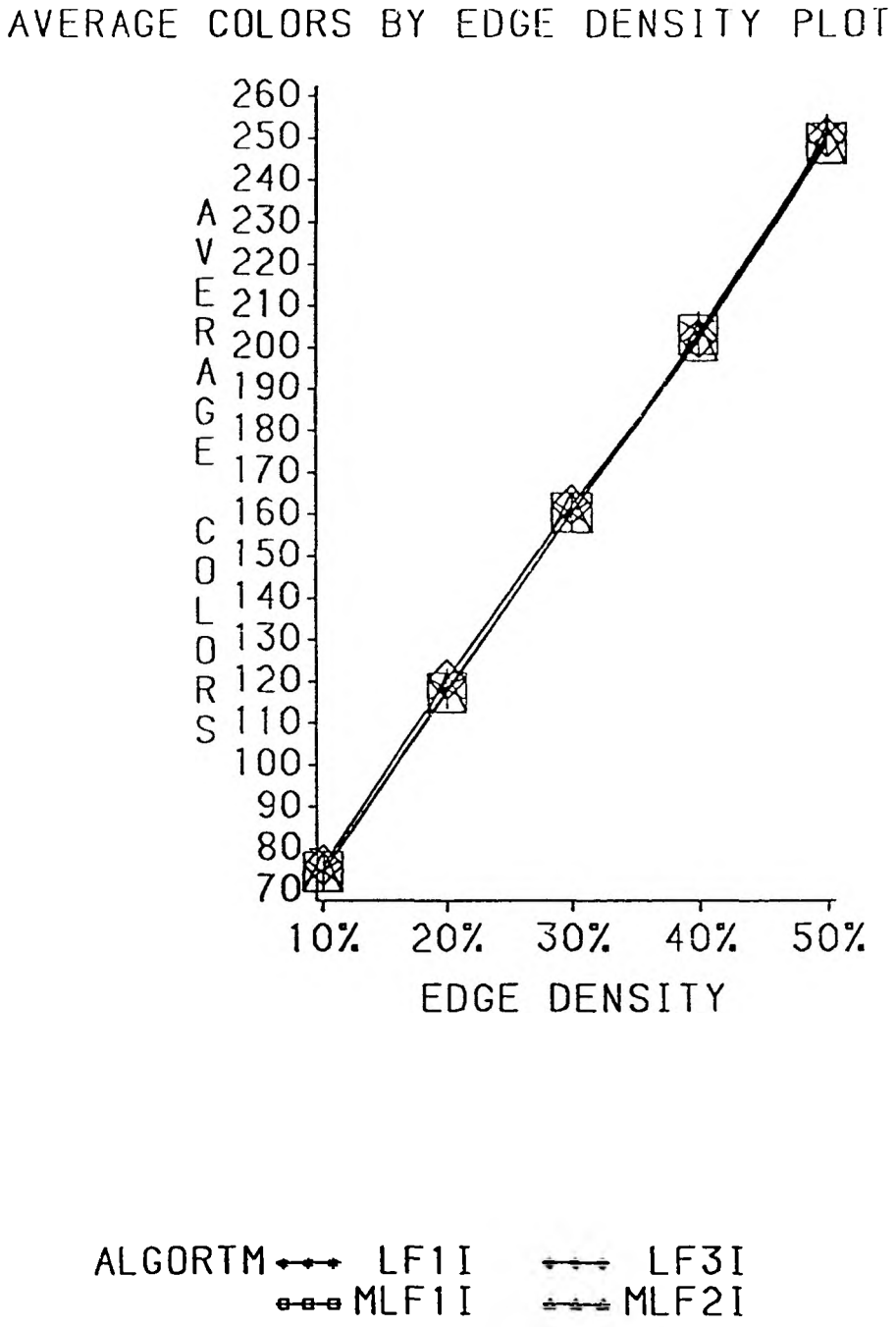

Figure 26. Graph with vertices = 200 and UR distribution type.

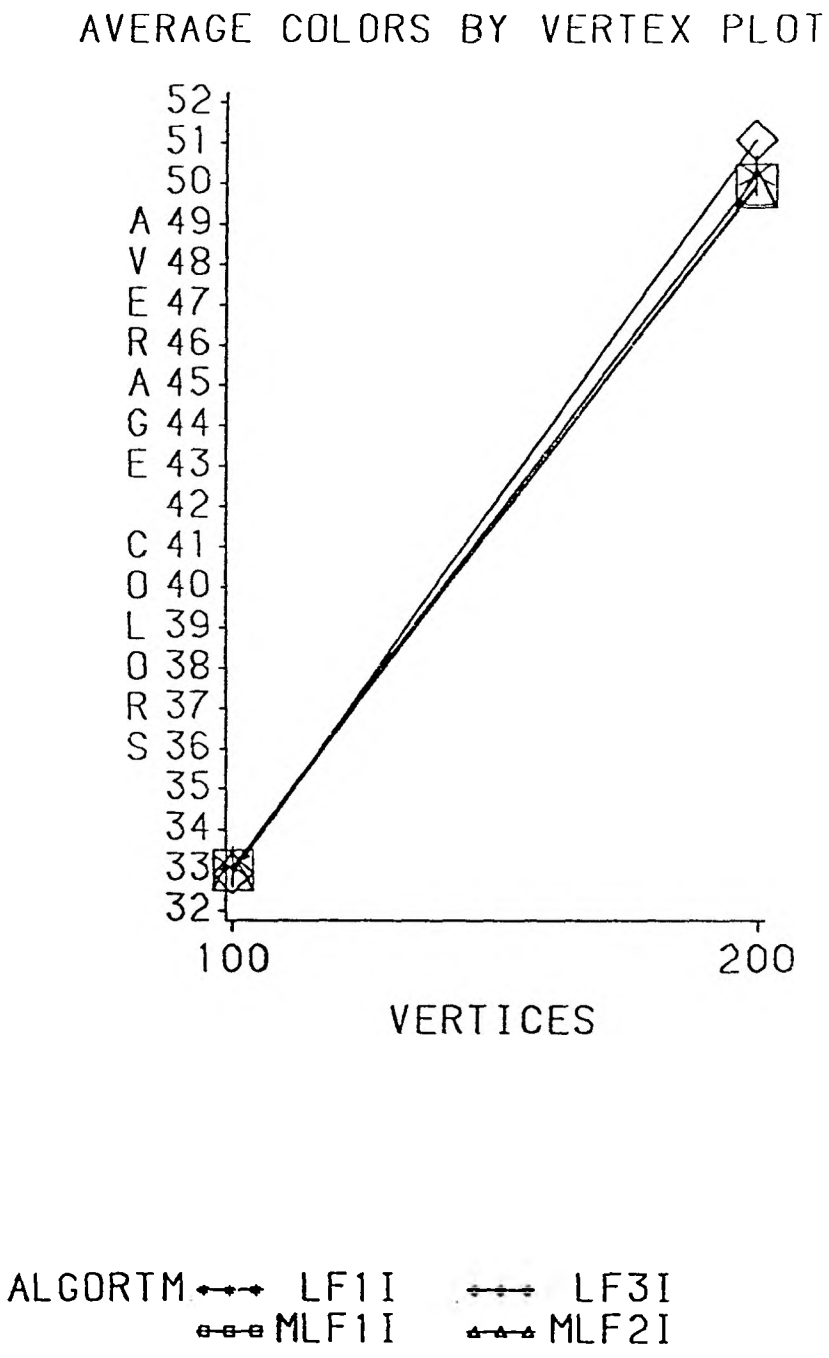

Figure 27. Graph with edge density = 0.10.

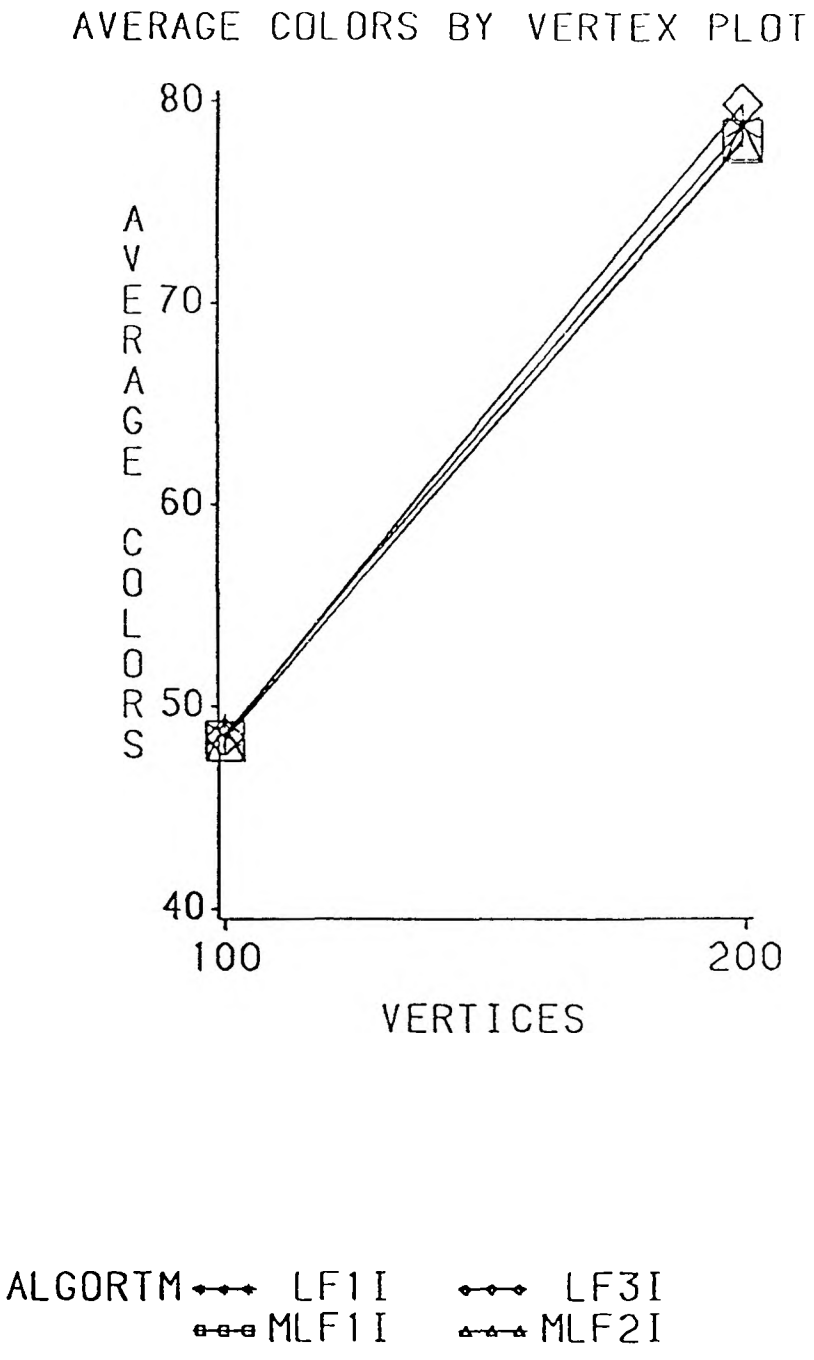

Figure 28. Graph with edge density = 0.20.

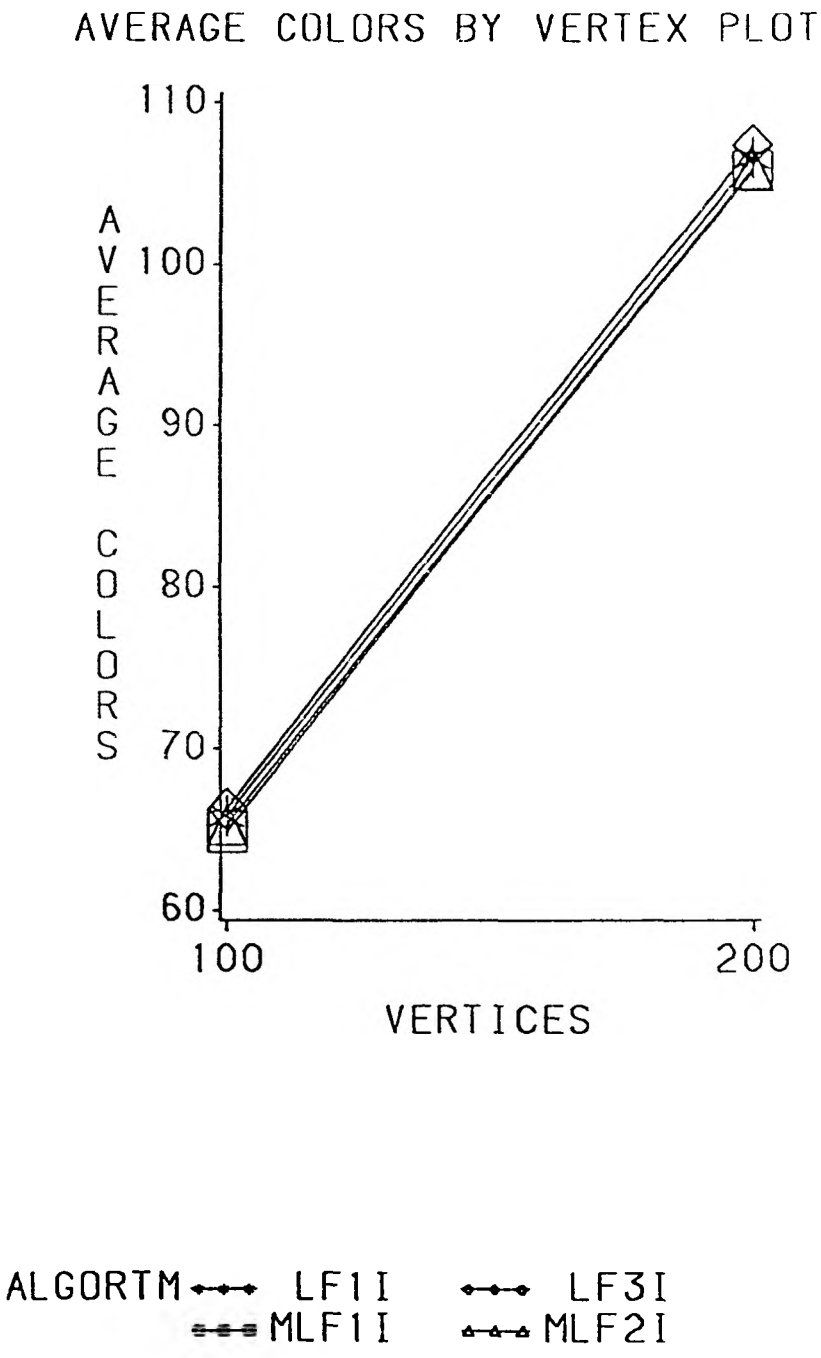

Figure 29. Graph with edge density = 0.30.

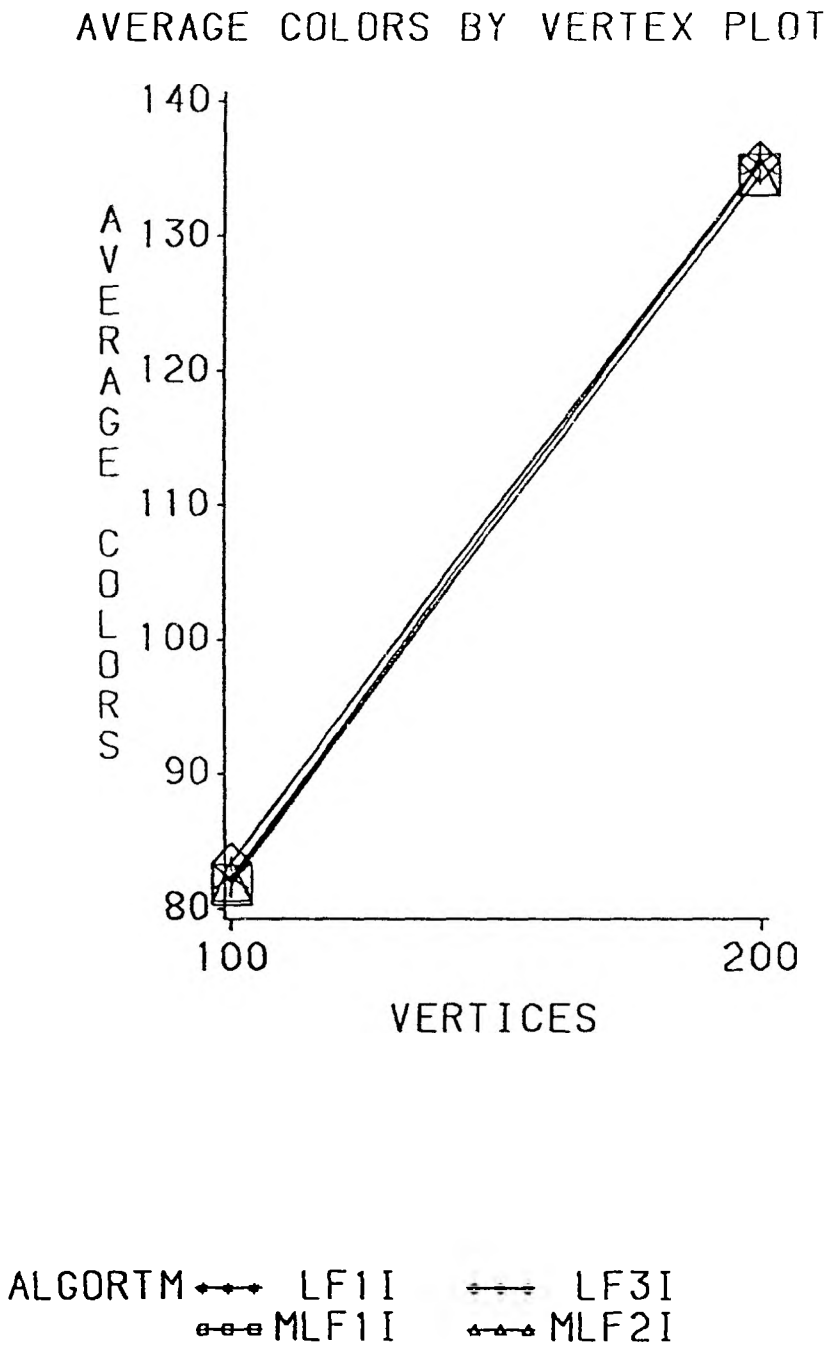

Figure 30. Graph with edge density = 0.40.

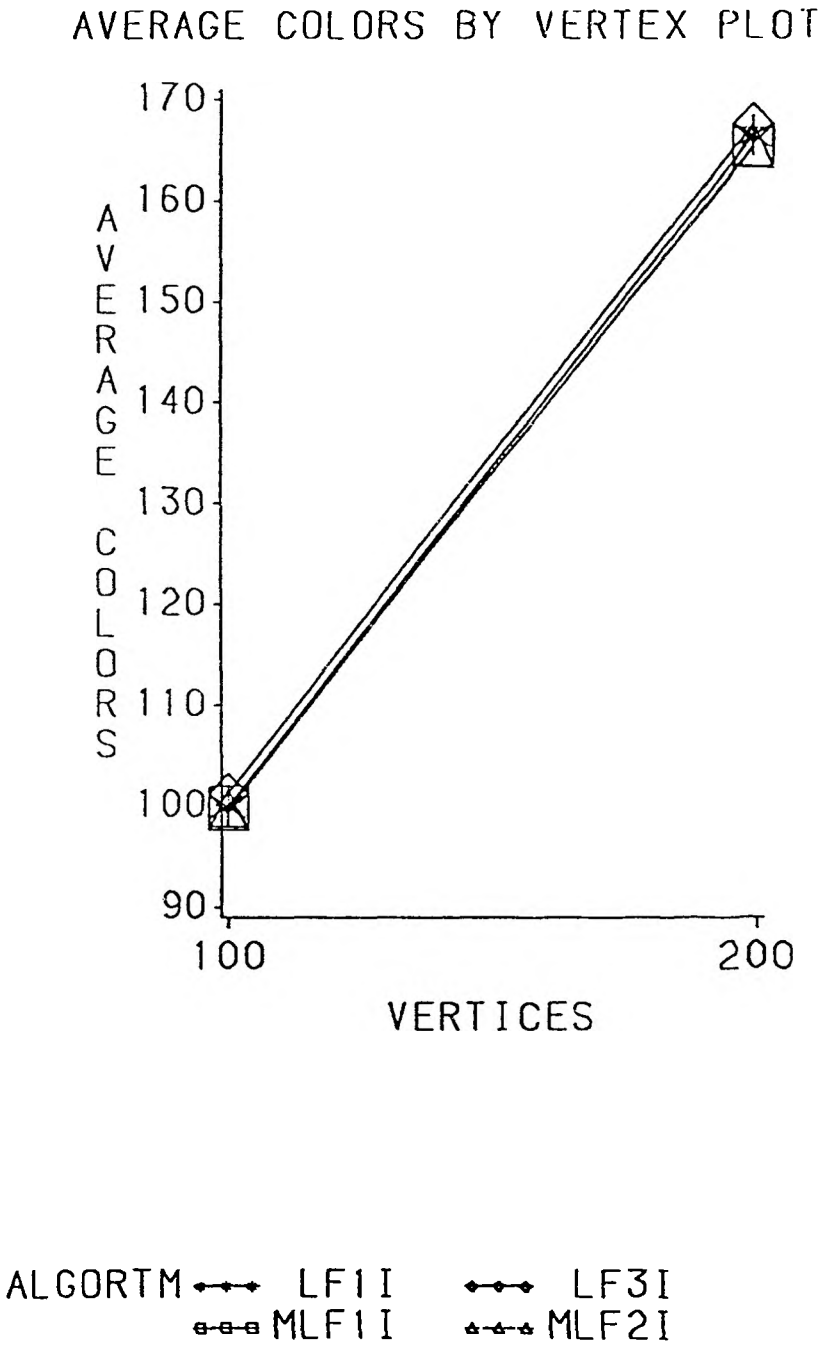

Figure 31. Graph with edge density = 0.50.

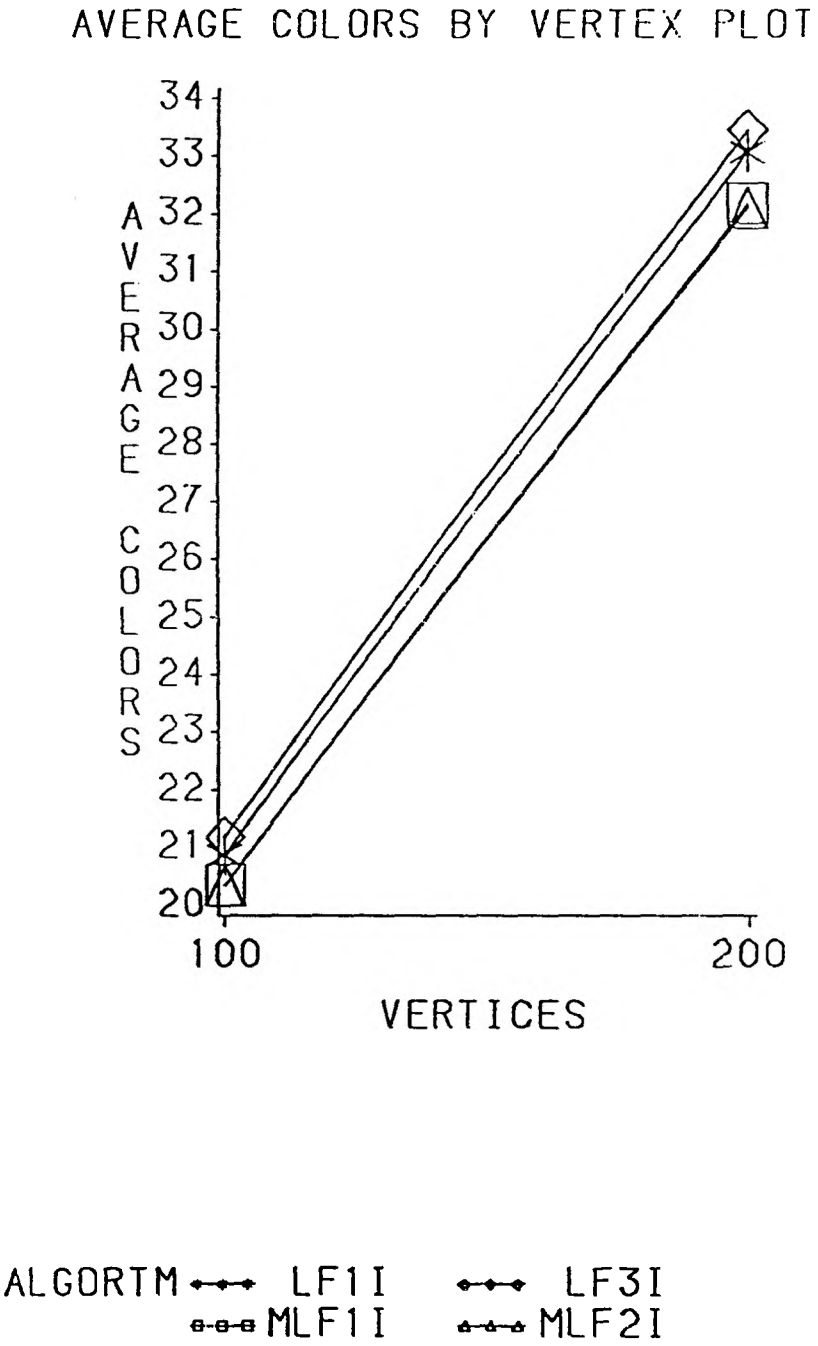

Figure 32. Graph with TP distribution.

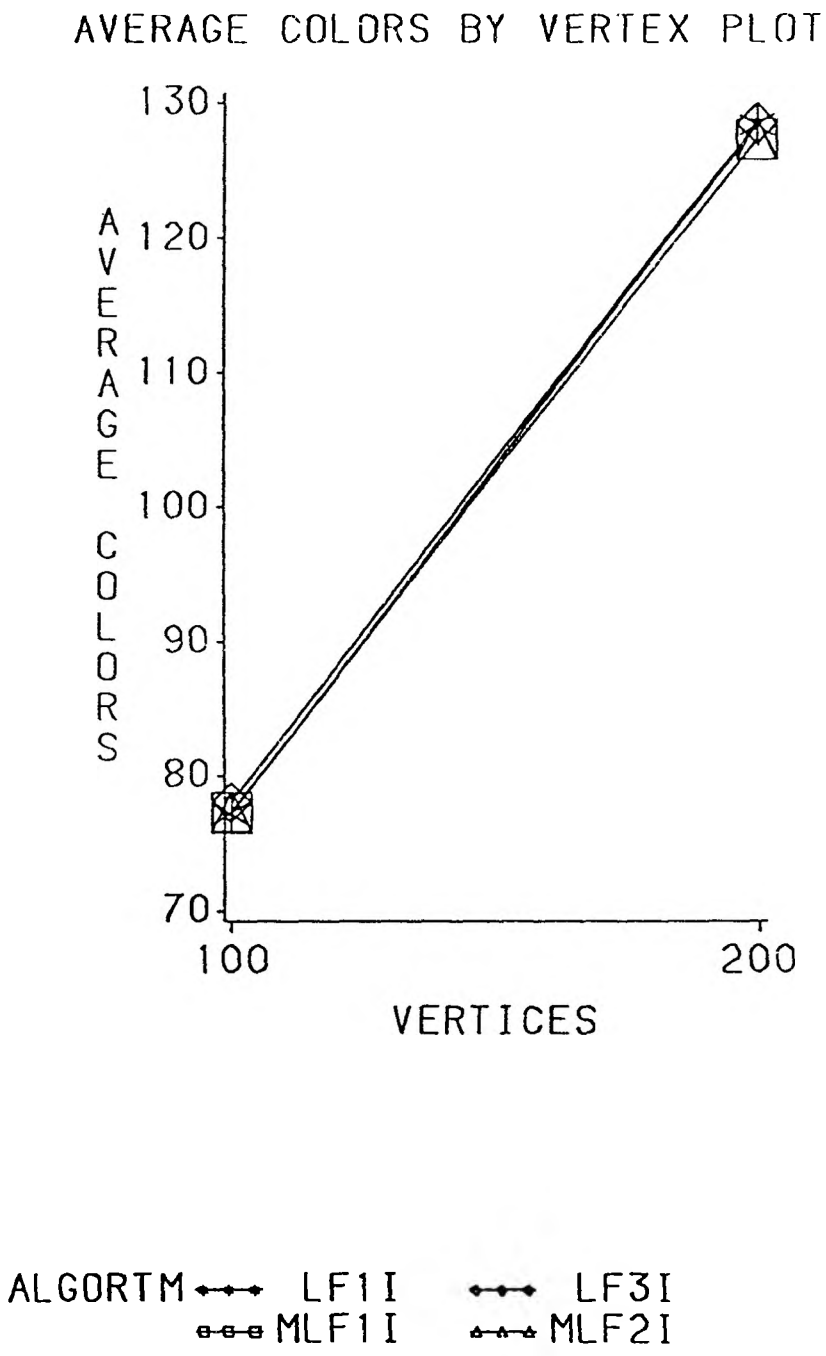

Figure 33. Graph with UF distribution.

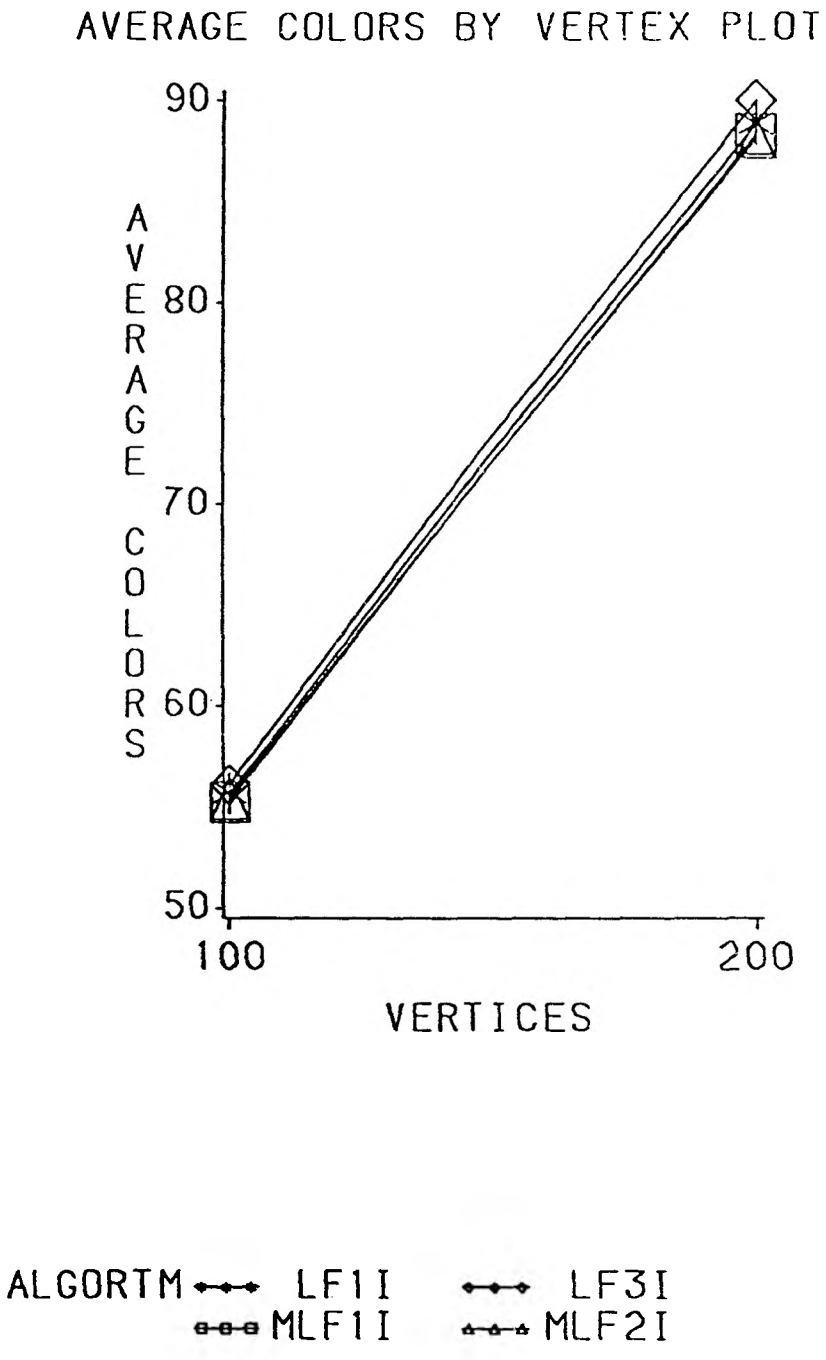

Figure 34. Graph with DR distribution.

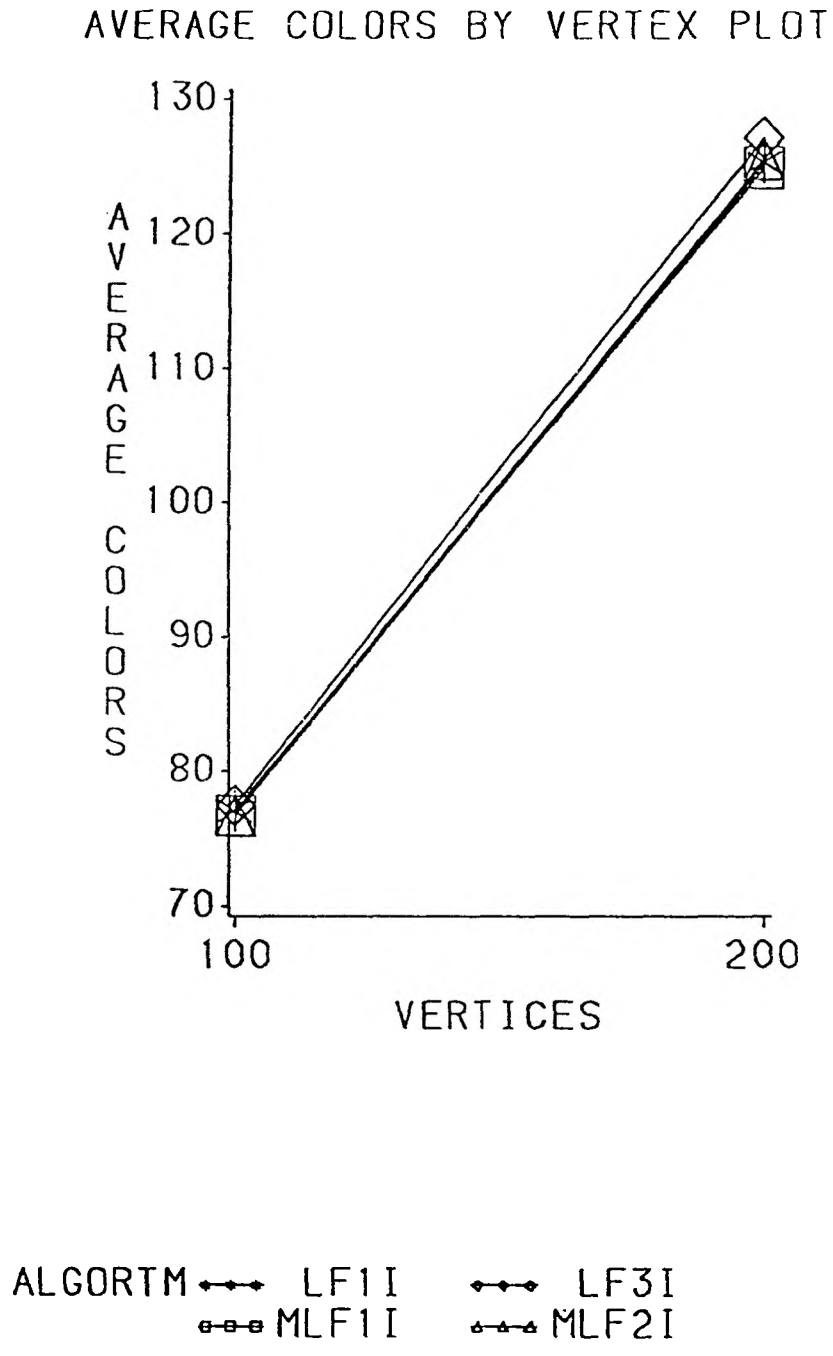

Figure 35. Graph with BN distribution.

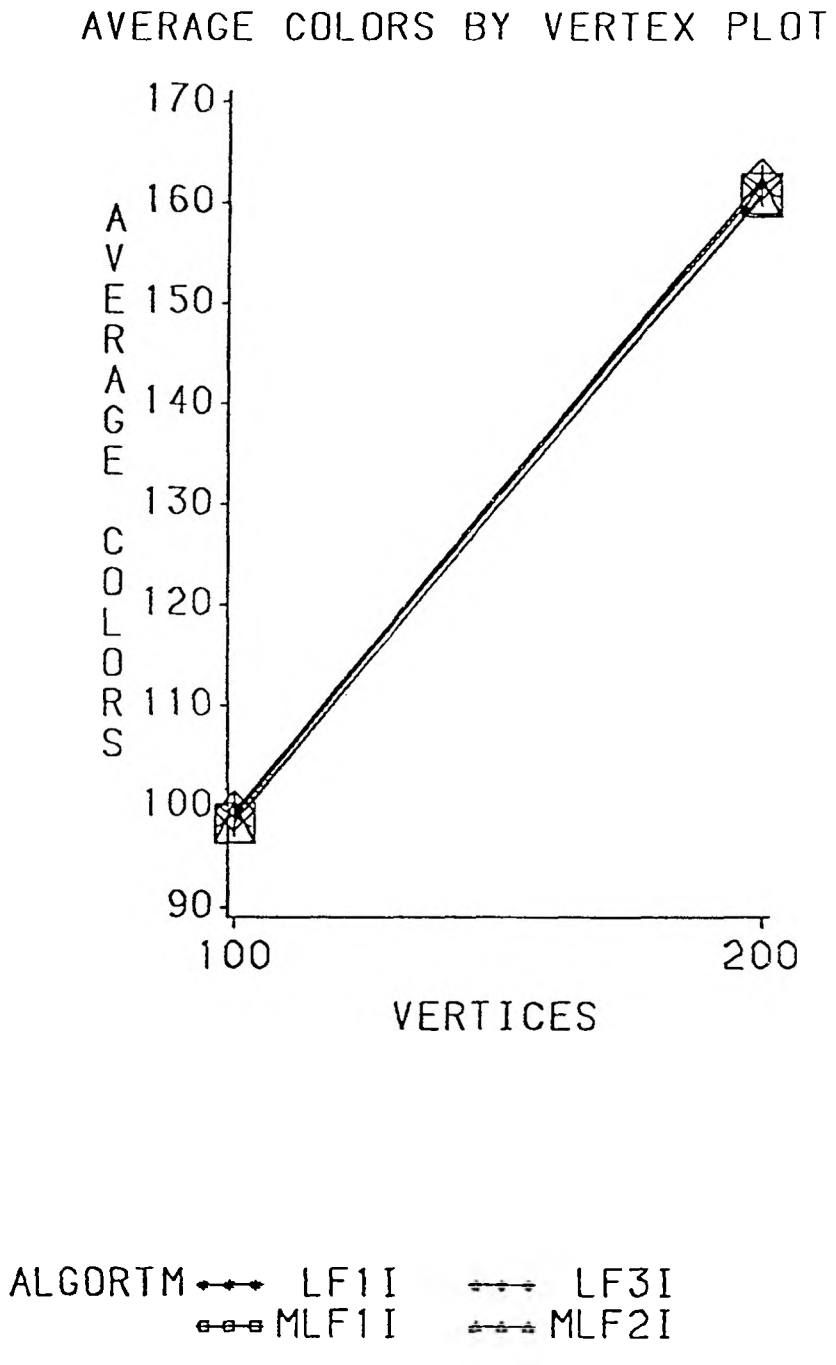

Figure 36. Graph with UR distribution.

By observing the number of wins of each of the best four heuristic algorithms there is clearly no one best algorithm which won consistently over all the test random composite graphs administered to it. To corroborate the current results with the experimentation by Clementson et al (3) and Roberts (5), four heuristic algorithms, LF1, LF1I, LF2 and LF2I, which were used in the experiments of Clementson et al and Roberts, were selected. Edge densities of 0.2, 0.3 and 0.4 with the exact truncated Poisson distribution from Clementson et al's experiment were used and furthermore, Roberts' five distributions (with probabilities rounded to three decimal places to the right of the decimal point) including the truncated Poisson distribution were used to corroborate with his experiment. The primary corroboration objective was to check/verify the consistency of the four heuristic algorithms in the current experiment with the four heuristic algorithms used in Clementson et al's experiment and Roberts' experiment. The secondary corroboration objective was to make sure that the chromaticity distributions corresponded to the average colors produced by the four heuristic algorithms when compared to the results of Clementson et al and Roberts. A method that was used to verify this process was to calculate the percentage deviation of average colors used in the other experiments with respect to the current experiment. The expression for calculating the % deviation is given by:

$$\%\ deviation = \frac{(x - y)}{y} \times 100$$

where x is the average colors used in Clementson et al (3) and Roberts (5) for each chromaticity distribution experiment and y is the average colors obtained from Table III, Table IV and Table V for each chromaticity distribution of this experiment. Table XII shows the % deviation in parenthesis besides the average color used in the chromaticity distribution experiments by Clementson et al (3) and Roberts (5).

n = 100, p = 0.2	LF1	LF1I	LF2	LF2I
TP (Clementson)	16.80(-2.33%)	15.60(-0.26%)	19.20(-5.14%)	16.20(-9.60%)
TP (Roberts)	17.24(0.23%)	15.68(0.26%)	19.68(-2.77%)	17.56(-2.01%)
UF (Roberts)	65.36(12.61%)	59.56(12.29%)	73.40(11.15%)	64.64(11.60%)
DR (Roberts)	46.12(-1.03%)	42.08(-1.31%)	52.20(-3.33%)	46.24(-2.61%)
BN (Roberts)	63.12(-0.76%)	57.08(-0.35%)	66.40(-1.95%)	59.40(0.20%)
UR (Roberts)	81.04(-1.84%)	74.24(-0.16%)	91.44(1.69%)	79.72(1.01%)
n = 100, p = 0.3	LF1	LF1I	LF2	LF2I
TP (Clementson)	21.80(-5.05%)	20.20(-4.18%)	25.60(-2.59%)	22.20(-6.72%)
TP (Roberts)	22.20(-3.31%)	20.68(-1.90%)	25.84(-1.67%)	22.56(-5.21%)
UF (Roberts)	85.72(0.33%)	77.76(-0.77%)	95.84(-3.74%)	85.64(-2.73%)
DR (Roberts)	59.48(-0.93%)	53.48(-1.84%)	68.72(-1.83%)	60.76(-0.13%)
BN (Roberts)	83.20(-1.05%)	75.44(-3.18%)	90.44(-2.25%)	79.08(-2.23%)
UR (Roberts)	106.16(0.19%)	97.60(0.33%)	119.96(0.71%)	104.56(1.12%)
n = 100, p = 0.4	LF1	LF1I	LF2	LF2I
TP (Clementson)	29.20(5.04%)	26.00(1.25%)	32.00(-0.50%)	29.60(0.82%)
TP (Roberts)	27.76(-0.14%)	25.28(-1.56%)	31.24(-2.86%)	28.20(-3.95%)
UF (Roberts)	104.84(-1.28%)	96.08(-2.75%)	120.20(-2.72%)	105.32(-3.16%)
DR (Roberts)	72.36(-3.16%)	68.12(-1.22%)	84.12(-3.22%)	75.68(-3.57%)
BN (Roberts)	101.68(-0.47%)	93.12(-2.47%)	113.36(0.00%)	99.24(0.32%)
UR (Roberts)	130.76(1.71%)	120.28(-2.05%)	148.00(-0.62%)	131.00(-0.49%)

Table XII. CORROBORATION OF RESULTS WITH CLEMENTSON ET AL AND ROBERTS.

The negative % deviation shows the average color used in this experiment for that particular chromaticity distribution is larger than that of Clementson et al or Roberts and the positive % deviation is just the opposite of the negative % deviation. By gathering the extreme (largest) positive % deviation and extreme (smallest) negative % deviation from Table XII, which are between -9.60% and 12.61%, there is 1.505% $\pm$ 11.105% recorded by Clementson et al and Roberts. It also shows the average colors in this experiment for graph vertices, n = 100 and edge density, p = 0.2, the average colors were slightly lower by 1.505% Note, edge densities of p = 0.3 and 0.4 are not included in the above conclusion because the two extreme % deviation were obtained at p = 0.2 with n = 100 where the average colors are small. It is known a slight change in small numbers will result in % deviation to swing widely and therefore it is paradoxical to assume for Table XII that the average colors in this experiment were lower by 1.505% from the average colors of Clementson et al and Roberts. However, as the average colors get larger and larger, the extreme points of % deviation will get smaller and smaller as can be seen for the case n = 100 with p = 0.3 and 0.4 of Table XII.

Among the eleven algorithms, four heuristic algorithms were competing fiercely with one another. Figure 11 to Figure 16 are plots with chromaticity distributions. All algorithms showed close colorings. In Figure 17 to Figure 26, varying the edge densities, did not retard any of the algorithms. As the vertices increased as in Figure 27 to Figure 36, all algorithms seemed to do well without dropping off from the others. Since most of the plots showed

close competitions, a table is set up to show the algorithm that has the most wins in the close race. Table XIII shows the number of actual wins of each algorithm from Table II to Table XI results. An algorithm is a winner if it has the smallest average colors when compared with other algorithms for that chromaticity distribution. If a tie occurs then those algorithms involved are declared winners. The results in Table XIII show the MLF1I algorithm as the overall winner follow very closely by the MLF2I algorithm. The LF1I algorithm came in as third follow by the LF3I algorithm as fourth.

Algorithms	LF1I	LF3I	MLF1I	MLF2I
Table II	-	2	2	1
Table III	1	1	2	1
Table IV	-	-	3	3
Table V	-	-	2	3
Table VI	2	-	1	2
Table VII	-	1	2	2
Table VIII	-	-	3	2
Table IX	1	-	3	1
Table X	1	-	2	3
Table XI	1	-	2	3
Total wins	6	4	22	21

Table XIII. NUMBER OF ABSOLUTE WINS OF EACH ALGORITHM.

VII. CONCLUSION

By observing the number of wins of each of the four best algorithms, LF1, LF1I, MLF1I and MLF2I, there is clearly no one best algorithm which won consistently over the test random composite graphs administered to it. The results showed the MLF1I to be better than the MLF2I algorithm. The overall winner position is not absolute due to the relative closeness of the number of wins of the two MLFxI algorithms.

Even though the MLFxI algorithms seized the top positions, another factor to consider is the run time of these algorithms. The preordering of vertices for all algorithms except the DSxI algorithms uses the heapsort method and therefore is $O(n \log_2 n)$. The DSxI algorithms only selects the vertex that has the highest maximal degree which used O(1). The degree saturation method is known to have the run time $O(n^2)$ as reported by Brelaz (7). The vertex sequential coloring algorithms without the interchange procedure has O(n) run time. Since the run time of the interchange procedure is not known, let it be on O(I). The vertex sequential with interchange algorithms run times are less than $O(n^2)$ when compared with the DSx algorithms coloring the same composite graphs. The worst case analysis of the interchange procedure is also not known. When comparing the run times, definitely the LF1I algorithm and the LF3I algorithm top the list.

The DSxI algorithms simply did not fair well in this experiment at all. However as a note of interest, when the DSxI algorithms were ran with

vertices n = 20 or less they performed very well against the rest of the algorithms in the current experiment.

Corroboration with Clementson et al's (3) experiment and Roberts' (5) experiment showed the reproduction of Clementson et al's algorithms, LF1, LF1I, LF2 and LF2I, were consistent with their algorithms. There were not much % deviations from the results of the current experiment compared to their experiments from the sampling done in Table XII.

Cross comparison of results from the current experiment with Roberts' experiment or Clementson et al's experiment is not possible. The reason is each experiment was conducted using a set of test random composite graphs which is different in each experiment. Therefore it is not possible to judge the results from each of the experiment and conclude the overall best algorithm from the experiments. However, a conclusion can be drawn by comparing the new algorithms with Clementson et al's algorithms in the current experiment. The new algorithms, LF3I, MLF1I and MLF2I, are as good as Clementson et al's algorithms if not better.

An exact composite graph coloring algorithm has been attempted in the research but due to the exponential run time of the algorithm, the algorithm was not used in the experiment to solve large graphs with the heuristic algorithms. The exact algorithm uses the implicit enumeration method to find an exact solution. No trimmed down version of the exact algorithm is attempted due to the unknown properties of composite graph.

On the applications of composite graph coloring, those applications listed namely, the timetabling problem, job shop scheduling problem, cpu scheduling problem and network assignment problem are not limited to those application problems nor limited to those exact problems presented. For example, in the timetabling problem the composite graph coloring problem can easily be used to schedule an examination timetabling.

The job shop scheduling problem certainly posed a limitation to composite graph formulation by not being able to prioritize the operations in a job. In the next chapter about future research, a modified composite graph is proposed to hopefully overcome the priority limitation as exists in the current composite graphs.

VIII. FUTURE RESEARCH

As mentioned before, the composite graph coloring problem is still relatively new in the world of graph coloring. Applications using the composite graph formulation have not been reported or published and certainly would be worth while to investigate further.

Heuristic algorithms developed so far did not show one superior algorithm over the rest. New heuristic algorithms should be attempted to gather a pool of good heuristic algorithms. To find a good coloring for a particular composite graph, all the good heuristic algorithms from the pool should be tried.

Properties of the composite graphs need to be investigated more thoroughly to assist in the development of better heuristic algorithms and possibly exact algorithms which will not be that prohibitive with respect to run time.

The job shop scheduling problem using the composite graph formulation as it is, caused a limitation in which operations represented by the vertices of the graph cannot be prioritized within the job. A modified composite is proposed. Suppose a vertex now represents a job instead of an operation. The job in turn might consist of a sequence of operations with arbitrary processing time in a given order. By allowing the vertex to hold all the operations of the job and fixing the order of the operations then perhaps a modified composite graph coloring algorithm can be allowed to color the operations in the given order. Another research possibility might

be to find the minimum completion time of each operation since the algorithm created in the research could not handle combinations of jobs and operations well. Overall, the job shop scheduling problem is worth while investigating further as with the rest of the other applications.

REFERENCES

1. Peemoller, Jurgen. Numerical experiences with graph coloring algorithms, European Journal of Operational Research, Vol. 24 (1986). pp. 146-151.

2. Larus, J.R. & Hilfinger, P.N. Register allocation in the SPUR LISP compiler, SIGPLAN Not. (USA), Vol. 21, No. 7 (July 1986) pp.255-263.

3. Clementson, A.T. & Elphick, C.H. Approximate Colouring Algorithms for Composite Graphs, J. Opl. Res., Vol. 34, No. 6 (1983) pp. 503-509.

4. Aho, A.V., Hopcraft, J.E. & Ullman, J.D. The Design and Analysis of Computer Algorithms, Addison-Wesley, Reading, MA (1974), pp.364-404.

5. Roberts, Johnnie C. Heuristic coloring algorithms for the composite graph coloring problem, Dissertation, Ph.D., UMR (1987).

6. Leighton, Frank T. A graph coloring algorithm for large scheduling problems, Journal of Research of the National Bureau of Standards, Vol. 84, No. 6 (Nov.-Dec. 1979) pp. 489-506.

7. Brelaz, Daniel. New Methods to Color the Vertices of a Graph, Comm. of ACM, Vol. 22, No. 4 (April 1979) pp.251-256.

8. Spinrad, J.P. & Vijayan, G. Worst case analysis of a graph coloring algorithm, Discrete Applied Mathematics, Vol. 12 (1985) pp.89-92.

9. Manvel, B. Coloring large graphs, Proc. 12th Southeastern Conference on Combinatorics, Graph Theory and Computing, (1981) pp.197-204.

10. Tremblay, J.P. & Manohar, R. Discrete Mathematical Structures with applications to Computer Science, McGraw-Hill Computer Science Series.

11. Ore, O. The four colour problem, Academic Press, New York. (1967).

12. Appel, K. and W. Haken. Every Planar Map is 4-colourable, Bulletin American Mathematical Society, Vol. 82 (1976) p.71.

13. Wood, D. C. A Technique for Colouring a Graph Applicable to Large Scale Timetabling Problems, Computer Journal, Vol. 12 (1968) p.317.

14. Williams, M. R. A Graph Theory Model for the Solution of Timetables, Ph.D. Thesis, University of Glasgow (1968).

15. Hale, W. K. Frequency assignment: Theory and applications, Proceedings of the IEEE, Vol. 68 (1980) pp.1497-1514.

16. Dencker, P., Durre, K., Fels, G., and Heuft, J. Compression of parser tables, preprint, Fakulat fur Informatik, Universitat Kalsruhe.

17. Matula, D. W., Marble, G., and Isaacson, J. D. Graph coloring algorithms in: Graph Theory and Computing, R.C. Read, Ed., Academic Press, New York, (1972) pp.109-122.

18. Johnson, D. S. Worst Case Behavior of Graph Coloring Algorithms, Proceedings of the 5th Southwest Conference on Combinatorics, Graph Theory and Computing, (1974) pp.513-527.

19. Christofides, N. Graph Theory, Academic Press, London, (1975) pp.31-57.

20. Welsh, D. J. A. and Powell. M. B. An upper bound on the chromatic number of a graph and its application to timetabling problems, The Computer Journal, Vol. 10 (1967) p.85.

21. Punter, A. School timetabling and graph coloring, Dissertation, Ph.D., C.N.A.A., Hatfield Polytechnic (1976).

22. Lotfi, V. and Sarin, S. A graph coloring algorithm for large scale scheduling problems, Comput. & Ops. Res., Vol. 13 No. 1 (1986) pp.27-32.

23. Mehta, N. K. The application of a graph coloring method to an examination scheduling problem, Interfaces, Vol. 11 No. 5 (October 1981) pp.57-65.