A Study of Graph Coloring Heuristics And its Application to a Time Tabling Problem In a Registration and Scheduling System

by

Saud Mohammed Al-Muhaidib

A Thesis Presented to the

FACULTY OF THE COLLEGE OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

In

INFORMATION AND COMPUTER SCIENCE

July, 1991

Order Number 1354054

A study of graph coloring heuristics and its application to a time tabling problem in a registration and scheduling system

Al-Muhaidib, Saud Mohammed, M.S.

King Fahd University of Petroleum and Minerals (Saudi Arabia), 1991

U·M·I

300 N. Zeeb Rd.
Ann Arbor, MI 48106

A STUDY OF GRAPH COLORING HEURISTICS AND ITS APPLICATION TO A TIME TABLING PROBLEM IN A REGISTRATION AND SCHEDULING SYSTEM

SAUD MOHAMMED AL-MUHAIDIB

A Thesis Presented to the

FACULTY OF THE COLLEGE OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

Dhahran, Saudi Arabia

In Partial Fulfillment of the

Requirements for the Degree of

MASTER OF SCIENCE

IN

INFORMATION AND COMPUTER SCIENCE

July 1991

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

This Thesis, written by

Saud Mohammed Al-Muhaidib

under the direction of his thesis committee, and approved by all the members, has been presented to and accepted by the Dean, College of Graduate Studies, in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN INFORMATION AND COMPUTER SCIENCE

Thesis Committee:

S. Ghanta
Chairman (Dr. S. Ghanta)

Muhammad Shafiq
Member (Dr.M. Shafique)

Member (Dr. N. M. Shaikh)

Dr. M. A. Al-Tayyeb
Department Chairman

Dr. A. H. Al-Rabeh
Dean, College of Graduate Studies

Date: 12/1/92

كلية الدراسات العليا
COLLEGE OF GRADUATE STUDIES

TABLE OF CONTENTS

Chapter *Page*

ACKNOWLEDGEMENT iv

ABSTRACT v

1. INTRODUCTION 1

1.1 Motivation 1

1.2 Informal Description of Registration and Scheduling Aspects 3

1.3 Why this Thesis 5

2. THE REGISTRATION SYSTEM 7

2.1 Current System 7

2.1.1 Student Life Cycle 7

2.1.2 Drawacks/Limitations of the Current Manual System 9

2.2 Proposed System 11

2.3 Design and Implementation 14

2.3.1 Requirements Analysis 14

2.3.2 Conceptual Design 15

2.3.3 Logical Design 17

2.4 Physical Design 22

3. EXAMINATION SCHEDULING AND TIMETABLING 23

3.1 Problem Statement 23

3.2 Previous Work 26
3.2.1 Grouping Operation 26
3.2.2 Langrangian Relaxation Method 29
3.2.3 Flow Methods 29
3.2.4 Improving Infeasible Timetables 30
3.3 Formualtion as a coloring problem 30

4. COLORING HEURISTICS 35
4.1 Greedy heuristic 35
4.2 Independent set Heuristic 37
4.3 No-Choice Algorithm 41
4.4 The Brelaz Heuristic 45
4.5 The Random-Clique Algorithm 49

5. ENHANCEMENTS 50
5.1 Node Splitting 50
5.2 Preassignment of Colors 52
5.3 Ordering and Priority 53
5.4 Node Spilling 54

6. EXPERIMENTAL RESULTS 55
6.1 Testing System 55
6.2 Test Results 58
6.3 Graph Analysis 60
6.3.1 Effect of Size and Ratio 60

6.3.2 Effect of Node Splitting 77

6.3.3 Effect of Changing Environment 78

6.4 Real Life Experiment 78

7. HYBRID ALGORITHM 106

7.1 Hybrid Algorithm 106

7.2 Test Results 107

7.3 Implementation of Hybrid Algorithm 124

8 CONCLUSIONS 126

APPENDIX 128

REFERENCES 137

ACKNOWLEDGEMENT

All praises and glory to Almighty Allah without Whose help no work can be accomplished. Acknowledgement is due to King Fahd University of Petroleum and Minerals for providing support to this research.

My heartfelt thanks and gratefulness go to my parents and wife who had the foresight to provide support and encouragement for my education.

I am also grateful to Dr. S. Ghanta, major thesis advisor, for his continuous guidance and assistance during the period of this research work. It was his insight and knowledge of the subject matter that made this task easier.

I wish to thank the thesis committee members Dr. M. Shafique and Dr. Nassar Shaikh for their criticism and valuable suggestions.

Finally the cooperation of Mr. Hussain Al-Sukaibi, the Principal of the Dhahran High School is appreciated.

ABSTRACT

The Ministry of Education of Saudi Arabia has converted most high schools in the Kingdom to the semester-credit-system (SCS). Even though it is a sophisticated system, a number of problems arose with its implementation. Registration of students, scheduling of courses and examination scheduling are tedious tasks of the system to be carried out smoothly.

In the first part of this work we design and implement a registration and examination scheduling system. The solution to scheduling uses Graph Coloring heuristics on the course conflict graph produced from the student registration file. Different variants of examination and time-tabling problems can be formulated as graph vertex coloring problems.

In the second part of this work we present various heuristics to the NP-Hard coloring problem. We also introduce the concepts of node-splitting and node-spilling. These concepts are helpful in situations where a desired k-coloring is necessary, even if the problem instance does not admit a k-coloring.

In the final part of the thesis we conduct a detailed empirical study of the efficiency and quality of various heuristics, and propose a hybrid algorithm that mimics the best heuristic for the given graph instance.

CHAPTER 1

INTRODUCTION

1.1 Motivation

High schools in the kingdom have been converted to the semester-credit-system (SCS). Under the SCS system a student can select the periods, instructors, and courses from an offered list, subject to certain rules. This flexibility causes a number of problems for the school staff. These problems are due to the tedium involved in registration, examination scheduling and course offering.

Final examinations are held at the end of each semester. All examinations should be finished in eight days. Two periods are set every day for the examinations. In total we have 16 periods to set the examinations in. Each examination period is called a time slot.

Courses, instructors, students and room assignments should all be considered to produce a good schedule. Two courses that are required by a student, taught by the same instructor, or require the same room cannot be offered in the same period. The examination of these courses cannot be held in the same time slot either; this is considered a conflict.

Currently the procedures of the SCS are carried out manually. In spite of the numerous advantages, it is burdening the faculty with excessive administrative work. Despite the fact that it is taking significant amount of time, the manual scheduling of courses and examinations are causing too many problems.

It takes a long time and a lot of effort to register the students. A school has on the average 500 students. Each student may register for courses different from those registered by his fellow student. Each student registers 10 courses on the average. The courses should be written on the student registration sheet and the student name should be written on the class roster. On the average each class has 20 students in it.

To produce a transcript, an advisor picks up the grade of each student from the class-grade-sheet. This process is tedious. For each student's course an advisor searches a number of class rosters.

Automation is required for the SCS system to save time and effort of school staff. There exist some automated systems such as:

1. Registration and scheduling system of KFUPM (King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia).

2. Registration and scheduling system of KSU (King Saud University, Riyadh, Saudi Arabia).

3. Sakhar registration system [SAKH89].

These systems are unsuitable for the requirements mentioned above due to the following reasons.

- They are too large to run on a personal computer.
- The ones that run on personal computer do not have scheduling systems.
- They are written by companies from outside Saudi Arabia. Technical support and customization to local needs is difficult to achieve.
- Systems with scheduling subsystems do not have good scheduling heuristics.

A good scheduling algorithm is required. An automated computerized implementation of the examination scheduling algorithm is highly desirable and so is the registration system.

1.2 Informal Description of Registration and Scheduling Aspects

SCS has many new features. We describe here only the important features from the design point of view.

In SCS different students can take different credit loads thus enabling students to progress at their natural speeds. To prevent an over ambitious

student from taking much more than what he can handle, and to allow students progress with a guaranteed rate of progress, there are maximum and minimum limits on the number of credits a student can take.

Exceptions do come into picture with respect to students under probation (those who are not doing well), the graduating students, and those who have demonstrated their superiority by performing very well during the past 2 or 3 semesters.

SCS allows different students to take courses at different periods with different instructors and during different semesters. This feature helps the student to register the difficult courses at his most convenient time and with the instructor that he can grasp the most with. It also allows the student to delay or start with the courses that he finds most convenient, provided that he is within the limits of maximum and minimum load, and within the framework of courses required for graduation. It allows the student to plan his day, when to start, when to end and when to have rest hours.

New subjects can be added to the system as desired by the ministry causing the least effect on the student schedule. The ministry can change the credits for a course, or cancel it. If a course cannot be offered in a certain semester, due to delay of text books or instructors, the student can take it in the following semester. In other words, SCS is meant to support an evolving curriculum which is smoothly adjusted according to need, demand and new developments.

1.3 Why This Thesis?

This thesis has been selected because of the following reasons:

1. *The wide effect of the results of the research*:

There are hundreds of high schools in the kingdom that follow the SCS system. Each school spends a significant amount of effort in solving its registration and scheduling problems.

2. *The repetition of the use of the results*:

The SCS schools repeat the registration and scheduling process every semester. This means a large number of hours is spent every semester on the same problems. This research can save a lot of time and effort for each school.

3. *The similarity between this problem* and *the graph coloring problem*:

There are many heuristics that can help if well selected or modified to suit this problem. A course can be considered as a vertex. A vertex conflict graph (a course conflict graph) is built as will be shown in Chapter 3.

The approaches for solution of the timetabling problem have varied from pure simulation of hand construction of a time table [BARR 65] to sophisticated algorithms. The programs developed by Purdue University in 1956 and CLASS (Class Load and Student Scheduling) developed by IBM shortly thereafter fall in the category of simulation of hand construction of timetabling. Most of the

previous studies have dealt with school timetabling problems (see for instance, [BARR 65], [LION 66]), [AUST 73], [AUST 76]. There are relatively few studies dealing with the College or University timetabling problem [ALMO 66], [ALMO 69], [AKK 73], [WHIT 76]. An attempt was made to solve the school timetabling by the flow method [WERR 71]. Attempts to solve the timetabling based on Langrangean relaxation are discussed in [FISH 84]. Finally employing graph coloring has been the theme of [WHIT 79], [MEHA 81], [TURN 88], [KARC 88], [CHOW 90].

In this thesis we first describe the registration system in Chapter 2. We study the current mode of operation, propose a system and then describe the design and implementation of the new system. It is at that point that the work related to examination scheduling and timetabling becomes more important. We define in Chapter 3 the scheduling problem, mention the previous work on it and then formulate it as a coloring problem. We study the examination scheduling algorithms in Chapter 4. Enhancements to these algorithms are covered in Chapter 5. The concepts of node splitting, preassignment, and vertex coloring priority and spilling are introduced in this Chapter. Experimental results are shown in Chapter 6. As well, a test was made on Dhahran high school examination scheduling in Spring 1989. We assess the previous heuristics and develop a hybrid algorithm that achieves good performance in Chapter 7. Finally we conclude this work and discuss the future work in Chapter 8.

CHAPTER 2

THE REGISTRATION SYSTEM

In this chapter, we will first describe the current mode of operation in the SCS schools. A computerized system will be proposed and then a data base is designed and implemented. Input to and output of the current mode of operation as well as the implemented system are shown in the appendices.

2.1 Current System

Since the SCS system is made to help the student in his selection of courses and time periods, we are going to illustrate the student life cycle to be able to notice the interaction between the student and the SCS system. This interaction information is very important to notice the bottlenecks of the system.

2.1.1 Student Life Cycle

The student follows a path of the following life cycle:

a- Students that enroll in SCS schools are usually:
- transferred from similar schools.
- graduated from intermediate schools.
- transferred from traditional high schools.

b- If the student is graduated from an intermediate school, he is offered one of the preset schedules. He then selects one of these preset schedules by registering his name in the class roster. If the student is transferred from a traditional school his degree is evaluated. Students transferred from similar schools join other students in the school in their course selection. They select the courses they will take from an offered course list. They do this by filling a registration form. Each student is assigned an advisor to help him in his academic program.

c- A student must have the approval of course instructor and his advisor for each course he intends to take. If courses are closed the student can revise his registration form and select other courses.

d- A student has the right to drop some of his courses provided that he carries a minimum of 20 credits. One can drop courses only during the first three weeks. To drop a course a student is required to fill a drop form and then have it approved by both the instructor and advisor.

e- If a student does not attend 15% of the course meetings, he is dropped from the course with an "omitted" mark as his grade for the course.

f- Class grades are submitted to the registrar to produce the transcripts. The student's grade in a course is made up of 25% midterm examination, 50% final examination, and 25% for the semester class work.

g- If a student fails he is given a probation. Three probations in a row

means that the student will be dismissed from the school. Once the student finishes all courses required for his graduation, he is given the graduation degree.

h- During this life cycle a student could change his major, withdraw for a semester, or quit from the school.

2.1.2 Drawbacks/Limitations of the Current Manual System

Quite a bit of work is required by the school to plan each student's education three years ahead. In its manual implementation limited time forces one to focus on the immediate needs and problems. As a consequence of this narrow focussing, too many difficulties are coming into picture as drawbacks. Some courses that have to be taken before others (prerequisites) are often overlooked for the graduating students. The cause of this is the delaying of necessary prerequisite course(s) until the very end and also the failing of students in the prerequisite courses.

Graduating students may have to take more than 8 three-hour courses which cannot be set in normal schedules. Or a collection of students might cause this problem. For example, student A needs courses (1,2,3,4,5,8), while student B needs courses (3,4,5,6,7) and student C needs courses (1,2,6,7,8). Hence, some courses may have to be offered twice in a semester with one or two students each.

In some periods all rooms are occupied and more rooms are needed while in other hours fewer rooms are used. This causes modifying course periods which is, along with its consequences, a cumbersome activity to deal with.

Students that did not pass in a major for two semesters must change their majors, thus causing the student to be late in some course sequences and fast in other course sequences. This limits the number of credits that the student can take because no prerequisite is offered (or courses need prerequisites that he does not have as he was not in the major).

Students with high G.P.A. have priority in registration. That causes inconvenience to students with low G.P.A. as they find courses closed. Then 2 weeks later the high G.P.A. students may drop from 5 hours or more leaving open courses, that were needed by low G.P.A. students, but who could not then get into them because addition of courses is not possible at that late time.

Students are allowed to transfer from and to traditional schools. Because of this transfer a student may have to take courses that are not in the plan (not offered) or take two courses such that one is a prerequisite of the other.

Tests have to be graded, total grades have to be calculated, letter grades have to be filled into transcripts and checked and rechecked within three days. Knowing that there are over four hundred students and an average of twenty eight hours/semester/student will make it easy to see the reason why many errors occur in the current system.

Prior to the beginning of each semester a scheduling committee is made. This committee has to produce a list of offered courses for the current semester. This list has to satisfy students' needs, instructors' and room constraints. This is a very difficult job. This job requires a significant amount of time to produce a good list, which is not available.

Ignorance of Arabic type writing and the SCS system by most of the instructors causes the work to progress very slowly. This increases the needed time to prepare transcripts and offered course list.

2.2 Proposed System

SCS consists of four subsystems. They are the admission, the scheduling, the registration and the examination scheduling subsystems. The interaction between these subsystems is shown in Figure 1. In this thesis, we focus only on the registration and examination scheduling subsystems, which are the most desired subsystems for automation.

The admission subsystem gets its inputs the students' admission forms. Evaluation of course transfers, updation of students' record with transferred courses, if any, is part of this subsystem's activity. Modification to admission requirements are also processed by admission subsystem.

The scheduling subsystem produces a list of courses to offer, subject to various constraints and the needs of students, a list of instructors' timetables,

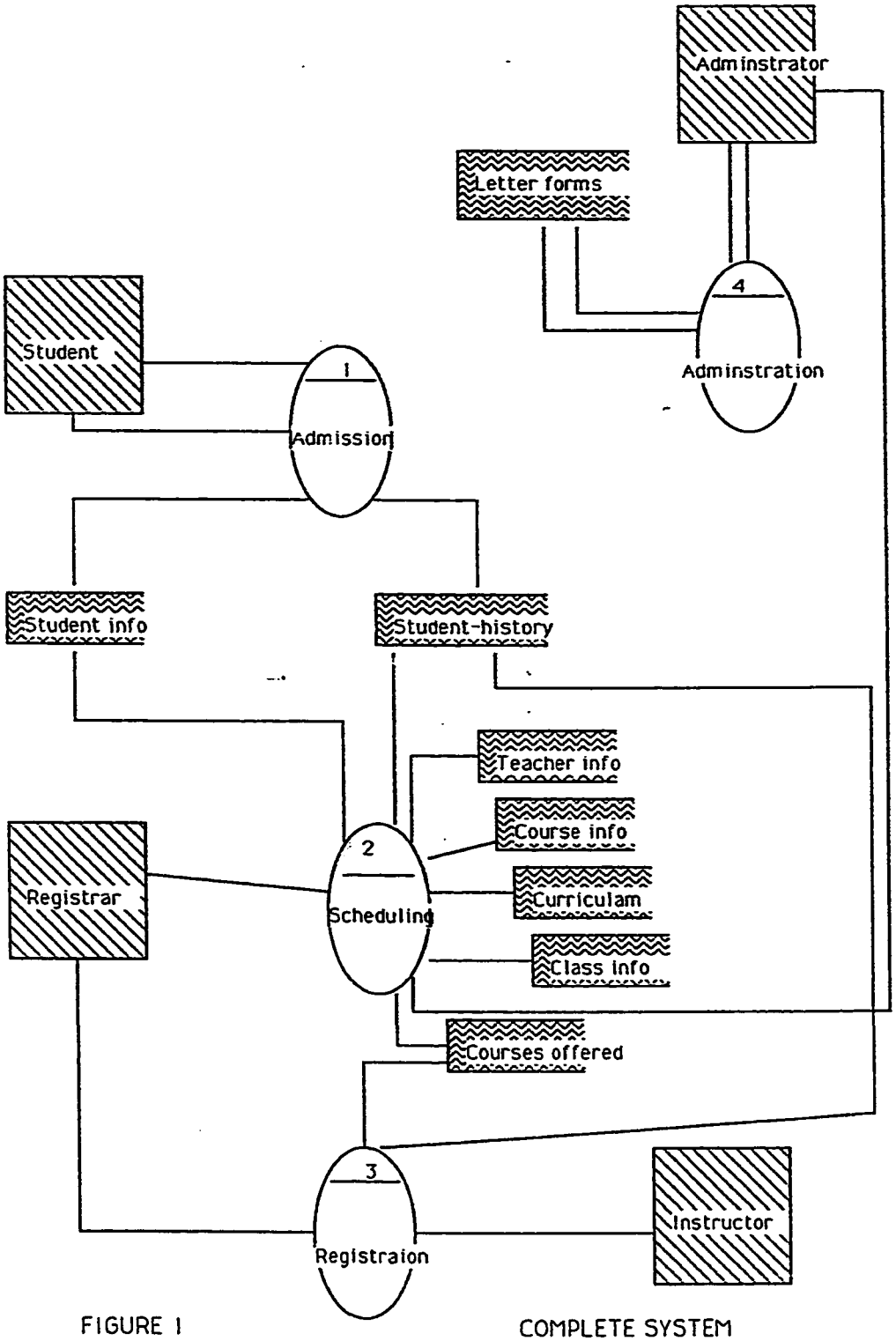


FIGURE 1

COMPLETE SYSTEM
DFD - LEVEL 1

rooms' timetables. The list of offered courses should include in it the periods, instructors name and room allocated. The curriculum (suggested sets of courses to be offered every semester) is set and modified by the scheduler.

The registration subsystem accepts the approved registration form. Each course in the form is approved if there is a place for the student in the section. Modifications can be made to the registration sheet. Adding and dropping courses are also processed by the registration subsystem. Grades are filled and transcripts are printed by the registrar.

The examination scheduling subsystem generates schedule for the examination periods of the courses. This subsystem needs the student registration information in the registration file. It builds a schedule that uses minimum number of periods to conduct the examinations without any student conflict, if possible.

This work proposes a data base registration system with an examination scheduling system. The proposed system stores, retrieves and modifies a number of files to support the above requirements such as an instructor, course, offered course and room files. The next section illustrates the design of this proposed system.

2.3 Design and Implementation

For the sake of brevity only the important steps of the design are shown. More details about sample inputs and outputs are given in the Appendix.

2.3.1 Requirements Formulation

Based on the study of the current system, its operational mechanism, its features and drawbacks, and based on many interviews with specialists who work with the current system, the following requirements are formulated that serve as the starting step for the design of the new system.

Customer Wishlist:

- Generally function keys are preferable, i.e., menu-driven interface.
- The system should process the admission requirements for each new student (fresh or transferred).
- The system should evaluate the certificate of students transferred from traditional schools.
- The system should allow for the transfer from one major to another.
- The system should help the registrar making the examination schedule by showing any conflicts as the registrar adds courses.

- The system should provide suggested examination timetables of teachers, classes and students.

- The system should allow for addition and deletion of courses by a student within the rules of maximum and minimum load.

- The system should allow addition or deletion of sections by the registrar showing him the consequences of that action.

The system should follow up the probations and decide about who should change major or who should be dismissed.

2.3.2 Conceptual Design

In this section SCS is shown as a Data Flow Diagram (DFD). There will be four DFD's. The top level one shows the complete system (See Figure 1). It is then made into three DFD's in the second level (see the Appendix).

The complete system DFD shows the data flow between the entities (student, registrar, instructor and administrator) and the data stores (student-info, student history, teacher-info, course-info, class-info, curriculum and courses offered) through the processes (admission, scheduling, registration and administration).

The admission process in level one is made into a DFD in level two. The admission DFD shows the data flow between the entities (student, registrar and

admission) and the data stores (admission requirements, student-info and student history) through the processes (admission requirements, modify requirements, admission evaluate courses, load courses, dismiss list).

The scheduling process in level one DFD is made into a DFD of level two as well. This DFD shows the data flow between the entities (registrar, administrator, student and instructor) and the data stores (student-history, course specifications, instructor-info, class, curriculum and course offered) through the processes (modify-course, modify-inst, modify-class, modify-curriculum, scheduling, modify-offered-list, list-course-timetable, list-offered, list-inst-timetabling, and list-room-timetable).

Finally the registration process in level one DFD is enhanced into a DFD of level two. This DFD shows the data flows between the entities (registrar, student, instructor) and the data stores (courses offered, student history and student-info) through the processes (register-student, modify-registration, produce-registration-slips, produce-probation-notices, load-grades, prepare-class-list, add-and-drop, change-major, list-expected-graduates, list-of-students-by-advisor, list-of-students-by-major, and grade-sheet-list.

After the design of the DFD in this section, we will derive the records and the relationships in the next section.

2.3.3 Logical Design

In this section, the relationships between the entities of SCS will be shown. An entity relationship diagram E-R will be given (Figure 2). Then the relation schema and the physical schema will be given.

The following ER diagram (Figure 2) shows a number of entities such as (course, major, student, dept, section, semester, and teacher), relations such as (major-courses, dept-majors, major-students, class, offering, teacher-speciality, teacher-offering-sections and teacher-dept) and some of the fields that belong to the entities or relations. The following table will show a detailed description of the relations and entity schemas.

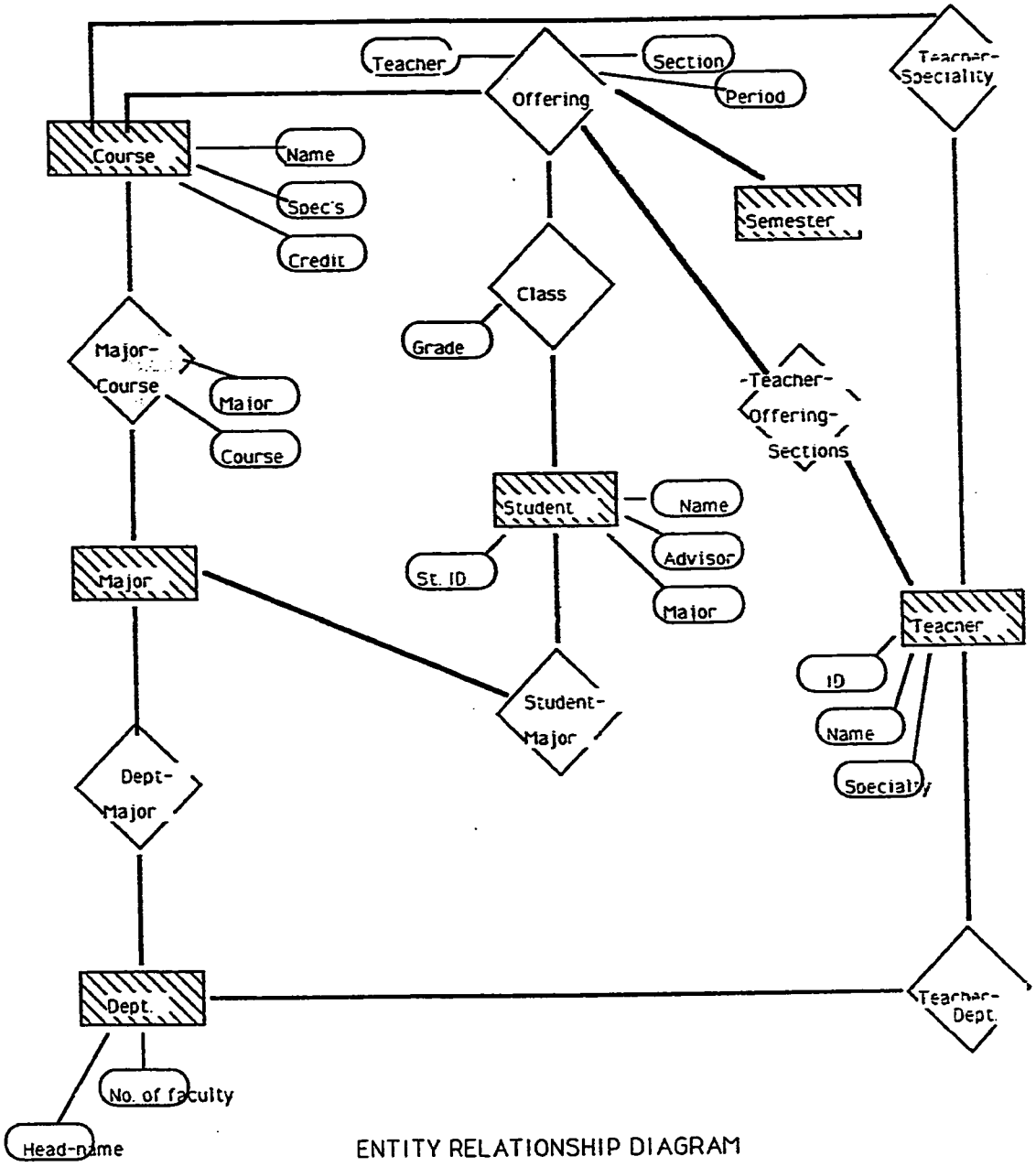
In this diagram, the rectangles represent entity sets, the ellipses represent attributes, diamonds represent relationship sets and lines link attributes to entity sets and entity sets to relationship sets [KORT 86].

Now we have the E-R diagram from which we can build the schema. For example, a class schema is: (course, course-name, semester #, section #, teacher-ID, student #, student-name, student-address, student-major, work grade, mid-term grade, final grade etc.). The following is the schema.

Name	Fields/Attributes
Class	course-ID, course-name, course-credits, section-number semester-ID, teacher-ID, room-ID, period, work-grade, midterm-grade, final-grade, student-ID, student-name
Teacher-speciality	teacher-ID, teacher-name, course#, course-name
Offered course	course-id, section-number, semester-id
Teacher-offered sections	teacher-id, teacher-name, course-id, section-id
Major-courses	major-id, course-id
Major-students	major-id, student-id
Dept-major	dept-id, dept-name
Teacher-dept	teacher-id, dept-number

A normalization process is made on this schema to produce normalized relations. An example is shown followed by the table of normalized relations.

We use the class schema as an example. This schema has a number of fields. In this process we will define the functional dependencies between the attributes of the schema. Attribute Y is said to be functionally dependent on attribute X if the value of X determines the value of Y ($X \rightarrow Y$).



ENTITY RELATIONSHIP DIAGRAM

FIGURE 2

course id → *course-name*
course id → *course-credits*
} *course relation*

course-id, section-number, semester-id → *teacher-id*
course-id, section-number, semester-id → *room-id*
course-id, section-number, semester-id → *period*
} *offering relationship*

((*course-id, section-number, semester-id*), *student-id*) → *work-grade*
((*course-id, section-number, semester-id*), *student-id*) → *midterm-grade*
((*course-id, section-number, semester-id*), *student-id*) → *final-grade*
} class info

student-id → *student-name* } *student relation*

We can replace the class schema by four relations (course, offering, class-info and student-info).

This process is repeated on all schemas and the resultant relations are checked to see if they imply the original set of functional dependencies. The result of the normalization process is shown in the following logical relational table.

Relation Name	Key Fields	Other Fields
Course	course-id	course-name, course-credit-hours, spec's, prerequisites, dept-name
Student	student-id	student-name, student-advisor-id, student-major, student-phone, student-date-of-birth, student-place-of-birth, student-enrollment-date, student-status
Major-courses	major-id course-id	
Teacher	teacher-id	teacher-name, teacher-id, teacher-speciality, teacher-phone, teacher-major, teacher-enrollment-date
Major	major-code	major name
Dept-teachers	dept-code teacher-id	
Room	room-No	room-size, room-type
Dept-Major	dept-id major-id	
Offered-courses	course-id semester-No Section-No	room-No, teacher-name, period
Class-Info	student-id course-id semester-No section-No	work-grade, midterm-grade, final-grade
Dept	dept-code	name, head, phone-No
Teacher-Speciality	teacher-id course-id	
Teacher-offered courses	teacher-id semester-id course-id	section-No

2.4 Physical Design

The relational table was used as a basis for the physical design. Relational schema that serves the registration system such as (student, class, offering) were used to implement a Dbase IV registration system. Fields were assigned sizes and types before their implementation.

SCS was designed using the DFD's, E-R diagrams and the normalization of its schemas. Selected registration schemas were physically designed and the registration system was implemented on a Dbase IV system. In summary, this Chapter described SCS. It proposed the development of the registration and examination scheduling of the four major components of SCS, the admission, the scheduling, the registration and the examination scheduling. A detailed design of the database has been carried out going through the requirements analysis, conceptual design, logical design. The next Chapter focusses on the examination scheduling subsystem.

CHAPTER 3

EXAMINATION SCHEDULING AND TIME-TABLING

In this chapter the examination scheduling problem will be discussed. It is known that the exam-scheduling and time-tabling problems are related [METH81, WHIT79]. So we focus our attention on the examination scheduling. This problem is part of the SCS subsystems described in the previous chapter. An illustrative example is given and then the previous work on the problem is mentioned. We then formulate the problem as a graph coloring problem.

3.1 Problem Statement

The examination scheduling problem is an NP-HARD problem. Even a restricted case of it is shown to be NP-HARD [EVEN 75]. Currently, the best known solution requires an exponential amount of computation time to produce an optimal schedule. A solution of the problem, that enumerates all possible solutions is very expensive and time-consuming as can be seen from the analysis below.

Let N be the number of courses offered.

16 examination time slots available for scheduling examinations

Course (1) examination can be held in any of the 16 examination time slots. The same can be said of courses (2), (3),....,(N). Therefore, the number of choices are:

16 choices for the first course x 16 for the second X....X 16

for the N-th course = 16 x 16 x 16 x ..... (N times) = 16^N choices.

But most of the choices in the above brute-force approach are not acceptable because of the occurrences of course conflicts. The role of the examination scheduler is to find which of these choices are acceptable. This job is tedious. In the conventional approach, it takes a significant amount of time and effort from the school staff.

A conflict graph is a structure that captures the essence of conflicts between courses and/or sections. There are two approaches to build the conflict graph depending upon whether the vertices of the graph are courses or sections.

The following example will illustrate this:

Ahmed registers Mathematics 1 Section A and Physics 2 Section A

Salch registers Chemistry 2 Section A and Mathematics 1 Section B

Mohamed registers Physics 2 Section B and Chemistry 2 Section B

If the course conflict approach is used then the course Mathematics 1

examination can neither be held at the same time slot as Physics 2, nor can it be held at the same time slot as Chemistry 2. This problem happens because neither Ahmed nor Saleh can have two examinations in the same time slot. Ahmed cannot have Mathematics 1 and Physics 2 in the same time slot, and Saleh cannot have Mathematics 1 and Chemistry 2 in the same time slot. Because of Mohamed, Physics 2 cannot be held in the same time slot as Chemistry 2. Three time slots are the least number of slots that can be assigned to the above case. That is because Mathematics conflicts with Physics and Chemistry, and Chemistry conflicts with Physics.

If sections-conflict-approach is used then the course Mathematics 1 Section A can be held at the same time slot as Chemistry 2 Section B and Physics 2 Section B. Likewise, Physics 2 Section A can be held at th same time slot as Mathematics 1 Section B and Chemistry 2 Section B. This means that the course is split into sections. This decreases the conflicts among sections. If we have 16 time slots, this approach increases the total number of slot selection to 16 choices per section. The total number of choices will be 16^N, where N is the number of sections instead of courses.

If the section-conflict-approach is used to solve the problem, then some sections of the same course can be merged in the same time slot. For example Mathematics 1 Section A and Mathematics 1 Section B may be held, if required, in the same time slot provided that the maximum number of time slots is not exceeded. This is called section [node] merging. This is usually more

convenient to the instructors, so that they can use the same examination paper.

If the scheduler produces more than one good schedule, other constraints are applied to the schedule selection. These constraints can be of high importance (rigid constraints). An example of that is a maximum number of students in a classroom. The other type of constraints are of low importance (soft constraints). An example of that is math examinations should be held on Saturday time slots to give the student more time to study during the weekend. Later in this Chapter the problem will be formulated as a coloring problem.

3.2 Previous Work

The examination scheduling problem received a significant amount of attention. Some of the papers followed similar method to papers presented earlier with minor changes. Others tried new ways to attack the problem. This Section will show some of these methods.

3.2.1 Grouping Operations [FISH84]

The idea of this method is to reduce the large number of interactions in the scheduling system. The grouping operations splits the interactions into a number of groups:

a- *Student Group:* is a group of students who are enrolled in the same stream.

Let $S_1,S_2,S_3,...,S_n$ be the subjects to be attended by the particular student group

Let $P_1,P_2,P_3,.....,P_n$ be the number of periods per week required by the subjects $S_1,S_2,.....,S_n$ respectively.

If $\sum_{i=1}^{n} P_i$ > NP (NP is the total number of periods per week)

then timetabling problem is infeasible and student group should be split into two groups.

e.g. if student group is S_1,S_2,S_3,S_4,S_5,S_6

then it could be a. S_1,S_2,S_4,S_6

b. S_1,S_3,S_4,S_5, or otherwise

b- *Subject Group*: the group of subjects which are attended by the same set of students.

student group / Subject	A	B	C
1	x		x
2		x	
3	x		
4		x	x
5	x	x	x
6	x	x	
7		x	
8	x		x
9	x	x	x
10	x		

e.g. subject group 1 = subjects 2,7 attended by B

subject group 2 = subjects 3,10 attended by A

subject group 3 = subjects 1,8 attended by A,C

etc.

C- *Room Group:* rooms of identical capacity and specifications are grouped into one room group. Each is associated with a subject group.

d- conflict matrix is drawn to represent the conflicts between the various subject groups.

So, this method tries to control the complexity of the problem.

3.2.2 Lagrangean Relaxation Method [FISH81]

A solution method based on Lagrangean relaxation coupled with subgradient optimization is used on the school timetabling problem. The method also incorporates a branch and bound procedure which takes advantage of special ordered sets of variables. It was applied on a large timetabling problem involving 900 subjects. This method maintains the effect of the constraints by giving them a 'price' and then applies that to any of the other methods. Two problems were encountered during computation. The first is large storage requirement, and the other is long computational time.

3.2.3 Flow Methods [WERR71]

The method constructs timetables by successive steps with the timetable being prepared one period at a time. Each step consists of the allocation of meeting to a determined period of the week. When we have completed the allocation for the k first periods, we call the allocation of the remaining meetings to periods $k+1, k+2, ..., n$.

A program has been written in Fortran IV to solve the problem. A series of experiments was carried out with program. It was not always possible to avoid failure in constructing the timetables.

3.2.4 Improving Infeasible Timetables [AUST75]

This method describes a way of improving infeasible timetables. It reduces the teaching resources, breaks and spreads infeasibilities in three stages. The first of these involves the solution of a series of capacitated transportation problems and is used when an initial timetable is not given. Under the limitations imposed by actual timetables this stage may be simplified. The other two stages involve solving a series of small integer programming problems which are called interchange problems, and they determine the movement of entries within the timetable. Such a method can handle fixed and block meetings, sets, allocation of special rooms and variable teacher availability while producing an acceptable spread of repeated meetings. The method has been programmed in Fortran and run on CDC 6400 computer.

3.3 Formulation as a Coloring Problem

It has been illustrated earlier that a number of methods have been used to solve the examination scheduling problem. In this work the Graph Coloring heuristics are used to solve this examination scheduling problem. A student course registration file is made. A course conflict table is built using the information in the course registration file. This table is used to build the course conflict graph. The graph is then colored with a minimum number of colors.

The registration system described earlier is designed with the course

examination problem in mind. Data is collected and sorted so that related data fields are put in the same files. This process saves a lot of redundancy of data, in other words it is normalized. One of the physical files is the student course registration file.

The student course registration file is to be used to build the course conflict table GRAPH(N,N), where N is the number of the offered courses. The building of the graph is illustrated as follows:

S1, S2 are students

C1, C2, C3, C4 are courses

Graph (4,4) is two dimensional table all of whose elements are initialized to 0.

S1 registers for C1 and C2, this is represented by adding 1 to both conflict graph (table) C1-C2 and C2-C1 as shown in the table below. This means that we connect vertex C1 with vertex C2 by an edge E1. If two courses (vertices) are connected then they have a conflict. A "1" is placed on the edge E1 to represent one student conflict.

GRAPH	C1	C2	C3	C4
C1	0	1	1	0
C2	1	0	0	0
C3	1	0	0	0
C4	0	0	0	0

S2 registers for C2, C4, and C1.

Then one is added to the graph (table) C2-C4 and C4-C2, one to the graph C2-C1 and C1-C2 and one to C1-C4 and C4-C1 as shown below:

GRAPH	C1	C2	C3	C4
C1	0	2	1	1
C2	2	0	0	1
C3	1	0	0	0
C4	1	1	0	0

Now we can see that C1 conflicts with C4 by one student and with C2 by two students. C4 conflicts with C2 by one student. Counting the number of conflicts between each two courses can be useful when using the coloring

heuristics. The following algorithm is used to build the conflict table in the way described:

```
build student-list
for each student do
    build student-course-registration-list
    J = 1
   for each course in the-student-course-registration-list do
      J = J + 1
    For I = J to the-number-of-courses-in-the-student-registration-list do
            graph (course(J), course-list (I)) =
                    graph (course(J), course-list (I)) + 1
     end
    end
  end
```

It is time now to color the graph. The graph will have N vertices, where N is the number of offered courses. For each conflict between two courses, an edge between the two vertices, representing the two courses is drawn. The number of students that have the conflict between these two courses can be seen as a weight of the edge as shown earlier.

Graph coloring heuristics are now to be applied on the graph to come up with a coloring. The best algorithm is considered using the following aspects.

A - Least number of colors used for the graph: this minimizes the examination time slots.

B - Speed in coloring the graph: this produces fast results giving the opportunity to apply more constraints.

C - Algorithm simplicity: this helps in the enhancements of the system.

D - Algorithm size: this is enable the algorithm run on a personal computer.

CHAPTER 4

COLORING HEURISTICS

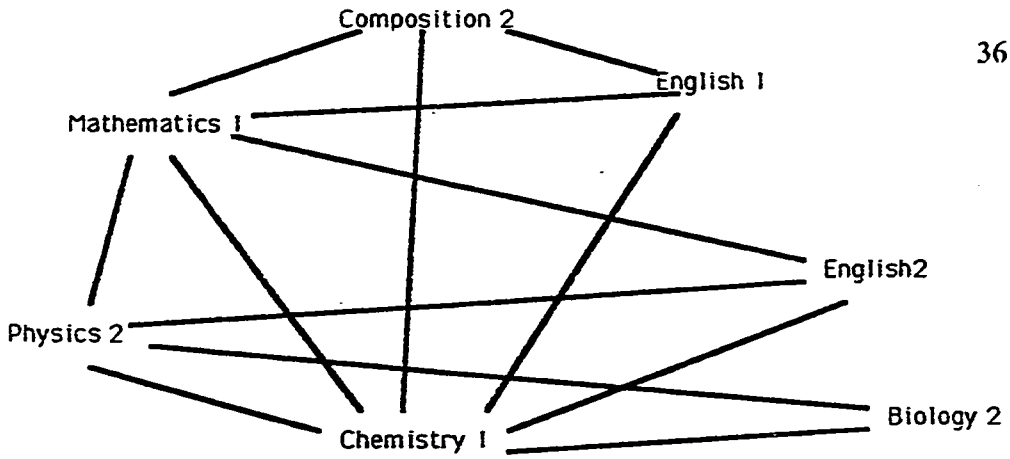
This chapter covers the description of five graph coloring heuristics. These five heuristics use the registration conflicts graph, GRAPH, built earlier to produce the least number of colors needed to color the graph. The heuristics should not color any adjacent vertices with the same color. The five heuristics are explained in this chapter with an illustrative example for each.

4.1 Greedy Heuristic

This is a very simple heuristic. It picks up a vertex and gives it the smallest color that is not used by its neighbours. The process is repeated for all the vertices. A list of vertices and their colors is then printed. If the number of colors exceeds the number of time slots the solution is rejected. This means that this heuristic does not produce an adequate solution.

The following example illustrates the heuristic:

Assume the shown graph is to be colored:



1. One of the vertices is picked, say, Mathematics1 and given color 1.
2. Any one of the remaining vertices is picked, say, Composition2. It is then given the smallest color that is not given to any of its neighbours. It cannot have color 1 since it is a neighbour to Composition2 so it is given color 2.
3. The same process is followed for English1, English2, Chemistry1, Biology2 and Physics2 given the following table:

VERTEX	*COLOR*
Mathematics1	1
Composition2	2
English1	3
English2	2
Chemistry1	4
Biology2	1
Physics2	3

The algorithm is as follows:

ALGORITHM

Load conflict graph

Initialize vector color of size n, where n is the number of vertices, to zero.

for i ← 1 to n **do**

begin

GET the adjacent colors to vertex (i)

Color [i] PICK first missing color

end

print color

---

where n is the number of vertices. Then from the above we can see that the algorithm has a time complexity of $O(n^2)$.

4.2 Independent Set Heuristic

This algorithm does the following:

1. Builds the largest Independent-set. Each set contains the vertices that are not related to each other pairwise. These vertices can be colored by the

same color.

2. Colors each vertex of the Independent-set by *color*1.

3. From the remaining vertices in the graph, finds the largest Independent-set.

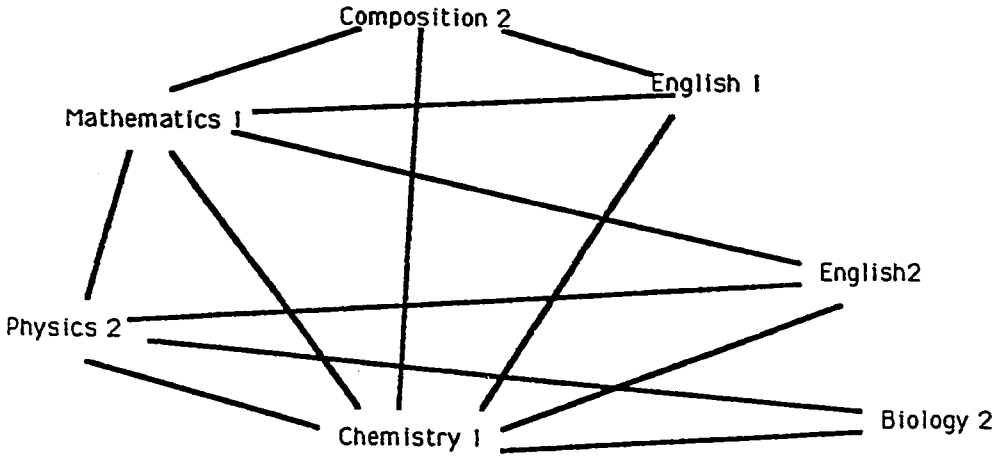
4. Colors each vertex of the Independent-set in *step* 3 by the next color.

5. Repeats *step* 3 and 4 until all vertices are colored.

6. Writes the list of vertices and their colors.

The following example illustrates the heuristic:

Assume the shown graph is to be colored:



1. The heuristic picks the vertex with the highest degree. In this graph it is Mathematics1.

2. Then it picks the next highest degree vertex that is not a neighbour to Mathematics1. In this graph it is Biology2.

3. It repeats step 2 until no more vertices exist. Each time the algorithm picks a vertex it makes sure that the vertex is not connected to any of the Independent-set vertices. In this case the set picked is (Mathematics1, Biology2) it is not the largest Independent-set. This shows that the heuristic (tries) to build the largest Independent-set but does not always succeed.

4. Mathematics1 and Biology2 are given color1. The heuristic is resumed on the rest of the vertices to build the next Independent-set. In this case it is English1 and Physics2.

5. This process is repeated until there are no more vertices to be colored. Every time a vertex is picked the degrees of neighbour vertices are adjusted.

Finally the following table will result:

VERTEX	*COLOR*
Mathematics1	1
Biology2	1
English1	2
Physics	2
Composition2	3
Chemistry1	3
English2	3

The algorithm is as follows:

Construct $X_1, X_2, \ldots X_{k-1}, X_k$ such that each X_i

contains vertices that can be colored by the same color

for i ← **to** k-1 **do**

begin

$X_i \leftarrow 0$

V ← G(V) - $U_{j=1}^{i-1} X_i$

while V < > 0 **do**

begin

pick a vertex u ∈ V with the minimum number of

neighbours in V.

$X_1 \leftarrow X_i U \{u\}$

$V \leftarrow V - \{u\} - \{$ neighbours of u $\}$

end

end

$Xk \leftarrow G(V) - U_{i=1}^{k-1} X_i$

This algorithm has the time complexity of O (nK^2) where K is the number of Independent-sets that are in the graph.

4.3 No-Choice Algorithm

The algorithm does the following:

1. Finds the largest set of vertices that are connected to each other, in a form that each vertex is connected to all other vertices in the set. This set is the clique. This process uses a greedy heuristic.

2. Color each vertex of the clique by a different color.

3. Some of the remaining vertices in the graph must be colored by a specific color. These vertices are called the No-choice vertices. These vertices are assigned to their No-choice colors.

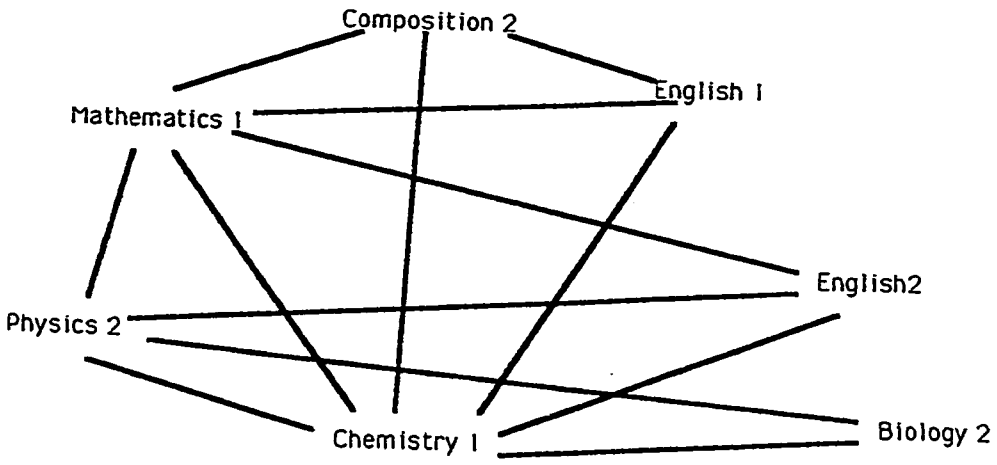
4. If there are still No-choice vertices then the algorithm repeats the previous step until there are no more No-choice vertices.

5. For the remaining vertices the algorithm picks a color for one of them.

6. If there are still some vertices the algorithm repeats steps 3, 4 and 5 until there are no more vertices.

The following example illustrates the heuristic:

Assume the shown graph is to be colored:



1. The heuristic build the largest clique, say (Mathematics1, Physics2,

Chemistry1, English2), and gives each vertex in it a unique color, say 1,2,3,4.

2. If the algorithm has to color the graph with only four colors then Composition2 must be colored with color 4. It is said to be a No-choice vertex.

3. It continues to color all No-choice vertices until no more vertices to color.

4. If there are vertices remaining that are not No-choice vertices pick one of them and repeat the process.

Finally the following coloring would be produced.

VERTEX	*COLOR*
Mathematics1	1
Physics2	2
Chemistry1	3
English2	4
Composition2	4
English1	2
Biology2	1 or 4

The algorithm is as follows: [TURN 88]

ALGORITHM NO-CHOICE

for all vertices $x \in X$ **do**

begin

Avail (X) = {*all#colors*}

end

Build a maximal clique, X.

Assign a unique color to every vertex of the clique X

for $x \in X$ **do**

begin

for $y \in$ *neighbours* (x) **do**

begin

avail $(y) \leftarrow$ *avail* $(y) -$ *color* (x)

if y has only one choice **then** Q $\leftarrow$ Q U { Y }

end;

end;

while $Q \neq [\,]$ **do**

```
begin
        X ← next vertex in the queue
        if |avail (x) | ≠ 1  then return false;
        color (x) ←min (avail (x))
        for y ∈ neighbors (x)  do
        begin
                avail(y)←  avail(y) − color (x)
                if y has only one choice then Q ← Q U { Y }
        end;
end;
```

This algorithm can be implemented to run in 0 (n + m log k), where n is the number of vertices and m is the number of edges and k is the size of the clique [TURN 88].

4.4 The Brelaz Heuristic

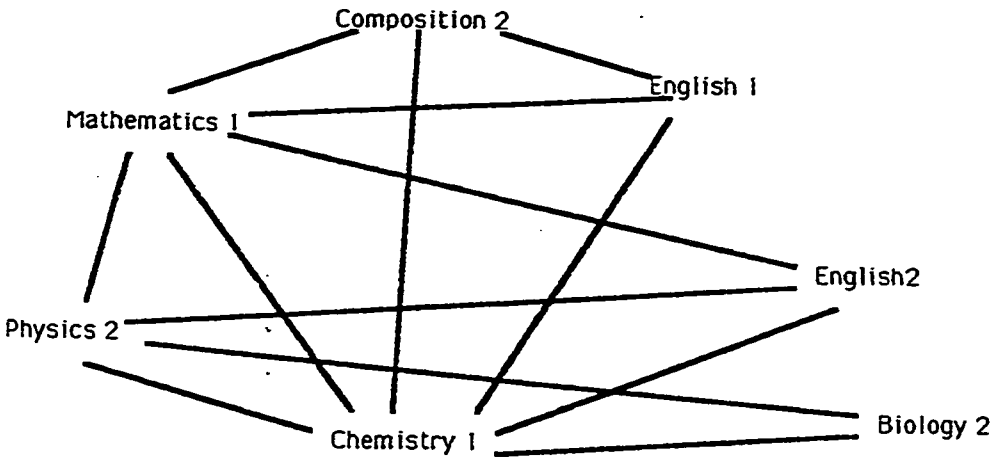
This algorithm does the following:

1. Builds the heap having the smallest degree vertex to be at the root.
2. While the heap is not empty, it picks the root and colors it with the smallest available color.

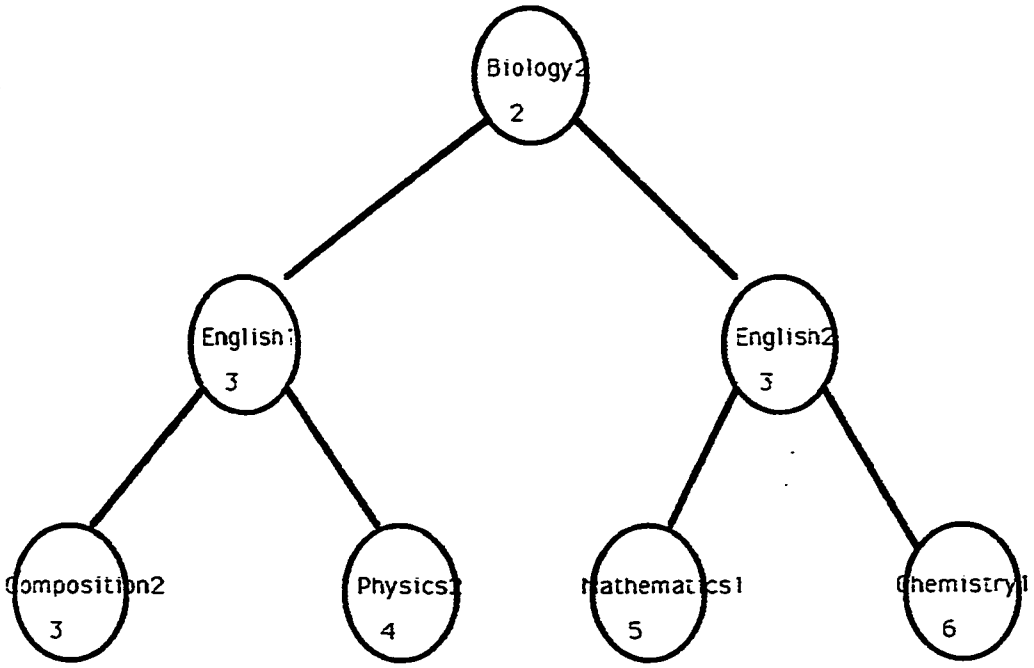
3. It decreases the degrees of the neighbour vertices and then for each one it adjusts the heap.

The following example illustrates the heuristic:

Assume the shown graph is to be colored:



1. The heuristic builds the heap to be as follows:



2. Biology2 is colored with color1 and the heap is adjusted after its neighbours' degrees are decremented.
3. It repeats this process of coloring with minimum available color until no more vertices remain to be colored.

Finally the following table results:

VERTEX	*COLOR*
Biology2	1
English1	1
English2	1
.	.
.	.
.	.
.	.

The algorithm is as follows [TURN 88].

ALGORITHM BRELAZ

```
for x ∈ { 1....n} do
begin
        Initialize colors of all vertices to zero;
        avail (x) ← { 1, . . .,n} ;
end;
Build the heap H of vertices (1, . . .,n) based on their degrees
while H ≠ { } do
begin
        Delete top of the heap (H)
        color (x) ←  min (avail color);
        for y ∈ neighbors (x) do
        begin
                if color (y) =0 then
                begin
                        avail (y) ← avail (y) − c (x) ;
                        deg(y) ← deg(y) − 1 ;
                        Adjust the heap (H) for vertex y;
                end;
        end;
end;
```

end;

This algorithm can be implemented to run in 0(m log n), where n is the number of vertices and m is the number of edges [TURN 88].

4.5 The Random-Clique Algorithm

This algorithm is similar to the No-choice algorithm with the following differences:

-When this algorithm picks the first vertex to build the clique around it, it randomly picks it up from a set of high degree vertices. This gives the ability to build different cliques from different starting points.

-It does the same thing every time it picks a vertex to be added to the clique. This gives the ability to change the composition of the clique being built.

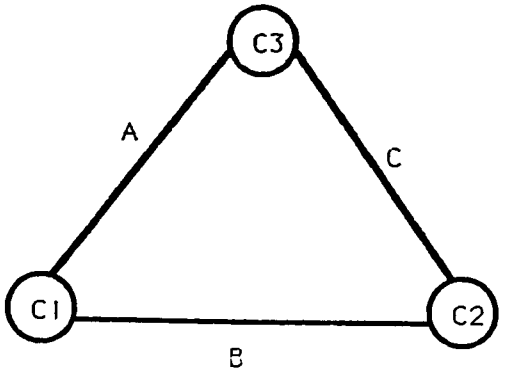
CHAPTER 5

ENHANCEMENTS

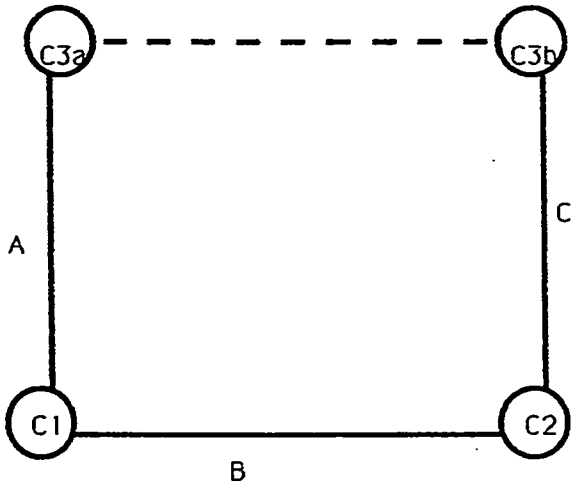
This chapter discusses some enhancements to the algorithms covered in the previous chapter. These enhancements are: the splitting of the nodes, preassignment of colors to some of the nodes, priority and ordering of vertices to be colored and finally spilling nodes. This chapter specifies how each of the previously mentioned enhancements affect each algorithm, discussed in the previous chapter.

5.1 Node Splitting

When a graph cannot be colored by a preassigned number of colors, one of the nodes that cannot be colored (A) can be split into two nodes ($A1$, $A2$) so that about half of the $A's$ neighbours are assigned to be neighbours of ($A1$) and the rest as neighbours of ($A2$). That is similar to splitting a course that cannot have its examination at a certain period into two sections so that some of the students can have their exams in one period and the others in the other period. For example, if student A attends course $C1$ and $C3$, B attends course $C2$ and $C1$, and C attends course $C3$ and $C2$. Then the following is the conflict graph.



If we have only two colors then we cannot color this graph. If we split node *C*3 into two nodes (course *C*3 into two Sections) then the following graph can be colored with two colors as shown below. The nodes split from the same original node are connected with a (dashed) edge to make sure that they do not get the same color.



Greedy Algorithm:

In this algorithm simply the first vertex that runs out of available colors before it is colored is the candidate for splitting. This process is to repeated until the graph is colored with the specified number of colors.

Independent Set:

After the *k-th* Independent-set is colored, where *k* is the maximum number of colors available, the remaining vertices are split one at a time. The coloring algorithm is run after each split. This new initiation could give different coloring

because the split vertex is connected to at least one of the Independent-set vertices.

No-Choice:

In this algorithm the vertices can run out of colors if the clique size is larger than the number of colors or when these are two connected vertices. In the case of the clique the node splitting algorithm splits the first vertex in the clique. In the other case it splits the first vertex that cannot be colored.

The Brelaz:

In this algorithm the first root that cannot be colored is split. The neighbour vertices adjust their degrees and the heap is adjusted. The whole process is repeated until the heap is empty.

The Random Clique:

This is very similar to the clique problem though the split vertex (might) not be as helpful as it is with the case of No-choice. That is because this algorithm is non-deterministic.

5.2 Preassignment of colors

Some of the courses are requested to be held at specific times. These courses

are given preassignment periods (colors). For example, a difficult course is to be scheduled on the first day of the week to give the student weekend to prepare for it. Two courses need a specific room might be assigned two different colors to make sure that they are not assigned the same time period. Therefore every time the algorithm wants to color a vertex it has to make sure that vertex is not neighbour to one of the preassigned vertices that has the same color.

5.3 Ordering and Priority

Some vertices have higher priority to be colored for a number of reasons. One of these reasons is that if the graph can not be colored by the specified number of colors then one makes sure that the high priority vertices are already colored. Different ordering of priority might give different results as will be described in Chapter 6. For the time being the ordering added is picking the vertices with respect to their degree. The Brelaz algorithm and Independent-set algorithm have that priority in their original procedure. The No-choice algorithm has it only in the clique building. It is added to it in two places, the first one while selecting one of the No-choice vertices. The second time while selecting one of the No-choice vertices. The second time while selecting the next vertex to be colored among the remaining uncolored vertices which are not No-choice vertices. The same can be said for the Random-clique. The simple Greedy algorithm is modified to adapt the priority changes in picking the next vertex to be colored.

5.4 Node Spilling

Spilling a node means removing it from the graph to be colored. Such removal will simplify the graph by reducing the number of nodes by one and the number of edges by the number of edges connected to the node. Turner discussed the spilling concept and emphasized three issues that should arise when spilling method is added. The cost benefits of deleting a specific node, the spilling should not wait until the coloring algorithm reaches a block, and the running time should be reasonable [TURN 88].

In conclusion, spilling is made on nodes if their removal has best effects on the graph coloring. This can be identified by having high degree and least loss which can be identified by the least number of students represented by these edges.

CHAPTER 6

EXPERIMENTAL RESULTS

This chapter describes the testing system, the results of the coloring algorithms, and the effect of node splitting on the results. Results are made into tables and graphs. Comparison analysis between the algorithms is made. A detailed analysis of each algorithm is shown. In the next Chapter this analysis is used to develop a hybrid algorithm.

6.1 Testing System

An automated system is developed to test the coloring algorithms. The system consists of three parts. The first part generates test files. Files that simulate the conflict graphs of examination scheduling. Different types of simulations are made for each graph size and complexity ratio. The second part runs the different algorithms against the conflict graphs produced in first part of the testing system. Test results are produced into a formatted file. The last part averages the results of the graphs of same number of nodes and ratios and places each average in a record of the results file. This testing system is implemented as a shellscript under UNIX.

The first part of the testing system simulates the examination scheduling conflict graph building. It reads the specification of the school it should simulate and produces its conflict graph. It reads the number of nodes which represents the number of courses offered, the complexity ratio of the nodes and produces the conflict graph. For each specification a number of graphs are made. This is done to obtain robust experimental results by averaging their results. The number of graphs for each graph specification is placed in the first record. A number of graph specifications are build in each run. These graph specifications have different number of nodes and complexity ratios. The number of graph specifications contained in the produced file is placed in the second record. Each conflict graph specification is shown by two numbers. The first number shows the number of nodes in the graph and the other shows the complexity ratio.

The first part of the testing system, the graph generation part of the system, builds the conflict graph randomly. It picks up two numbers randomly, say *A* and *B*. Each of these numbers represents a node. If these nodes were not connected from the previous tries, a '1' is placed in the conflict graph, say GRAPH [A,B]. This '1' represents the drawing of an edge between these two nodes. The number of edges that should be placed in the graph is calculated using the formula:

$$Edges = Ratio * Nodes * Nodes/2$$

This number is calculated as soon as the complexity ratio is read into this part of the system. This number is decremented every time an edge is added to the

graph. When the number reaches zero, this means that the desired number of edges are placed. This is repeated for the same graph specification as required in the first record, to provide different random graphs with the same specification. The previous work is done for a number of conflict graph specifications.

The conflict graphs produced by the first part of the system are read into the second part of the system, the coloring part. For each algorithm different tests are conducted. The first test finds the best coloring the algorithm can produce for a specific conflict graph. The second test finds the CPU time required to produce a coloring using a preassigned number. This number of colors is a number that the algorithm can color the graph with. The third test finds the effect of node splitting on the graph. This process is conducted for every graph produced by the first part of the testing system.

The results of the tests are placed into a file, each record of this file represents the three tests on a single conflict graph by one algorithm. Each record contains the number of nodes of that conflict graph, its complexity ratio, the algorithm tested, the minimum number of colors required to color the graph, the CPU time required for a coloring by a selected number of colors (K), K, where K is selected so that all algorithms can use it to color the graphs. It is greater than or equal to the minimum number of colors required of all algorithms, and the CPU time required for a coloring by a selected small number (W), W, where W is selected so that it is less than the required number

of colors required by all the algorithms. It shows the effect of node splitting on these algorithms. If K is very small then W is given the same number as K because splitting the nodes is hard. The third part of the system reads the file produced by the second part and averages the records of the same graph specifications into one record of the results file.

The three parts of the testing system are executed using shellscripts. This procedure runs the graph generation part, runs the algorithm testing part and averages the test results with the previous tests in the results file.

6.2 Test Results

The tests made produced 16 different conflict graph specifications. The conflict graphs specifications have different sizes. The sizes ranges from 50 to 250 nodes. Also the conflict graphs have different ratios. The graph complexity ratios used are 0.05, 0.1, 0.2 and 0.3. Each such graph specification is used to generate 20 different graph instances. Knowing that we have five algorithms to test, there will be 1600 different initial result records. Each group of 20 of these records resulting from the same graph specification and algorithm test is averaged into one result record giving at the end 80 result records. These aggregate results are plotted to ease their analysis.

Forty two plots were produced from the results file. These plots can be put into three major categories. The first one consists of 16 plots (A1 to A16). Each

one of these plots shows the behavior of all the five algorithms. This behavior is presented as a curve on the plot. Four of these plots show the effect of different conflict graph sizes (number of nodes) on the CPU time required by each algorithm. Each of these four plots uses one of the previously mentioned ratios as a fixed ratio to the plot. Four others show the effect of complexity ratio size on the CPU time required by the algorithm to color the conflict graph. Each plot uses one of the conflict graph sizes mentioned earlier for each plot. Four plots show the effect of the different graph sizes on the number of colors. Each plot uses one of the ratios mentioned earlier. And the last four in this category show the effect of complexity ratio on the minimum number of colors required by the algorithms to color the conflict graph.

The second category consists of 16 histograms. Each histogram represents a conflict graph test against the five algorithms using a preassigned number of colors. Each histogram has a fixed conflict graph size and a fixed ratio. The preassigned number used should force some of the algorithms to split their nodes. These graphs show the effect of node splitting from a CPU-time point of view, on the graphs that require higher number of colors.

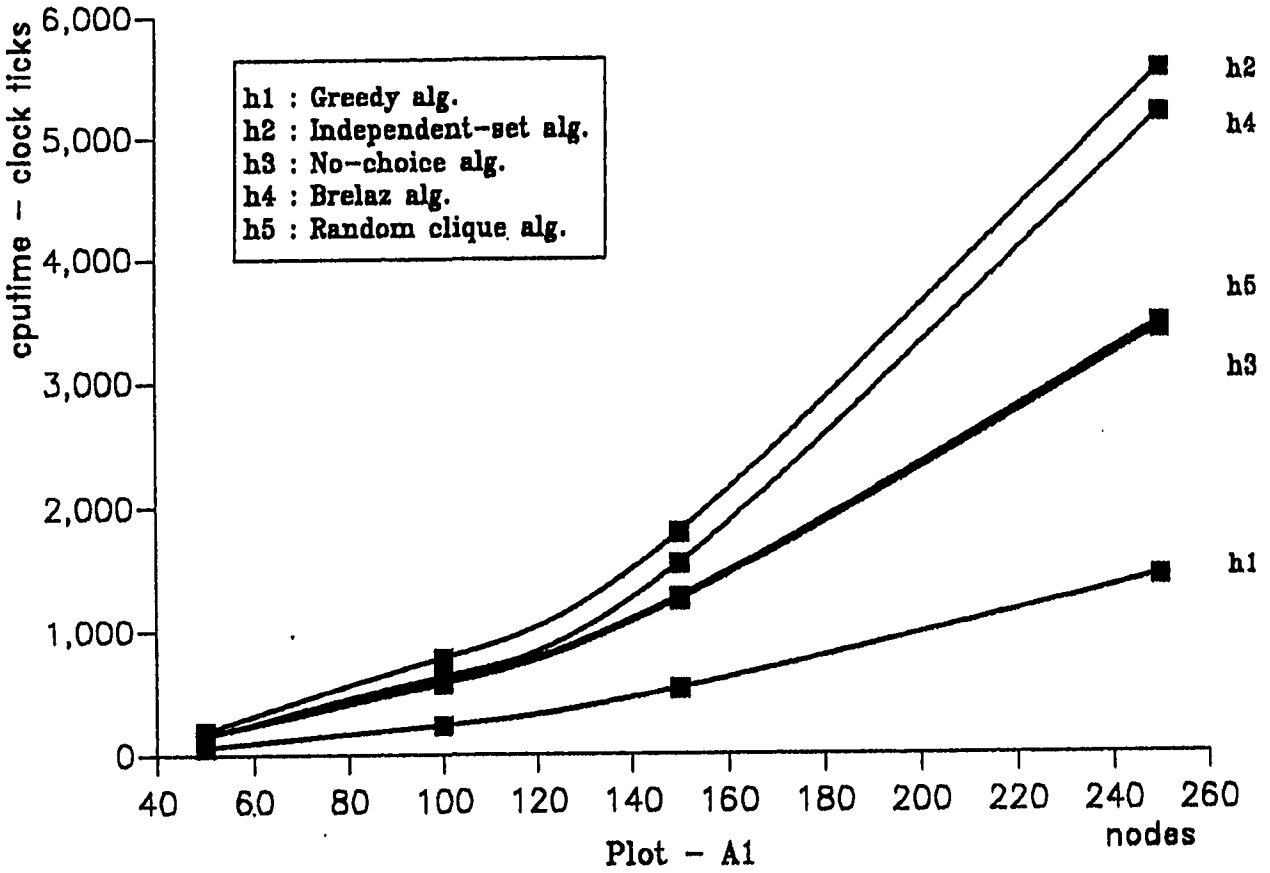
The third category consists of 10 plots. Two plots for each algorithm. One to show the effect of different conflict graph size on colors produced by the algorithm. On each plot four curves are made one for each ratio. The other shows the effect of different ratios on the colors produced by the algorithm. On each plot four curves are made one for each conflict graph size.

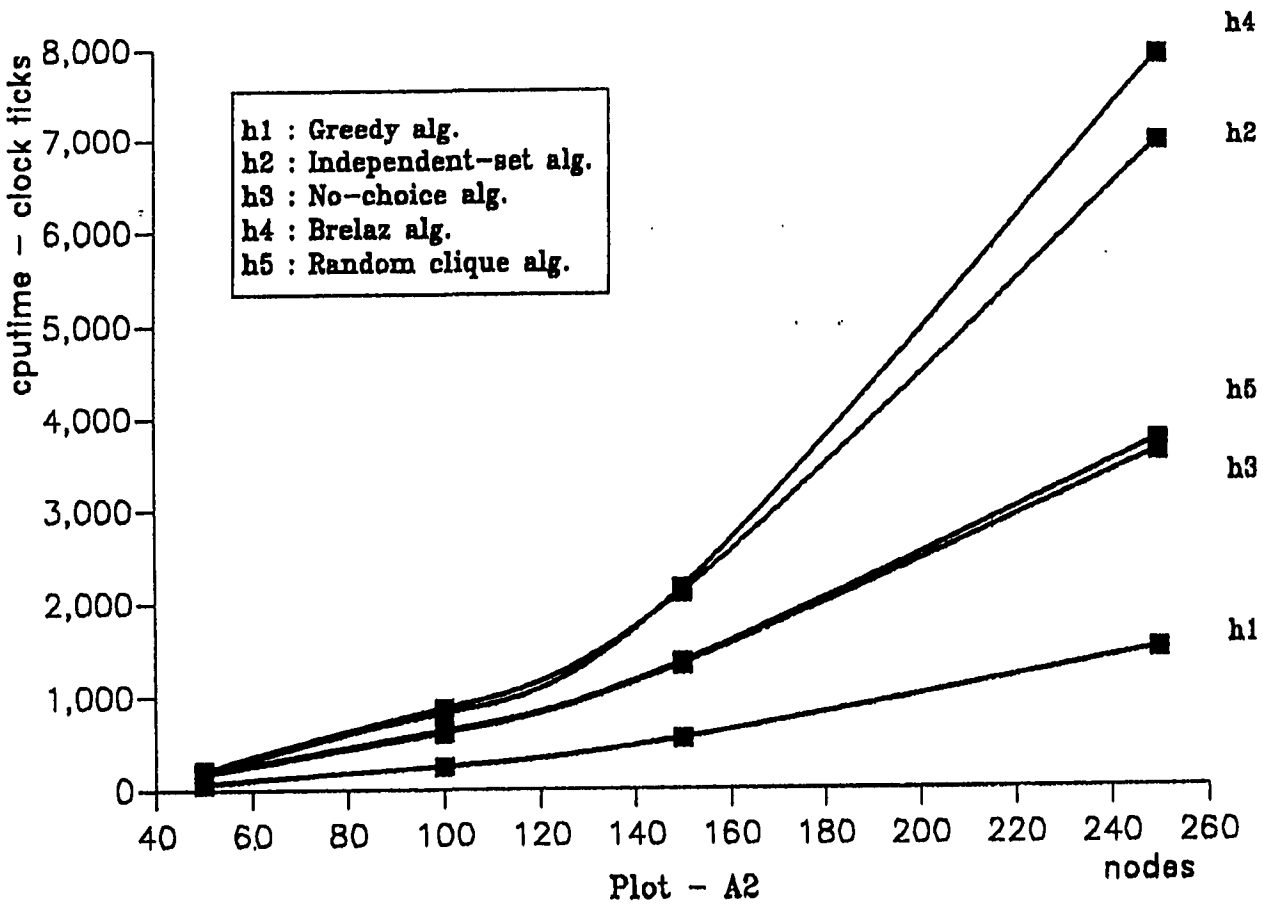
6.3 Graph Analysis

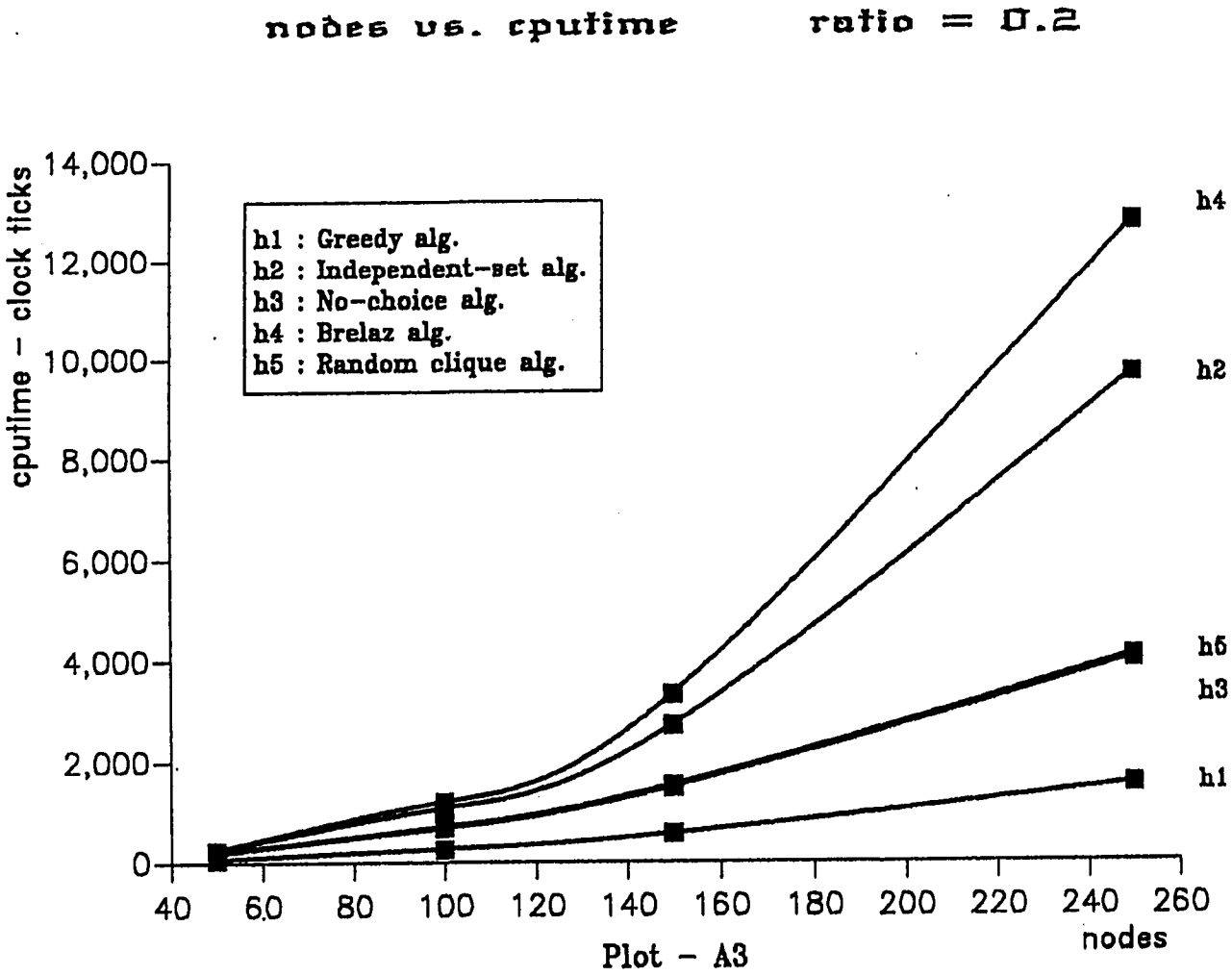
Four types of analyses are made. One is to analyze the behavior of the algorithms towards the different sizes of graphs. The second is to analyze the behavior of the algorithms with different ratios. Both of these analyses use the CPU time and the number of colors produced to measure the differences between the algorithms. The third analyses shows the different behavior of the algorithms towards a preassigned number of colors. It shows the effect of node splitting on the algorithms that require many colors. The fourth analyses is a detailed study of each algorithm towards the changing graph size and ratio.

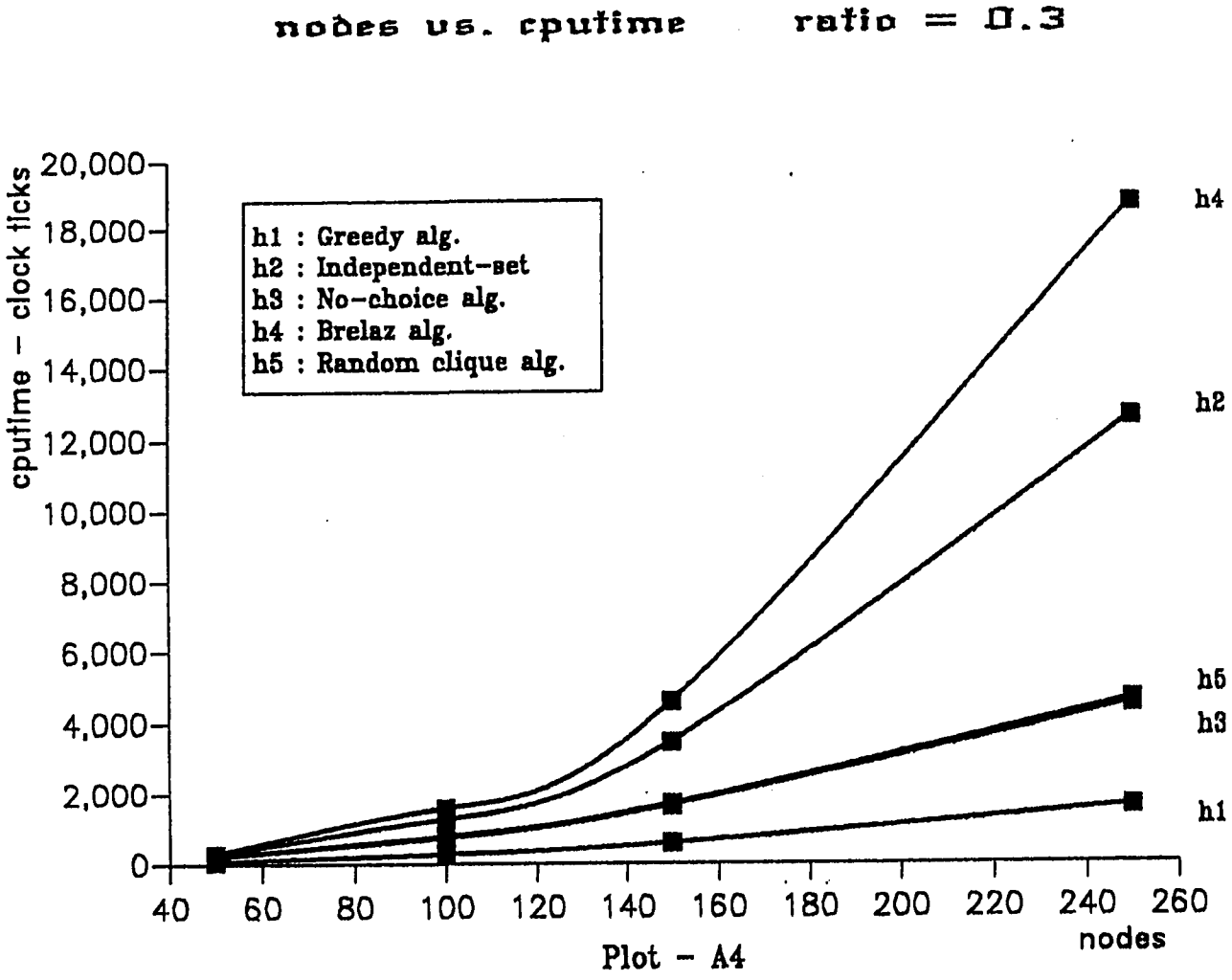
6.3.1 Effect of Size and Ratio

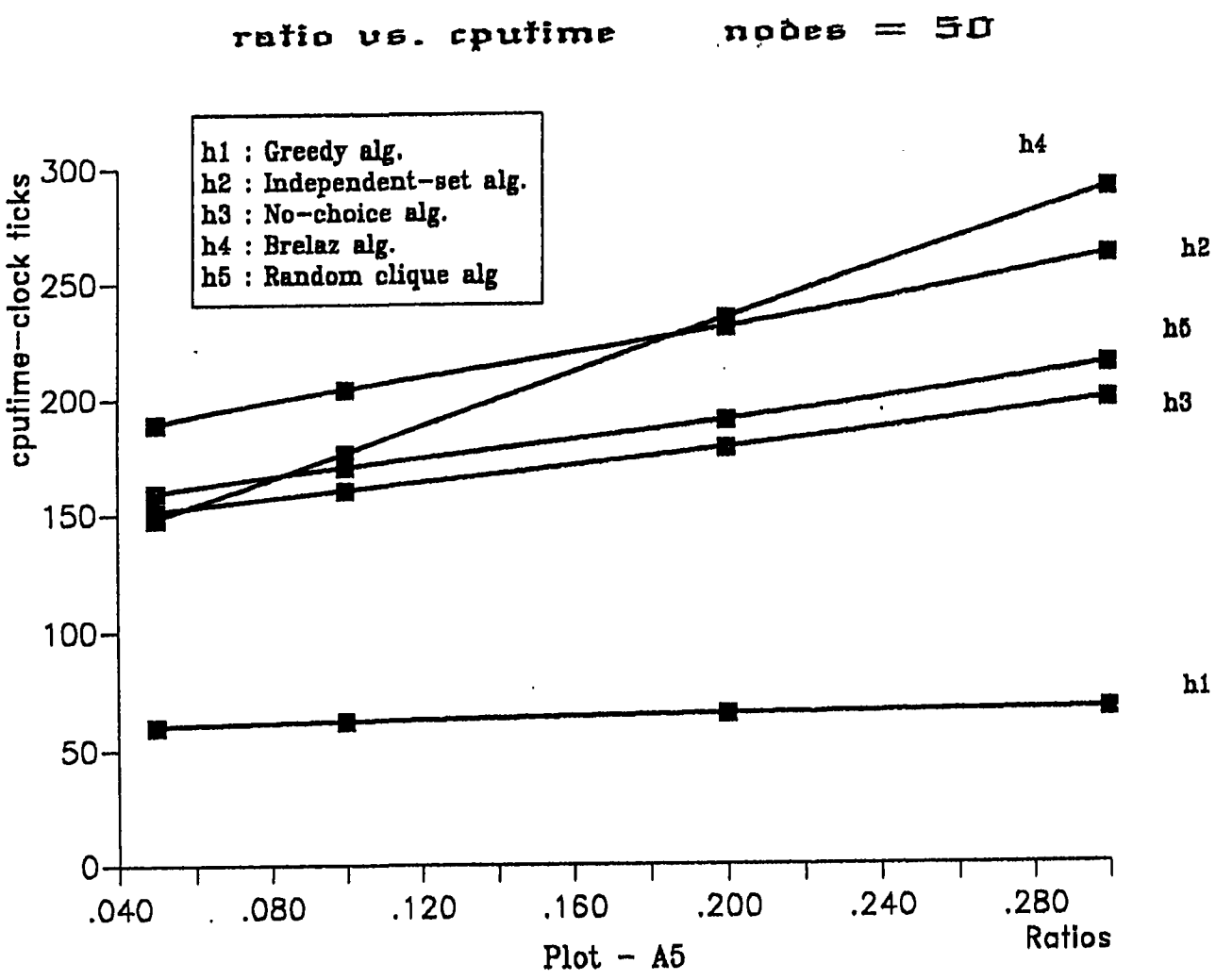
This section covers the first and second type of analyses. The other two are described in different sections. From plots A1 to A8 we can see that the greedy algorithms have always the best CPU time. It is noticeably different from the other four algorithms in that respect. As the graph size increases more difference is seen between these algorithms. The Greedy algorithm is followed by the Random-clique algorithm followed by the Independent-set algorithm and finally the Brelaz algorithm. In the Brelaz algorithm, the CPU time increases drastically as the size of the graph and ratio grow. The Independent-set algorithm is almost as poor as the Brelaz algorithm. The No-choice algorithm and the Random-clique algorithm look similar to each other. They tend to have intermediate CPU time requirements among the five algorithms tested. For very

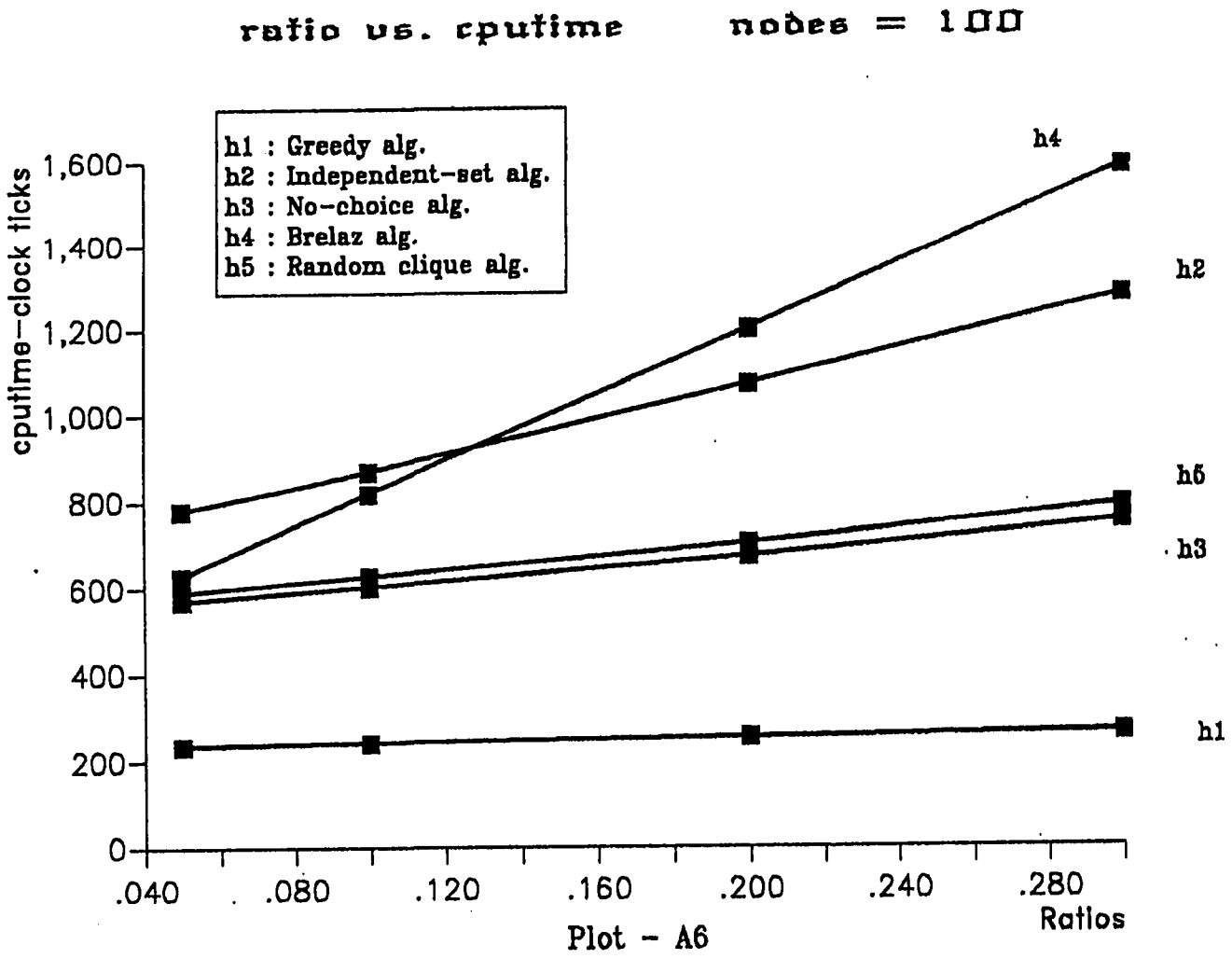

nodes vs. cputime
ratio = 0.05
cputime - clock ticks
6,000
5,000
4,000
3,000
2,000
1,000
0
h1 : Greedy alg.
h2 : Independent-set alg.
h3 : No-choice alg.
h4 : Brelaz alg.
h5 : Random clique alg.
h2
h4
h5
h3
h1
40
60
80
100
120
140
160
180
200
220
240
260
nodes
Plot - A1

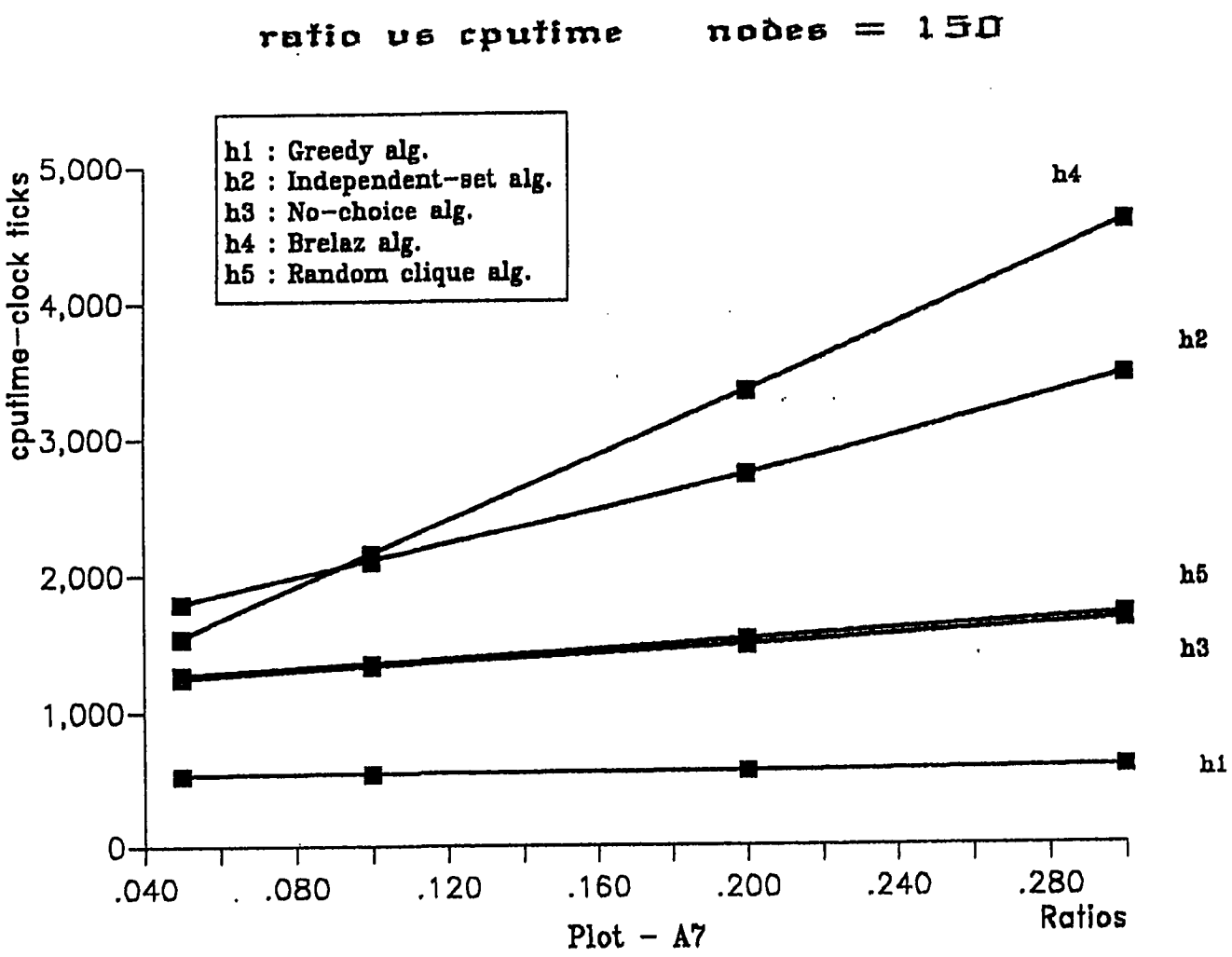

nodes vs. cputime ratio = 0.1
cputime - clock ticks
8,000
7,000
6,000
5,000
4,000
3,000
2,000
1,000
0
h1 : Greedy alg.
h2 : Independent-set alg.
h3 : No-choice alg.
h4 : Brelaz alg.
h5 : Random clique alg.
h4
h2
h5
h3
h1
40
60
80
100
120
140
160
180
200
220
240
260
nodes
Plot - A2

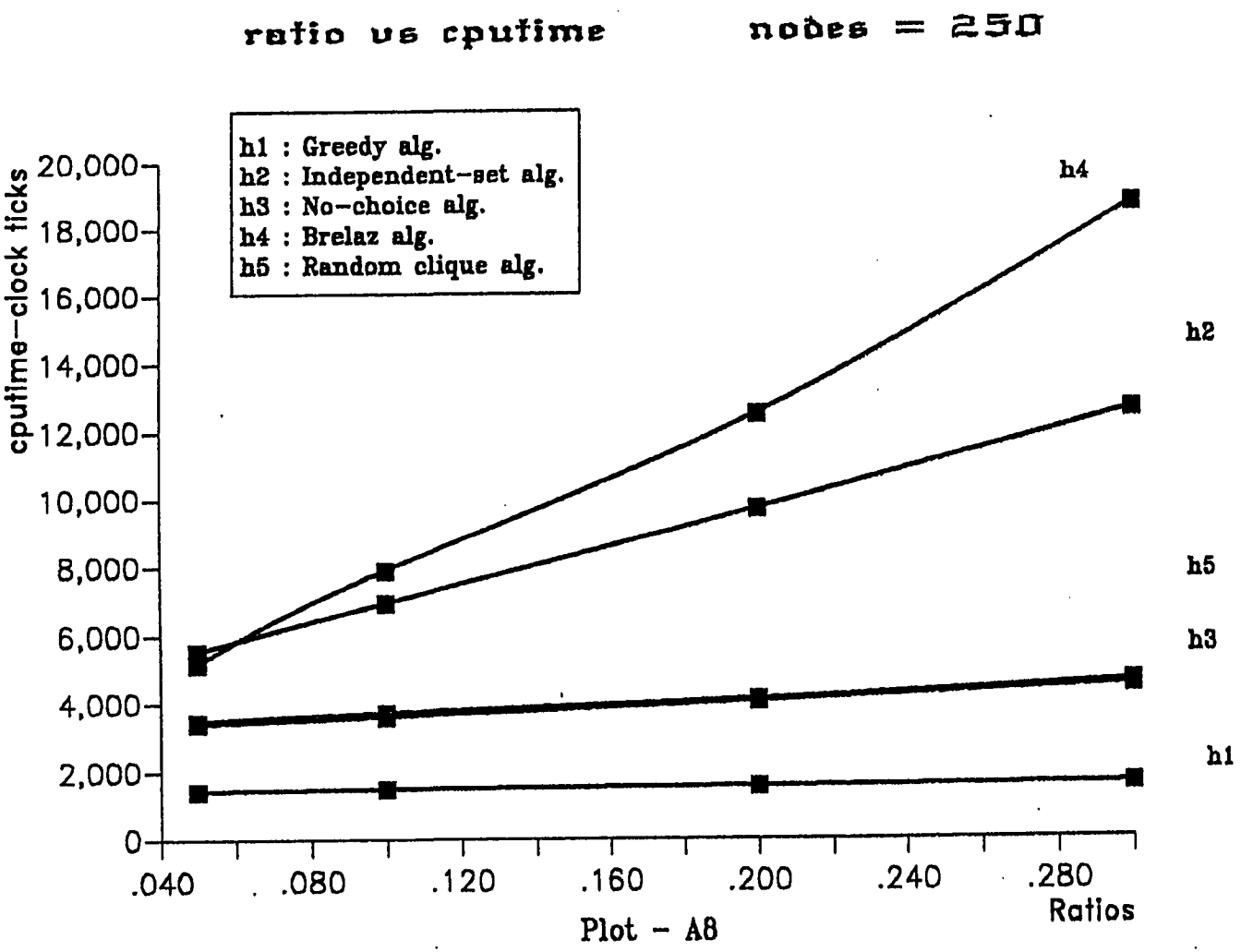

nodes vs. cputime ratio = 0.2
cputime - clock ticks
14,000
12,000
10,000
8,000
6,000
4,000
2,000
0
h1 : Greedy alg.
h2 : Independent-set alg.
h3 : No-choice alg.
h4 : Brelaz alg.
h5 : Random clique alg.
h4
h2
h5
h3
h1
40 60 80 100 120 140 160 180 200 220 240 260
nodes
Plot - A3

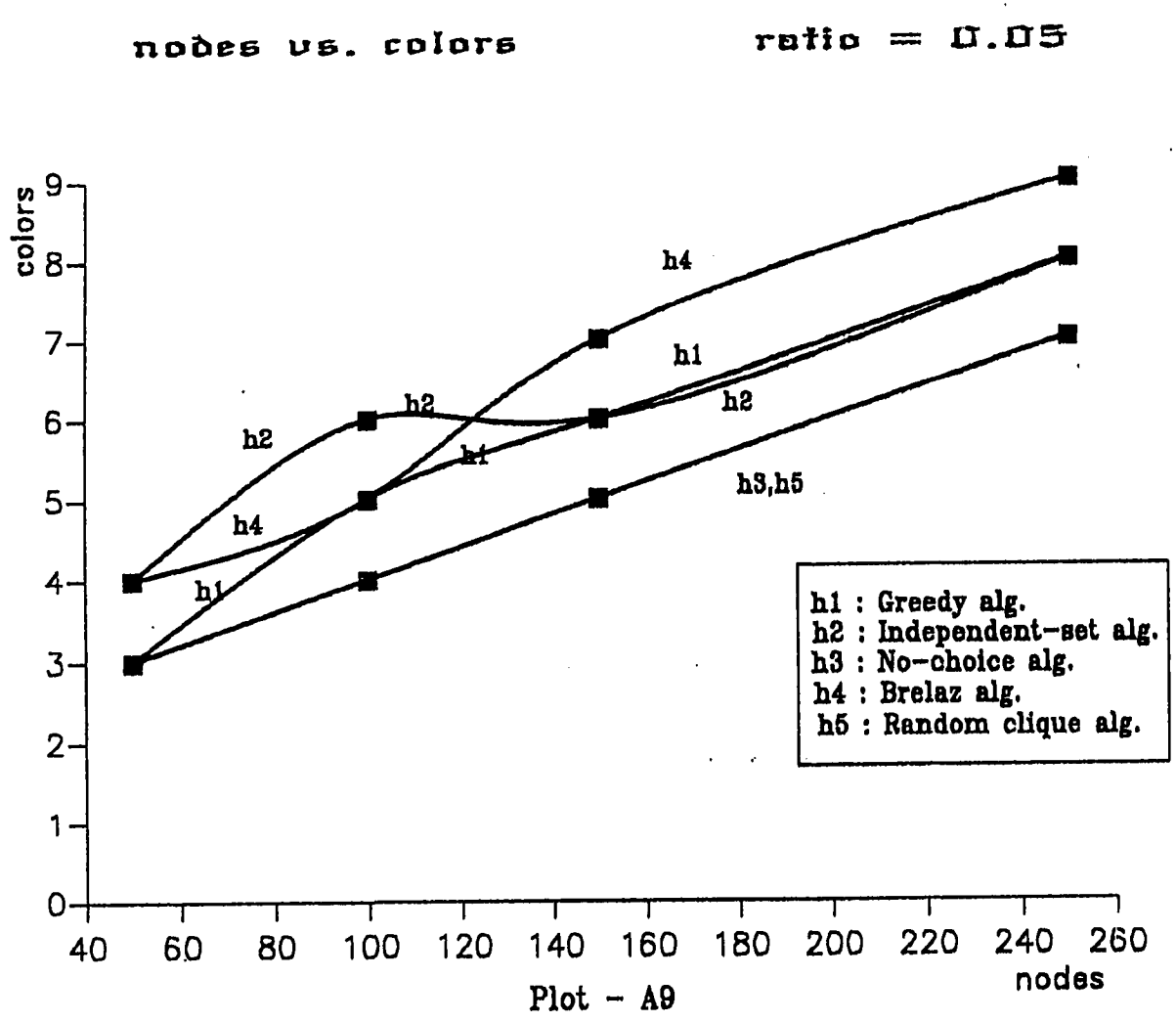

nodes vs. cputime ratio = 0.3
cputime - clock ticks
20,000
18,000
16,000
14,000
12,000
10,000
8,000
6,000
4,000
2,000
0
h1 : Greedy alg.
h2 : Independent-set
h3 : No-choice alg.
h4 : Brelaz alg.
h5 : Random clique alg.
h4
h2
h5
h3
h1
40
60
80
100
120
140
160
180
200
220
240
260
nodes
Plot - A4

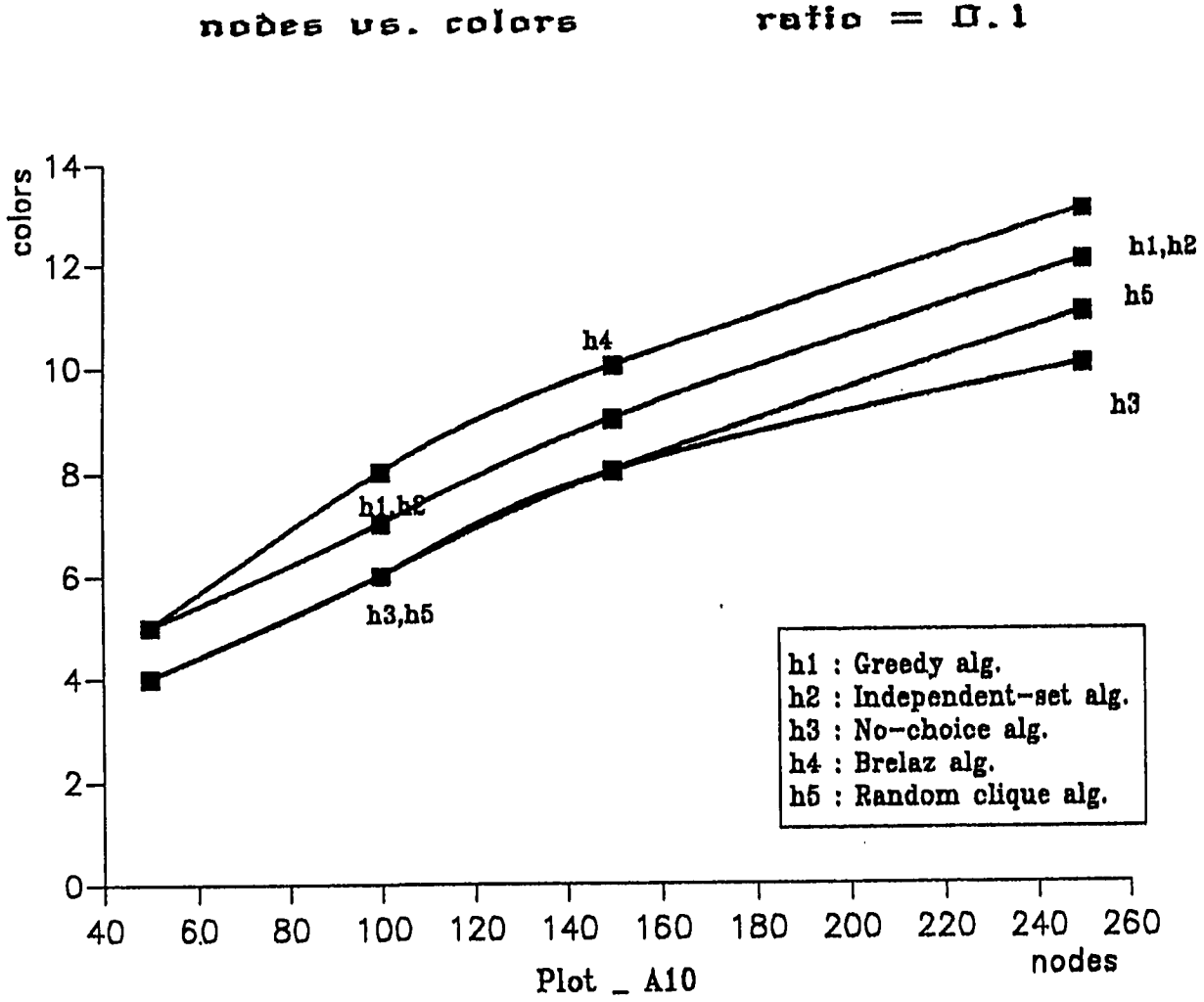

ratio vs. cputime nodes = 50
h1 : Greedy alg.
h2 : Independent-set alg.
h3 : No-choice alg.
h4 : Brelaz alg.
h5 : Random clique alg
cputime-clock ticks
300
250
200
150
100
50
0
.040
.080
.120
.160
.200
.240
.280
Ratios
Plot - A5
h4
h2
h5
h3
h1

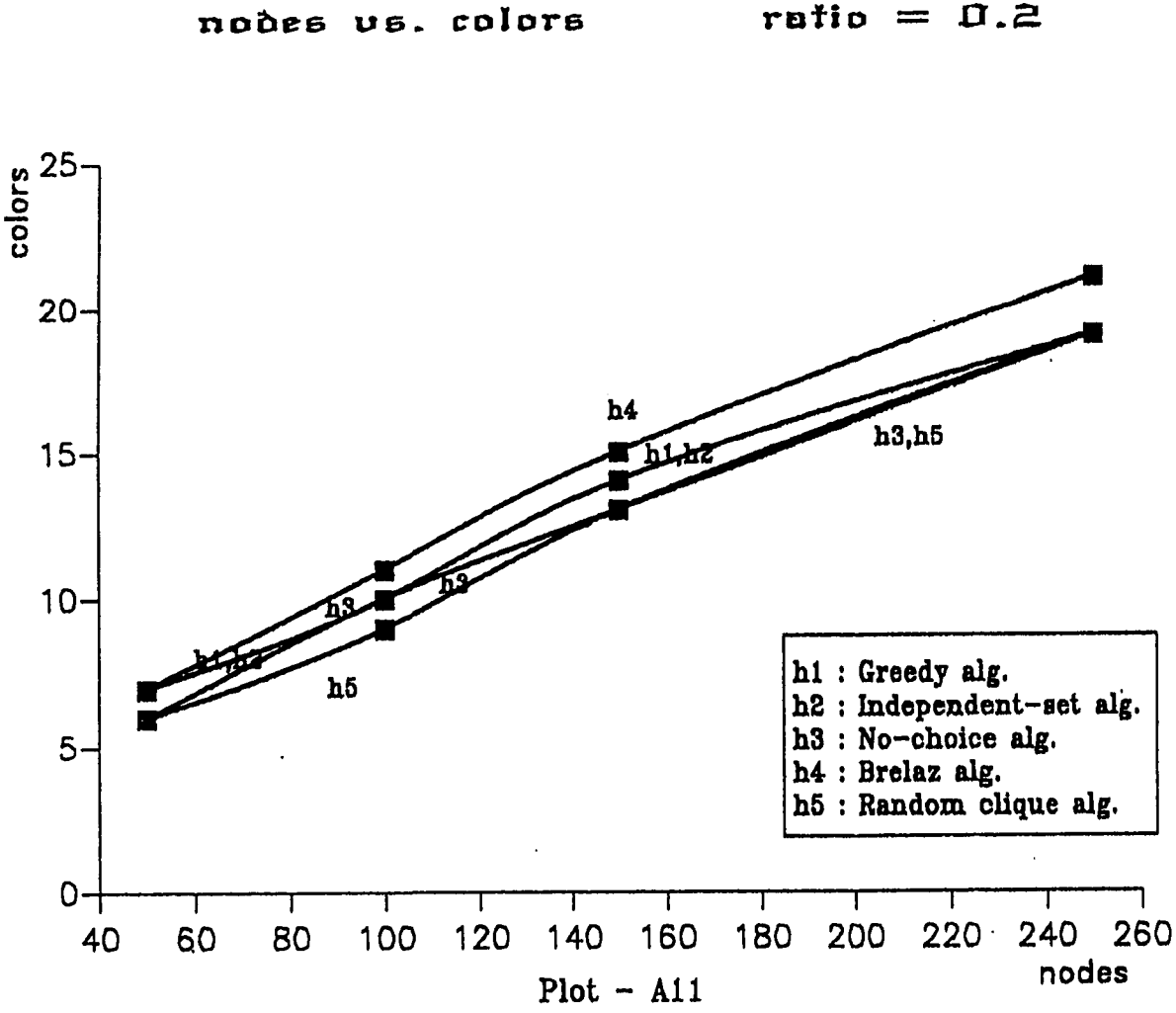

ratio vs. cputime nodes = 100
h1 : Greedy alg.
h2 : Independent-set alg.
h3 : No-choice alg.
h4 : Brelaz alg.
h5 : Random clique alg.
cputime-clock ticks
1,600
1,400
1,200
1,000
800
600
400
200
0
.040
.080
.120
.160
.200
.240
.280
Ratios
Plot - A6
h4
h2
h5
h3
h1

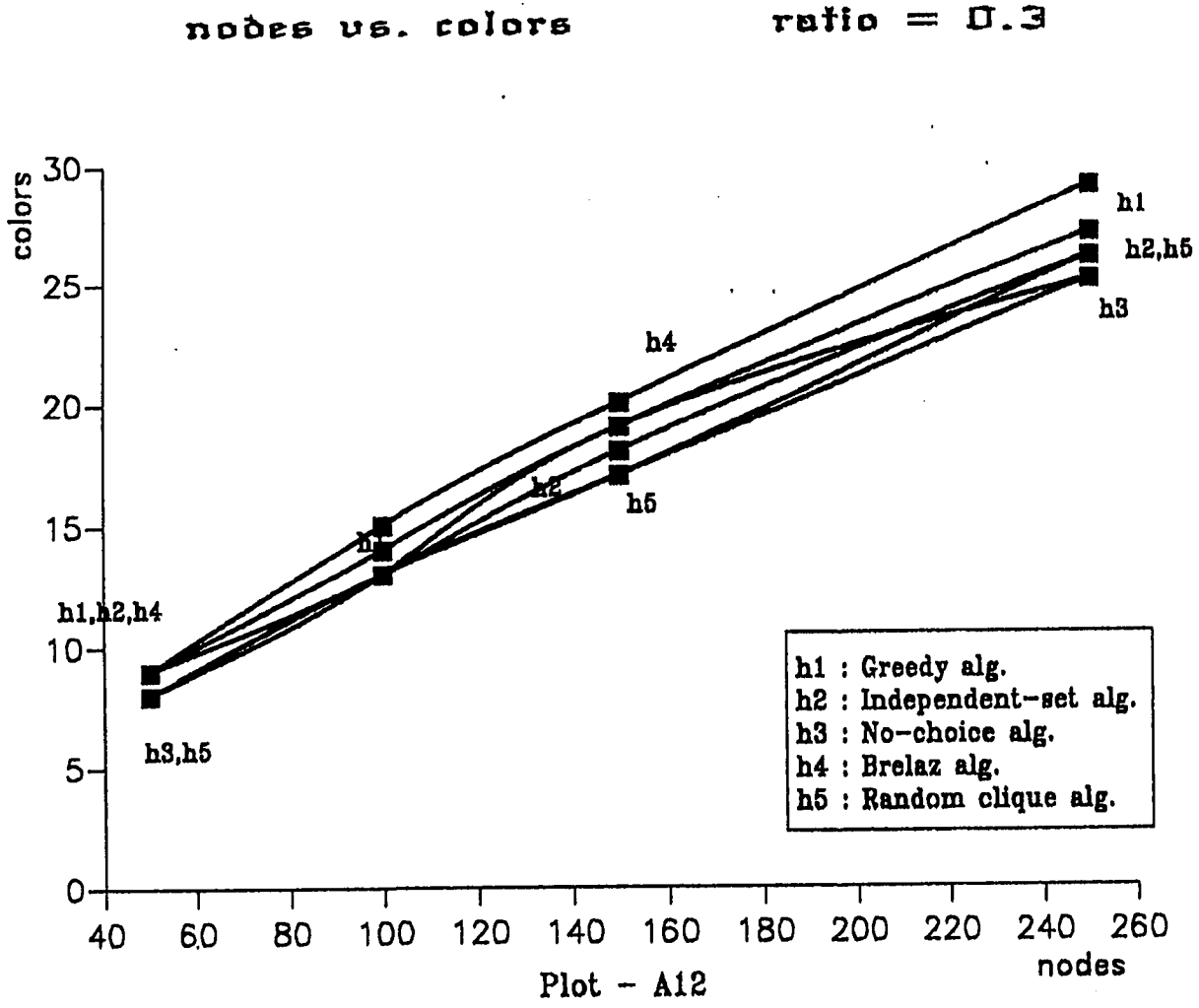

ratio vs cputime nodes = 150
h1 : Greedy alg.
h2 : Independent-set alg.
h3 : No-choice alg.
h4 : Brelaz alg.
h5 : Random clique alg.
cputime-clock ticks
5,000
4,000
3,000
2,000
1,000
0
.040
.080
.120
.160
.200
.240
.280
Ratios
Plot - A7
h4
h2
h5
h3
h1

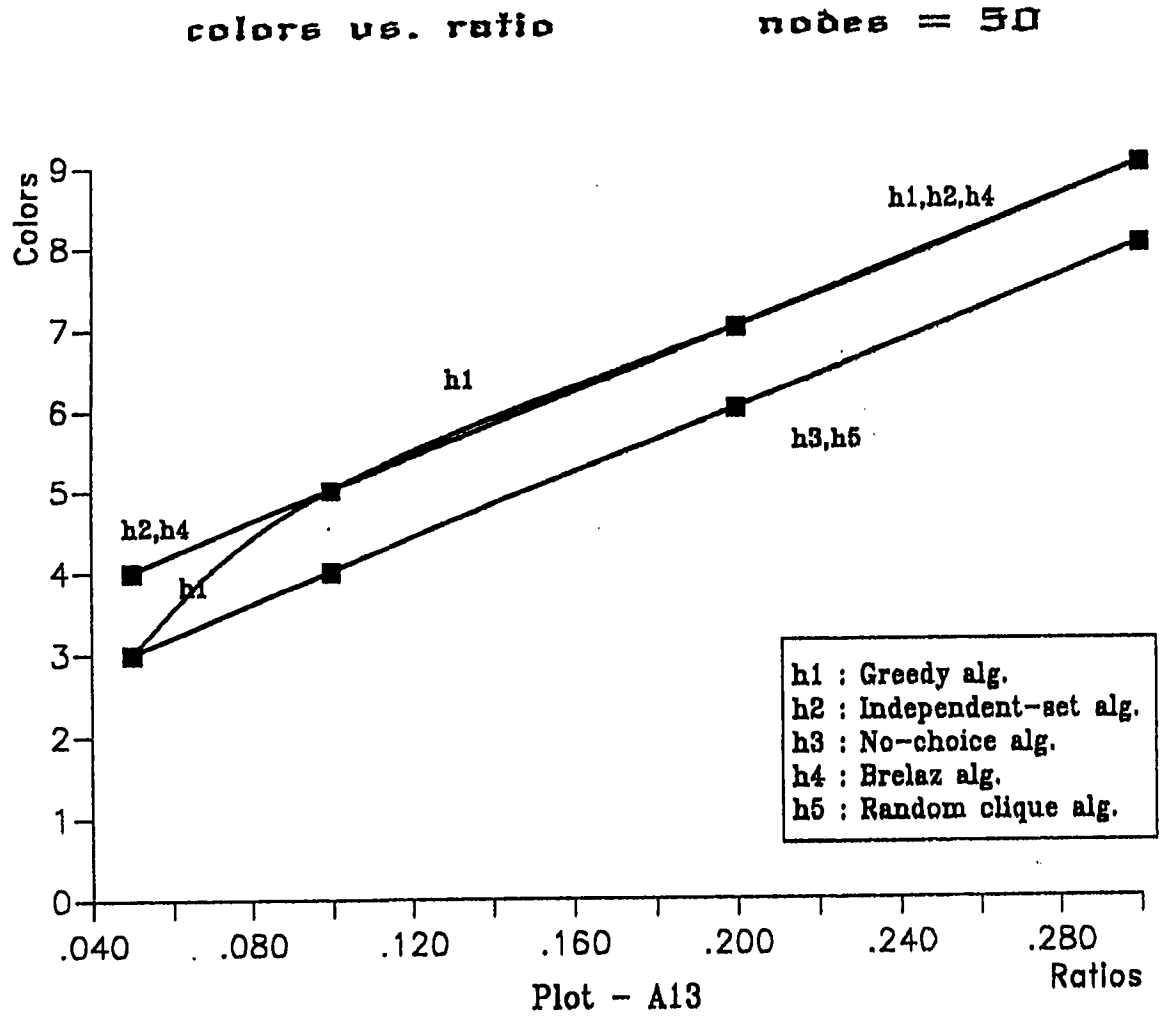

ratio vs cputime
nodes = 250
h1 : Greedy alg.
h2 : Independent-set alg.
h3 : No-choice alg.
h4 : Brelaz alg.
h5 : Random clique alg.
cputime-clock ticks
20,000
18,000
16,000
14,000
12,000
10,000
8,000
6,000
4,000
2,000
0
.040
.080
.120
.160
.200
.240
.280
Ratios
Plot - A8
h4
h2
h5
h3
h1

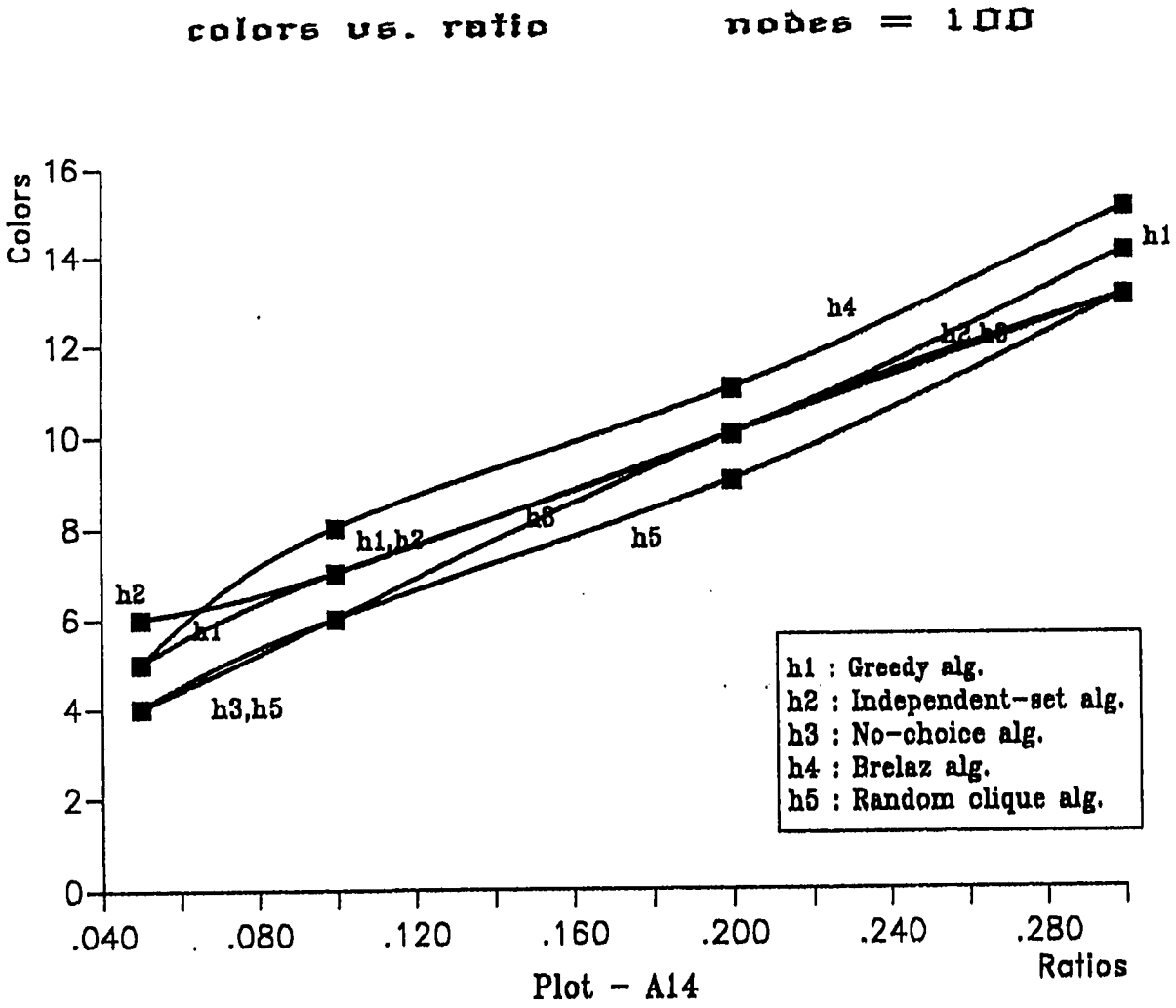

nodes vs. colors
ratio = 0.05
colors
nodes
Plot - A9
h1
h2
h3,h5
h4
h1 : Greedy alg.
h2 : Independent-set alg.
h3 : No-choice alg.
h4 : Brelaz alg.
h5 : Random clique alg.

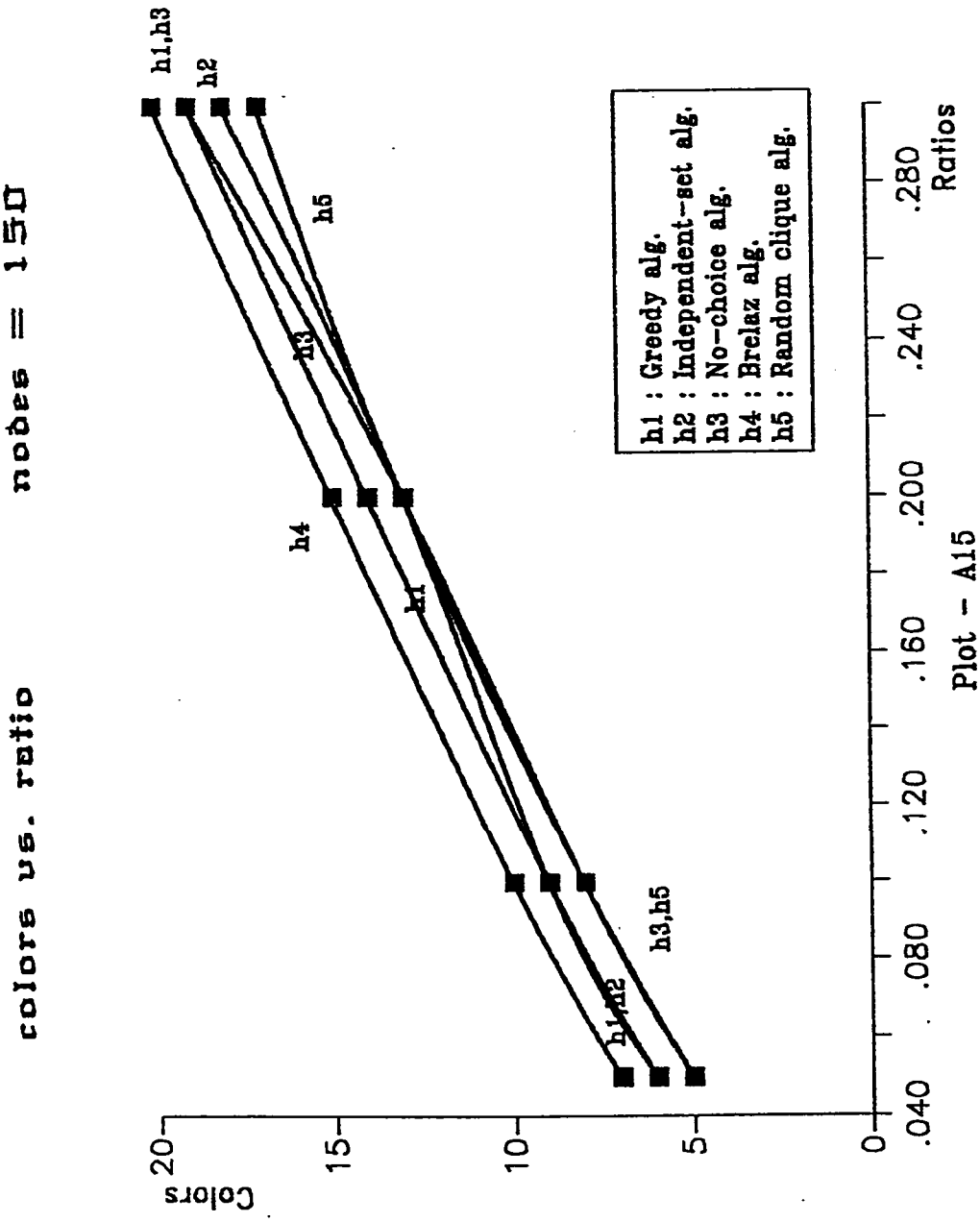

nodes vs. colors
ratio = 0.1
colors
14
12
10
8
6
4
2
0
40
60
80
100
120
140
160
180
200
220
240
260
nodes
h1,h2
h5
h3
h4
h1,h2
h3,h5
h1 : Greedy alg.
h2 : Independent-set alg.
h3 : No-choice alg.
h4 : Brelaz alg.
h5 : Random clique alg.
Plot _ A10

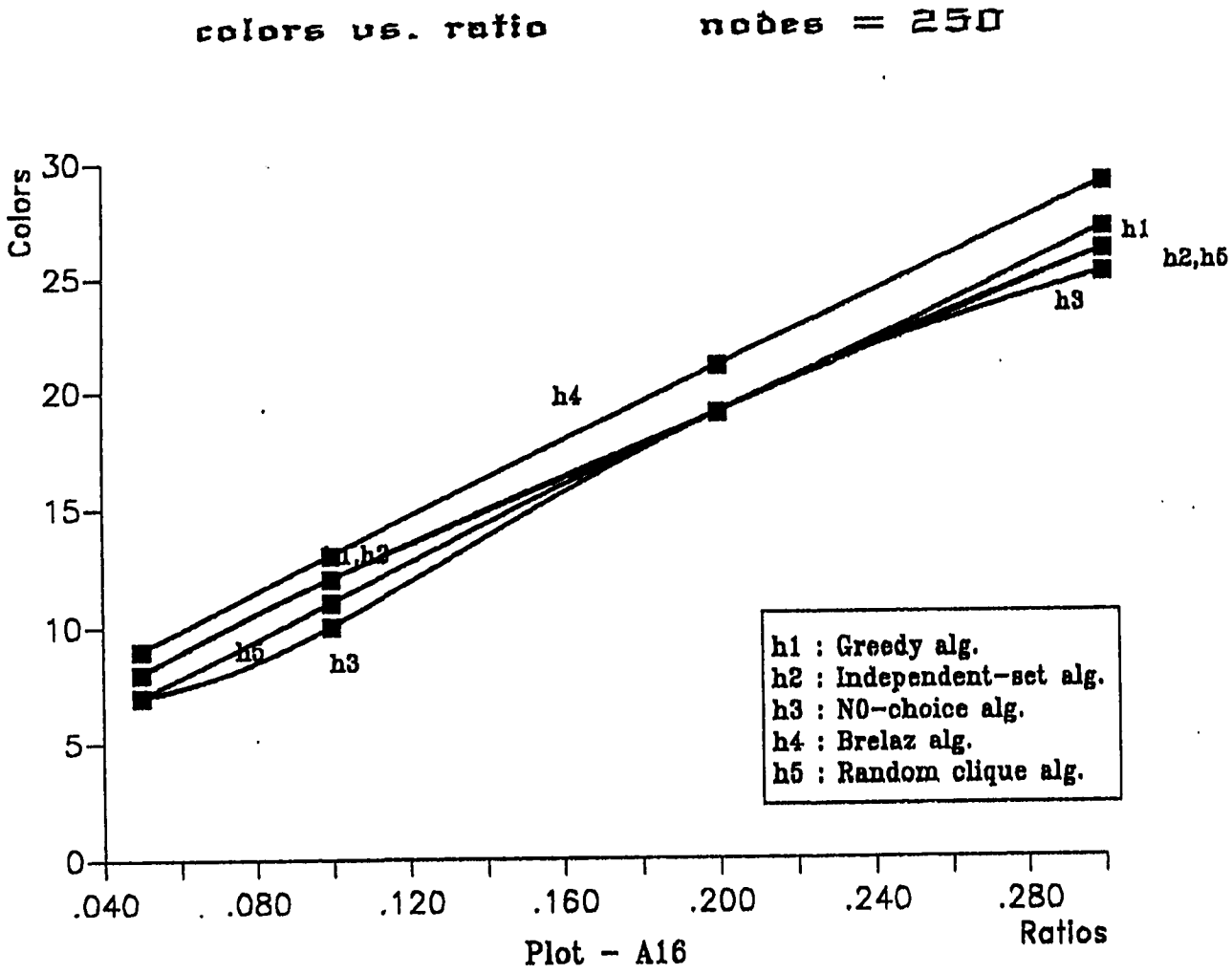

nodes vs. colors
ratio = 0.2
colors
25
20
15
10
5
0
40
60
80
100
120
140
160
180
200
220
240
260
nodes
h4
h1,h2
h3,h5
h3
h2
h1,h4
h5
h1 : Greedy alg.
h2 : Independent-set alg.
h3 : No-choice alg.
h4 : Brelaz alg.
h5 : Random clique alg.
Plot - A11


nodes vs. colors
ratio = 0.3
colors
30
25
20
15
10
5
0
40
60
80
100
120
140
160
180
200
220
240
260
nodes
h1
h2,h5
h3
h4
h2
h5
h1,h2,h4
h3,h5
h1 : Greedy alg.
h2 : Independent-set alg.
h3 : No-choice alg.
h4 : Brelaz alg.
h5 : Random clique alg.
Plot - A12


colors vs. ratio
nodes = 50
Colors
9
8
7
6
5
4
3
2
1
0
.040
.080
.120
.160
.200
.240
.280
Ratios
h1,h2,h4
h1
h3,h5
h2,h4
h1
h1 : Greedy alg.
h2 : Independent-set alg.
h3 : No-choice alg.
h4 : Brelaz alg.
h5 : Random clique alg.
Plot - A13


colors vs. ratio
nodes = 100
Colors
16
14
12
10
8
6
4
2
0
.040
.080
.120
.160
.200
.240
.280
Ratios
h1
h4
h2,h3
h1,h2
h3
h5
h2
h1
h3,h5
h1 : Greedy alg.
h2 : Independent-set alg.
h3 : No-choice alg.
h4 : Brelaz alg.
h5 : Random clique alg.
Plot - A14


colors vs. ratio
nodes = 150
Colors
20
15
10
5
0
.040
.080
.120
.160
.200
.240
.280
Ratios
Plot - A15
h1,h3
h2
h5
h3
h4
h1
h3,h5
h1,h2
h1 : Greedy alg.
h2 : Independent-set alg.
h3 : No-choice alg.
h4 : Brelaz alg.
h5 : Random clique alg.


colors vs. ratio
nodes = 250
Colors
30
25
20
15
10
5
0
.040
.080
.120
.160
.200
.240
.280
Ratios
h1
h2,h5
h3
h4
h1,h2
h5
h3
h1 : Greedy alg.
h2 : Independent-set alg.
h3 : NO-choice alg.
h4 : Brelaz alg.
h5 : Random clique alg.
Plot - A16

small graphs and ratios the Brelaz algorithm tends to give better results than the other algorithms.

From plots A9 to A16 we can see that the Random-clique algorithm gives the best coloring in most cases followed by the No-choice algorithm, followed by the Independent-set followed by the Greedy algorithm followed by the Brelaz algorithm. The No-choice algorithm tends to give better results as the size of graph and ratio decreases, as in the case of (100 nodes or less and 0.1 ratio).

Two tables that rank the algorithms are shown below. One shows the ranking of the algorithms in an increasing order of number of colors required. The second ranks the algorithms in an increasing order of CPU time consumed to color the graph. Different ranks are for different intervals of graph size and ratio.

In the case of low ratio and size the Independent-set algorithm gets better results. As the ratio increases all the algorithms increase their coloring drastically.

6.3.2 Effect of Node Splitting

In practice, we have limited number of colors to color the graph. Algorithms that require more colors need to split some of their nodes so that it can provide a proper graph coloring. Plots A17 - A32 show the effect of node splitting on some of the algorithms, ones that require large number of colors.

Each plot, histogram, reflects the amount of CPU time required by each algorithm for a specific graph. The number of colors to color the graph with is preset. It is less than the number required by some of the algorithms. It can be seen that the Brelaz algorithm requires large amount of time to keep its coloring as low as the others. The Independent-set also requires high amount of time but not as much as the Brelaz algorithm. As the maximum number of colors required by the user is farther than the number required by the algorithm, the amount of time increases drastically.

6.3.3 Effect of Changing Environment

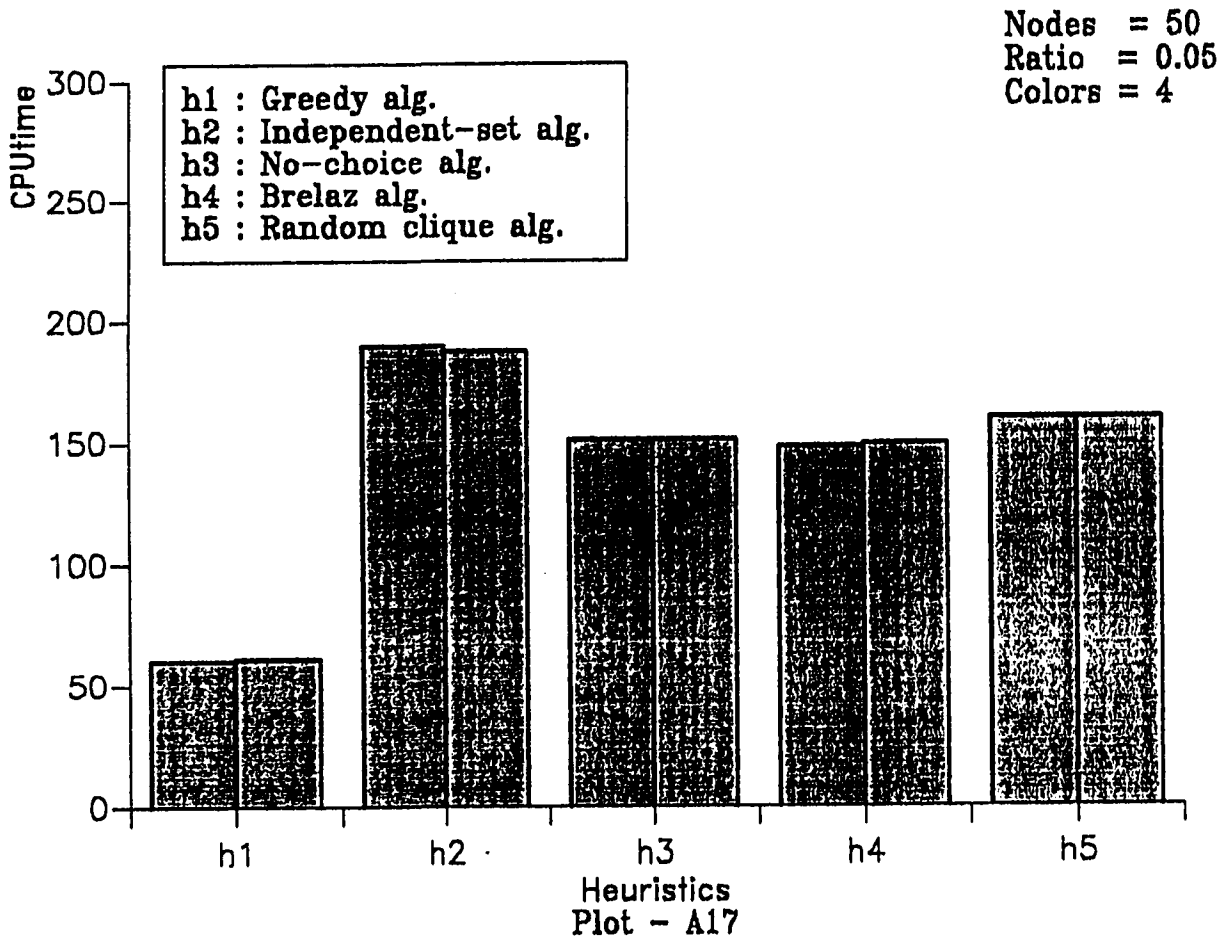
Plots A33-A42 show the behavior of algorithms towards the changing environment. The behavior of each algorithm is presented in two plots. One shows the effect of ratio on the coloring of the graph and the other shows the effect of graph size on the coloring. In the first plot four curves are plotted one for each graph size and in the second four curves are plotted each represent different ratios. We can see from the plots A33-A42 that the curves are smoothly plotted to illustrate that the algorithms are linearly affected by the environment.

6.4 Real Life Experiment

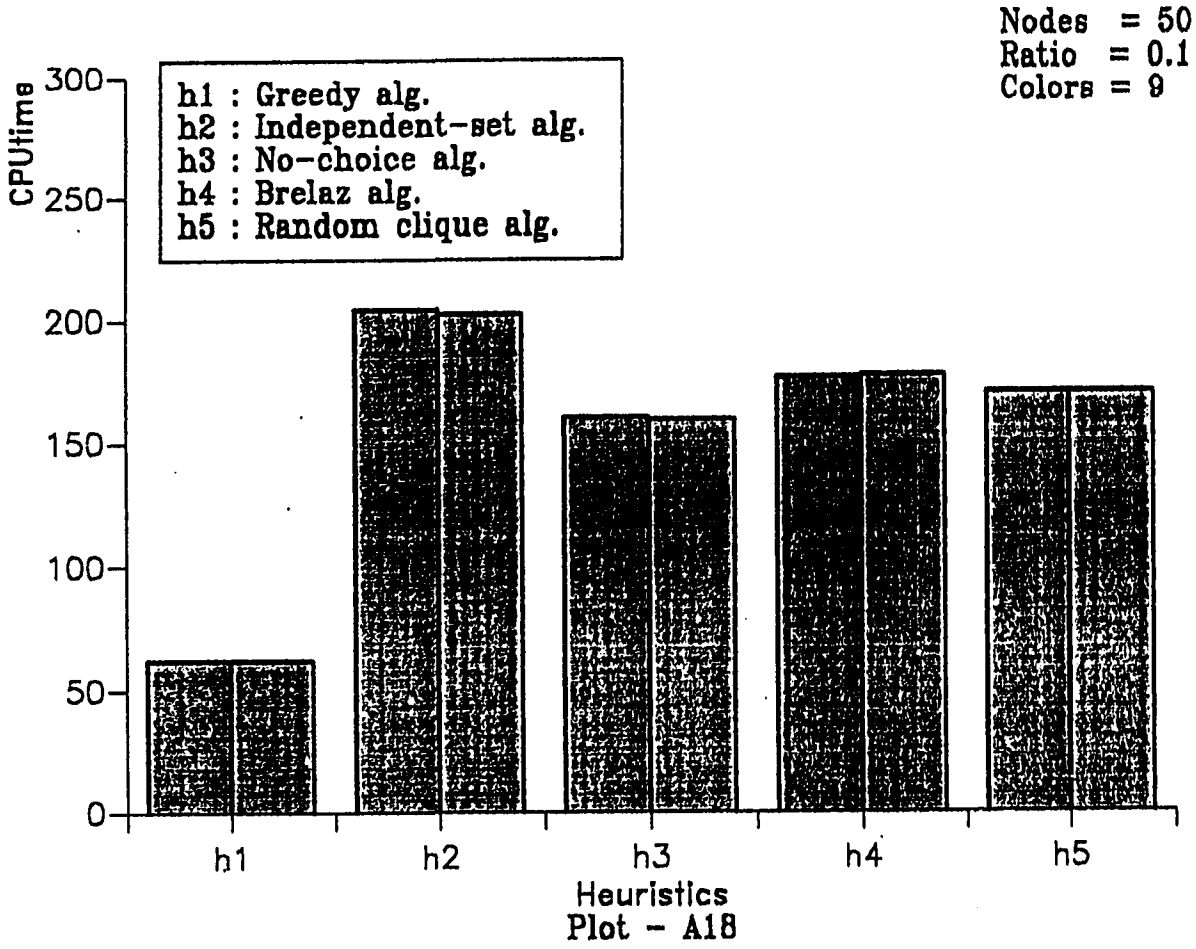
The above algorithms have been tested on the Dhahran high school data.

This school has about 500 students, 32 instructors, and 120 offered sections out of 85 courses. In practice far too many course conflicts occur. This problem forces the school to do many fixations some of which even violate policies. One of these decisions is to have a student take three periods in a day. Another decision is to have a student taking the examination in different time than his classmates. The above algorithms in their first run (with no node-splitting) on the high school data resulted in 22 time slots.

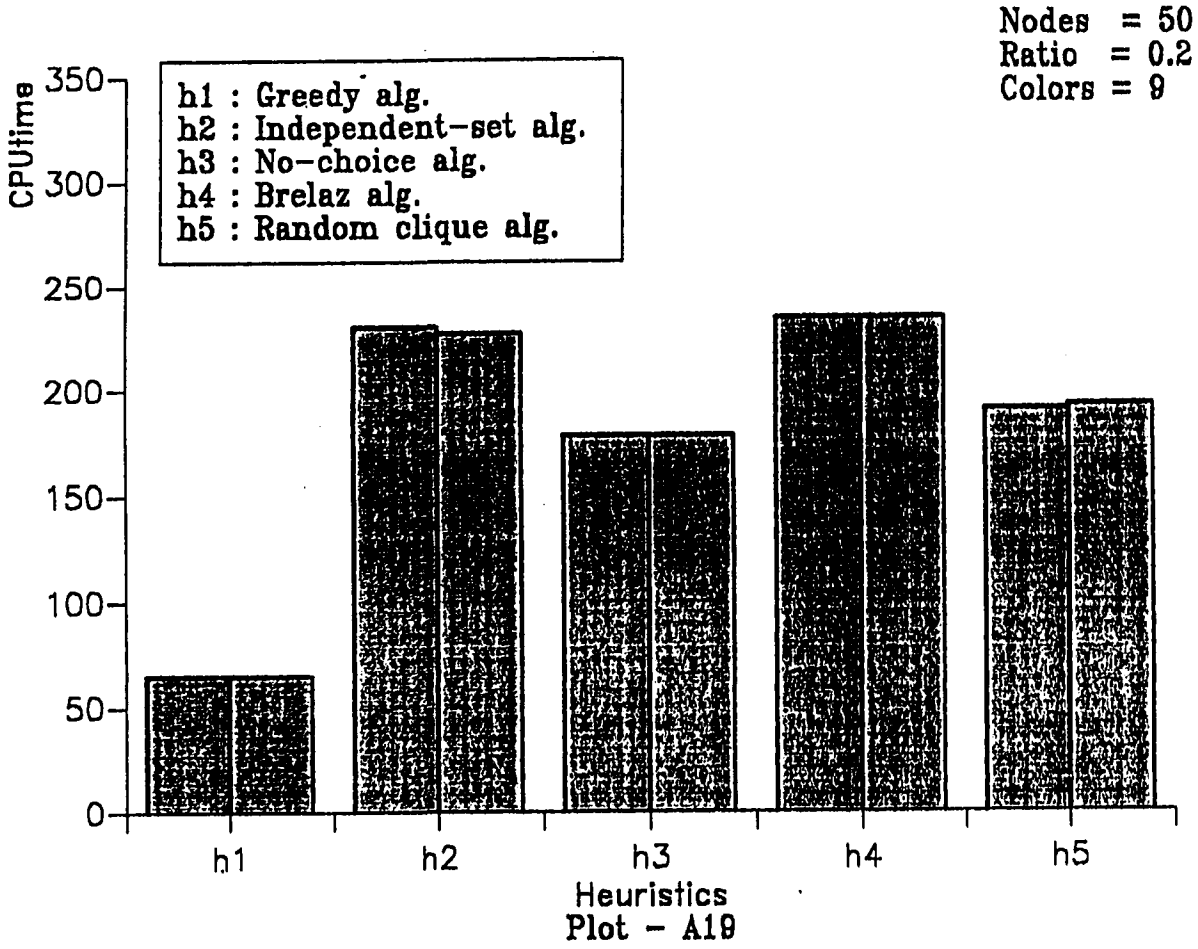
EFFECT OF NODE SPLITTING ON CPU-TIME



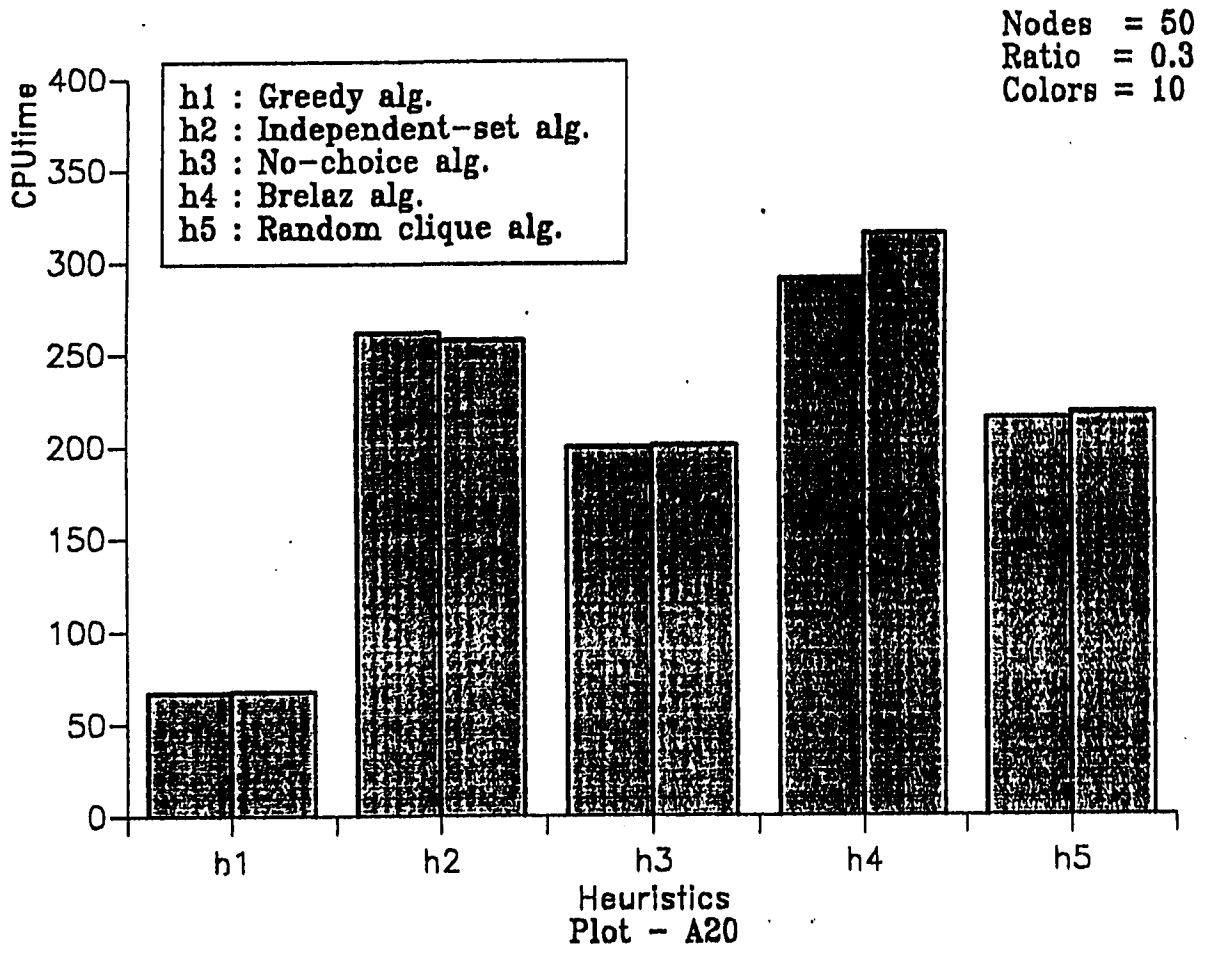
EFFECT OF NODE SPLITTING ON CPU-TIME



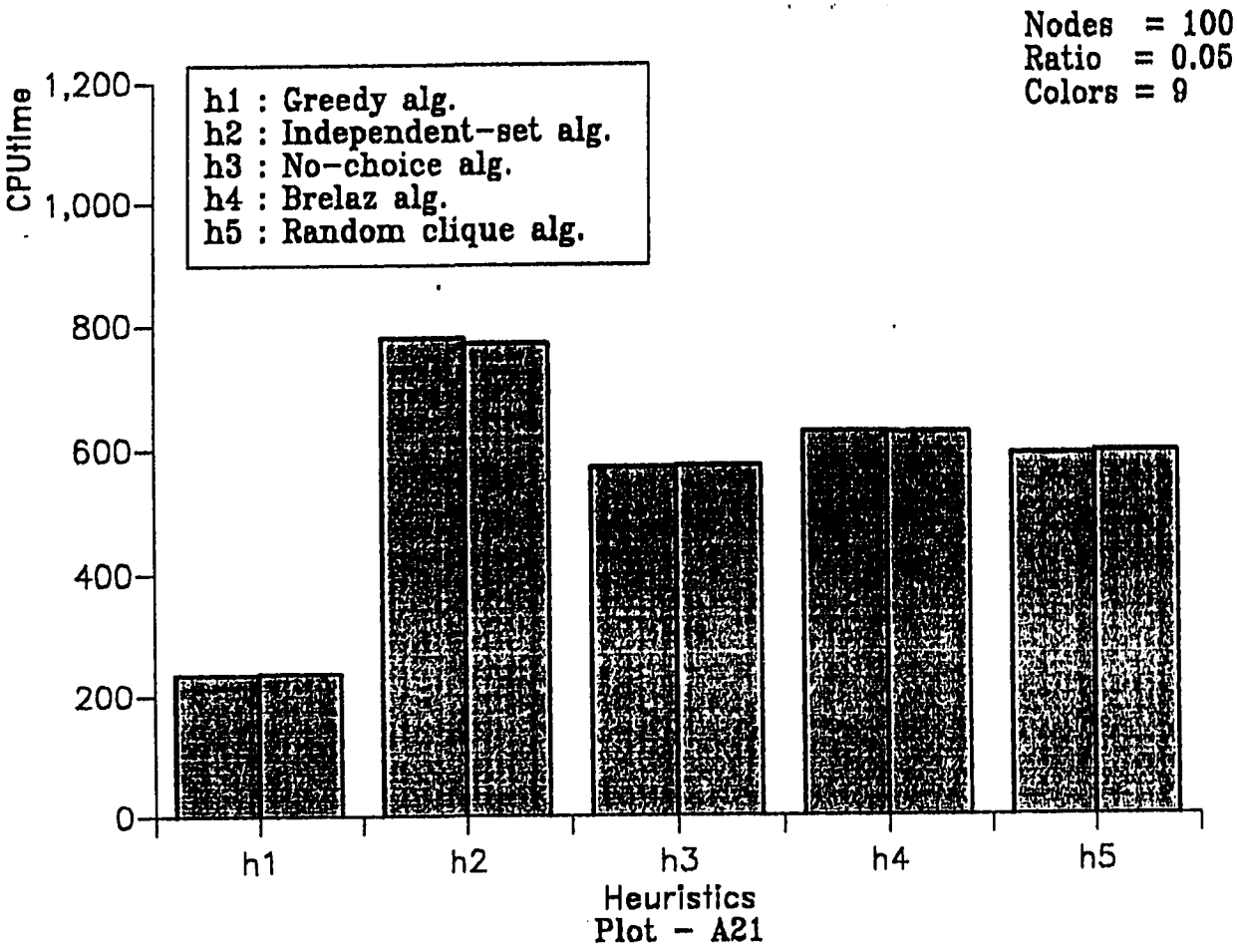
EFFECT OF NODE SPLITTING ON CPU-TIME



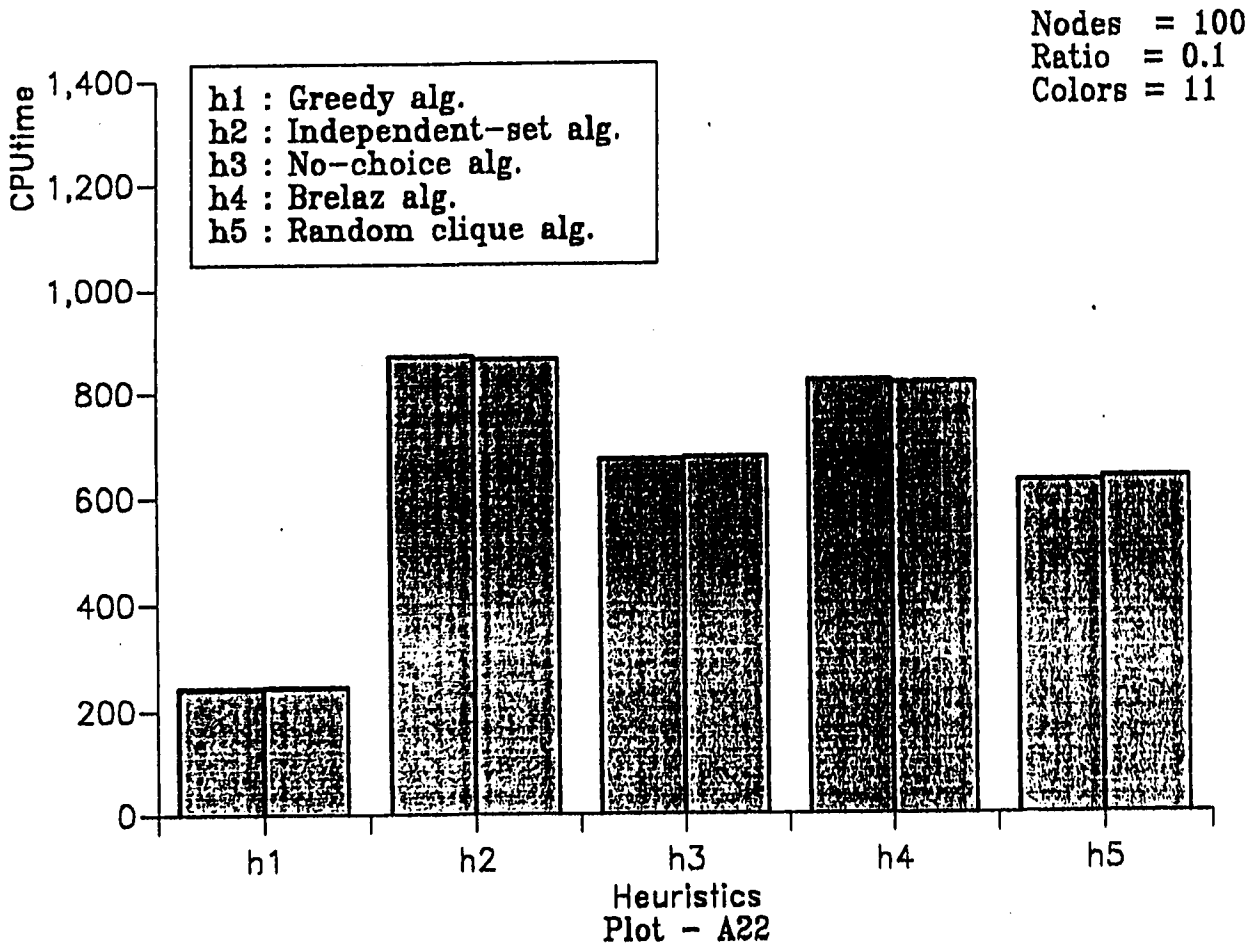
EFFECT OF NODE SPLITTING ON CPU-TIME



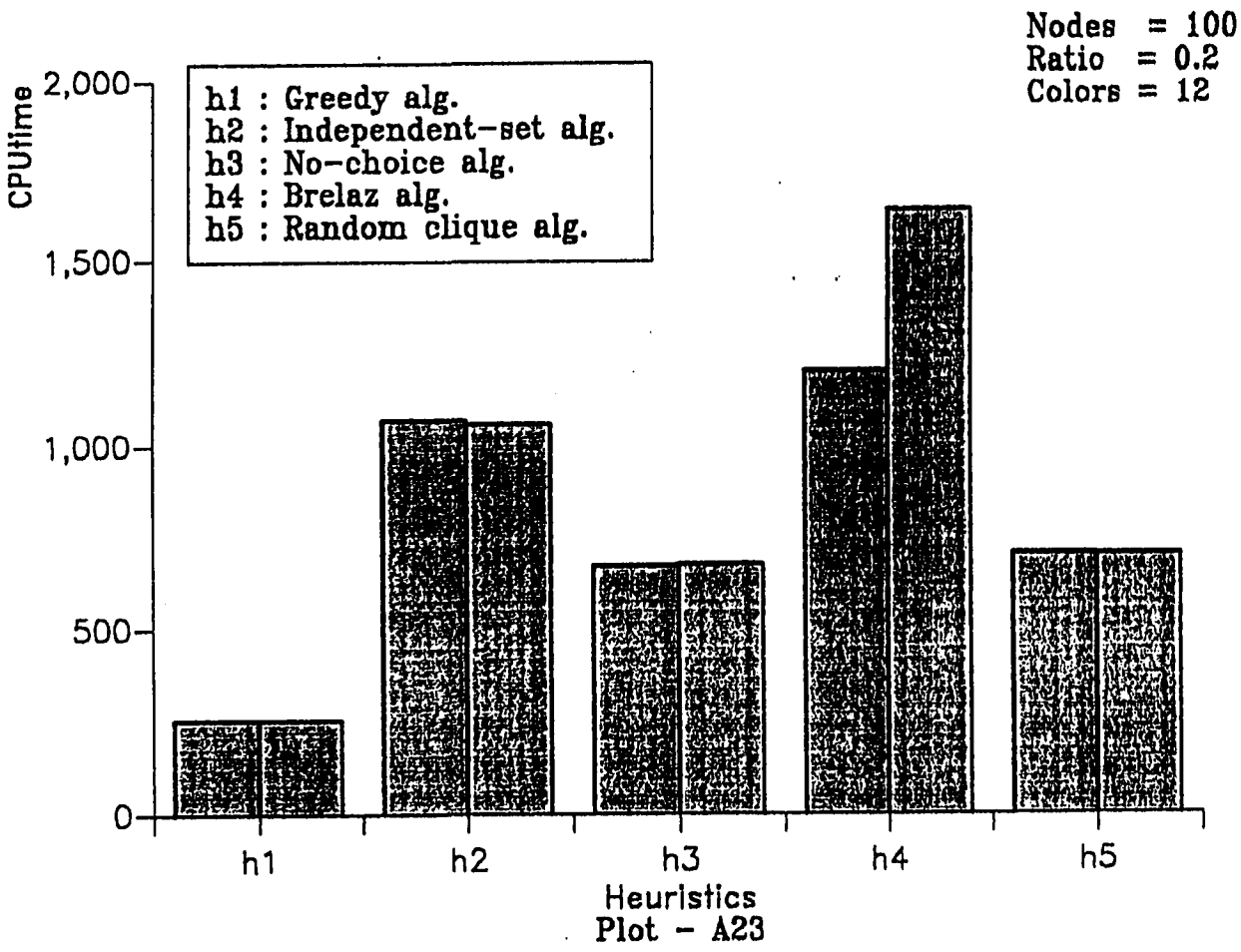
EFFECT OF NODE SPLITTING ON CPU-TIME



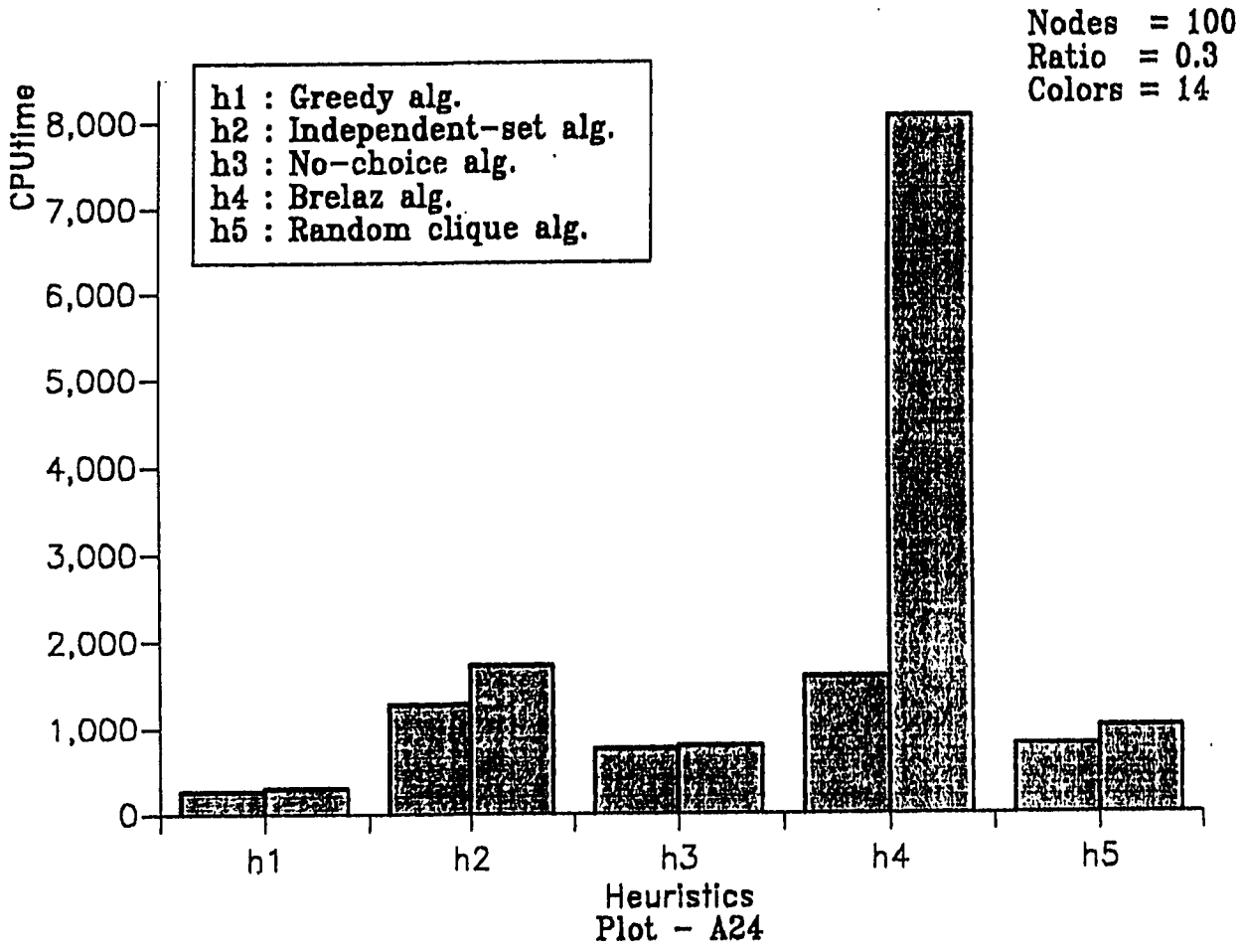
EFFECT OF NODE SPLITTING ON CPU-TIME



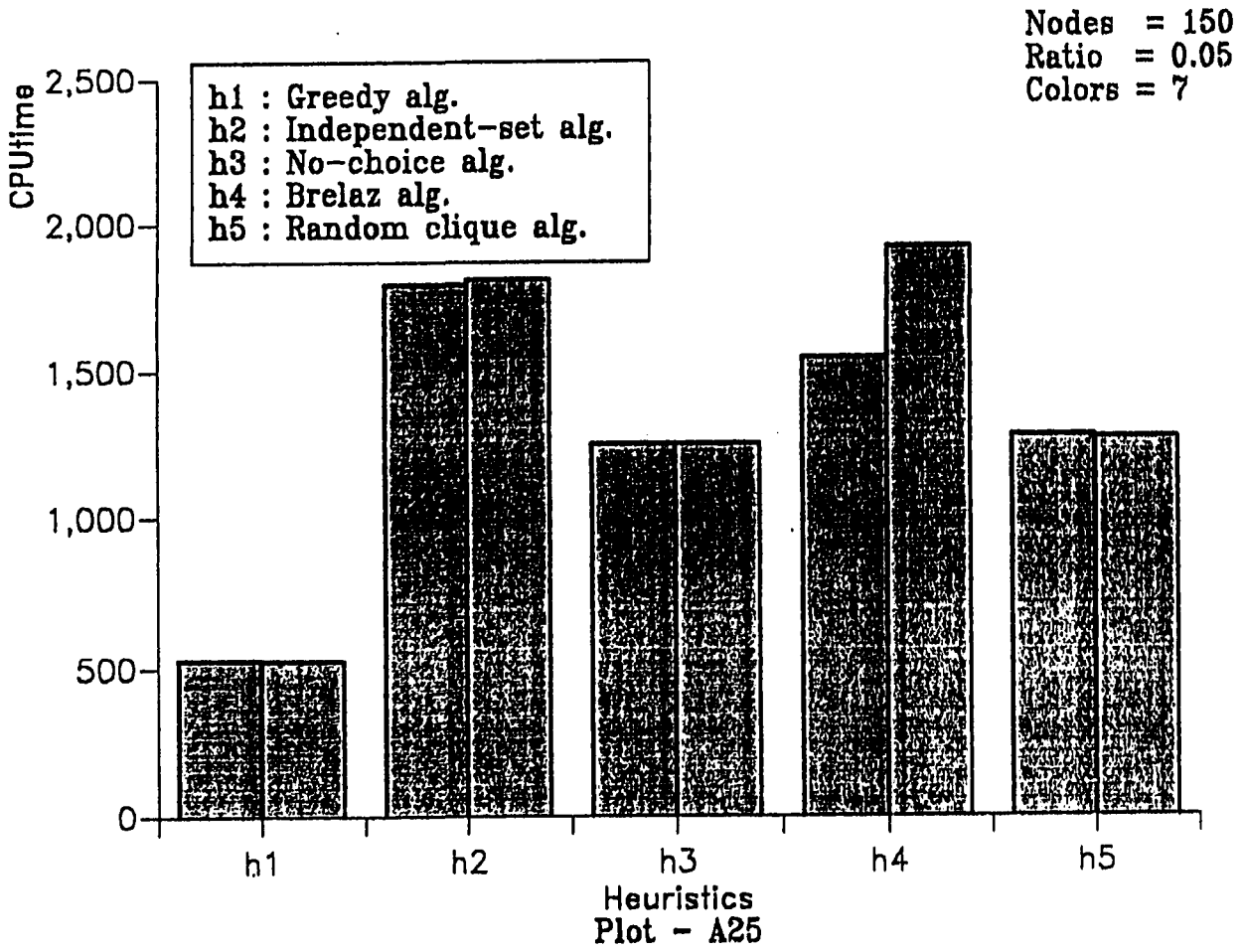
EFFECT OF NODE SPLITTING ON CPU-TIME



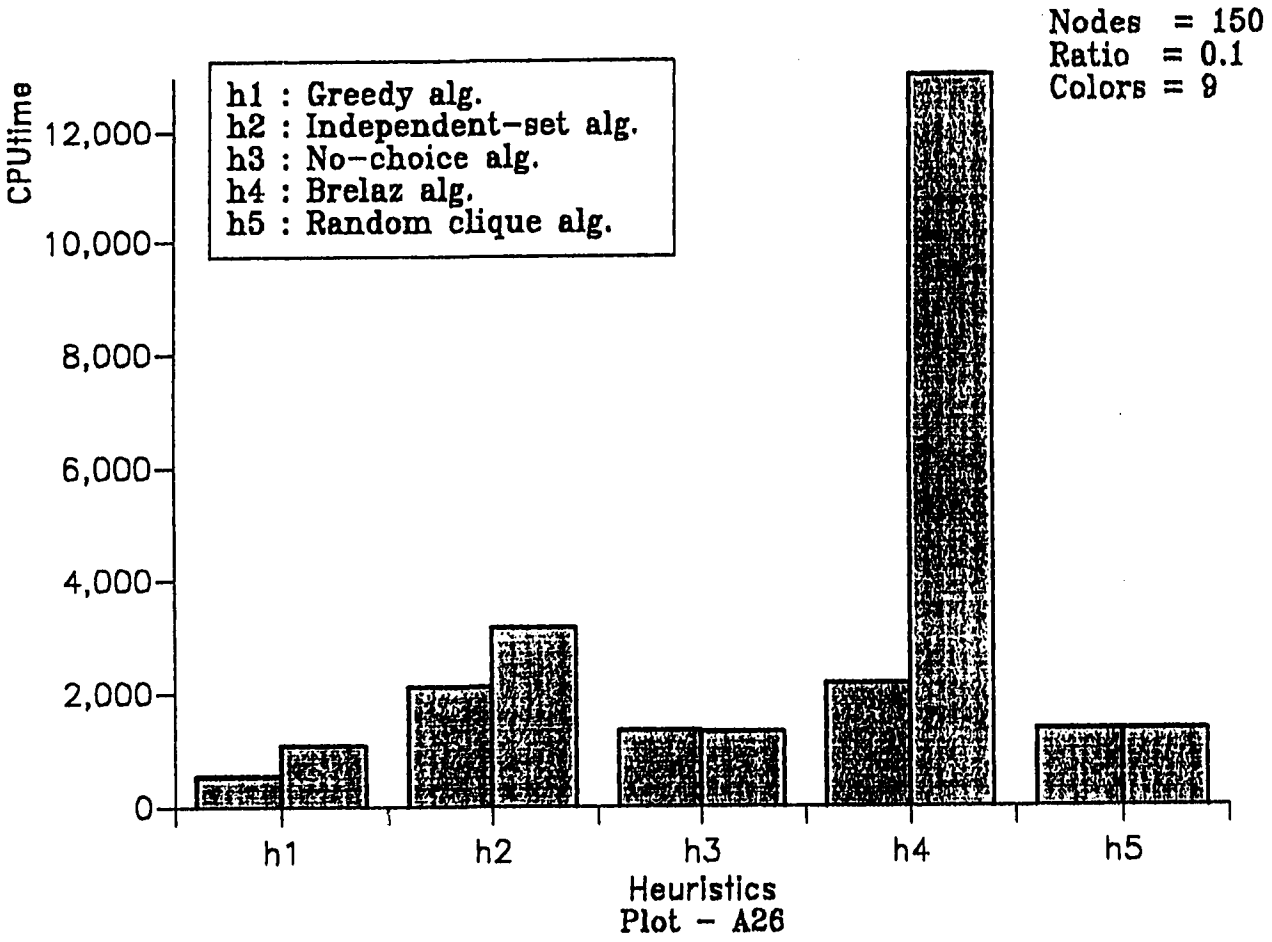
EFFECT OF NODE SPLITTING ON CPU-TIME



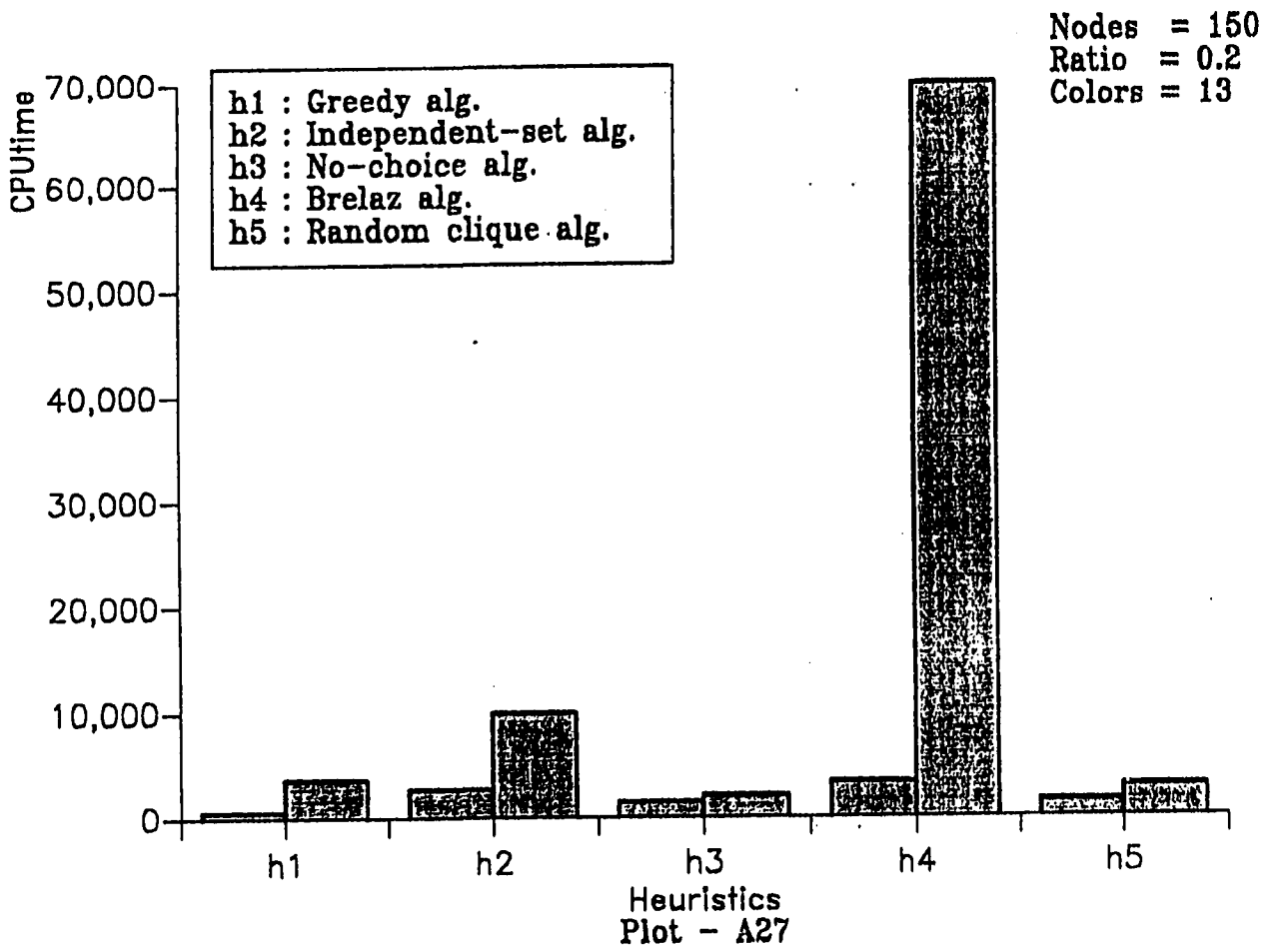
EFFECT OF NODE SPLITTING ON CPU-TIME



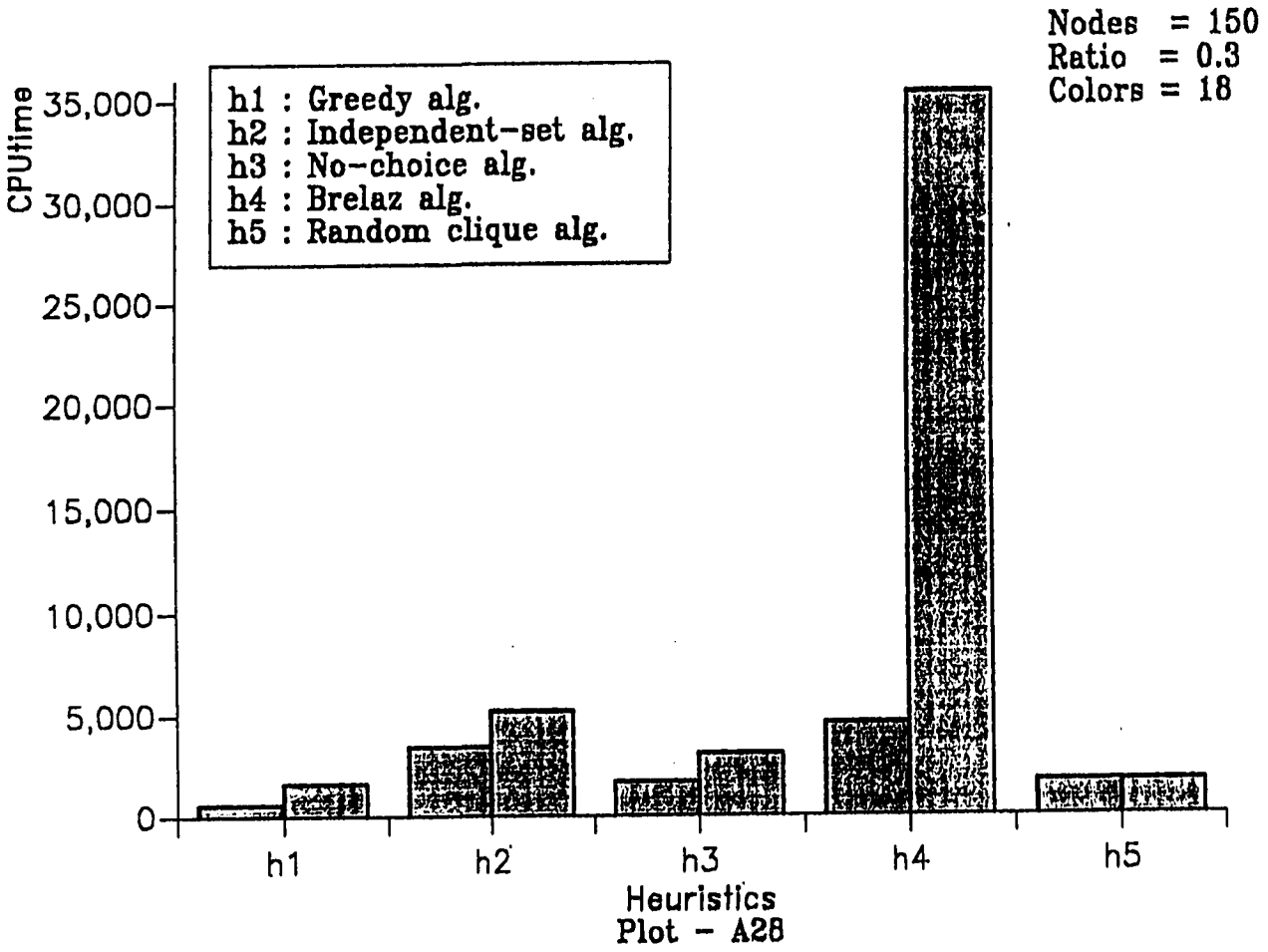
EFFECT OF NODE SPLITTING ON CPU-TIME



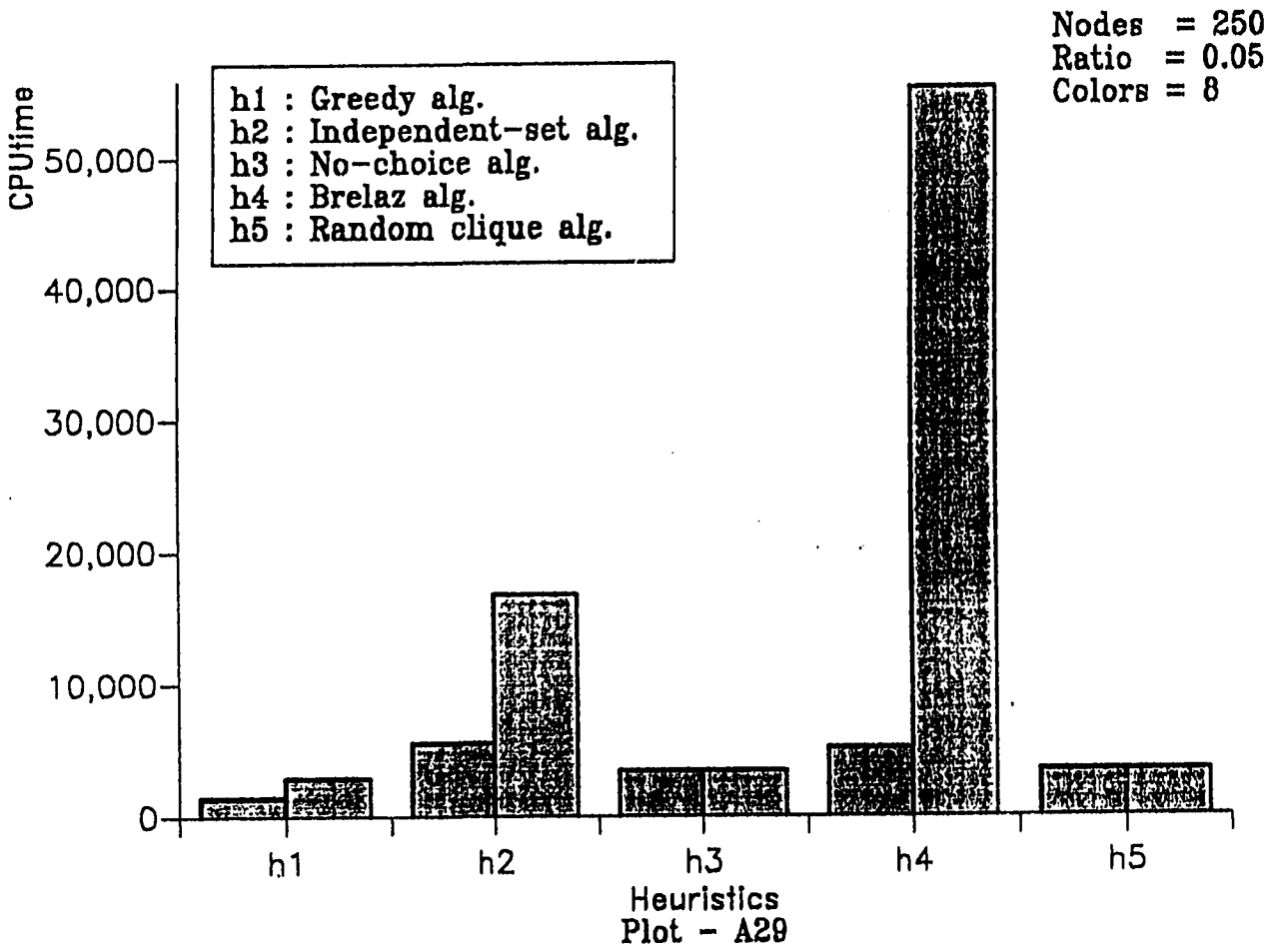
EFFECT OF NODE SPLITTING ON CPU-TIME



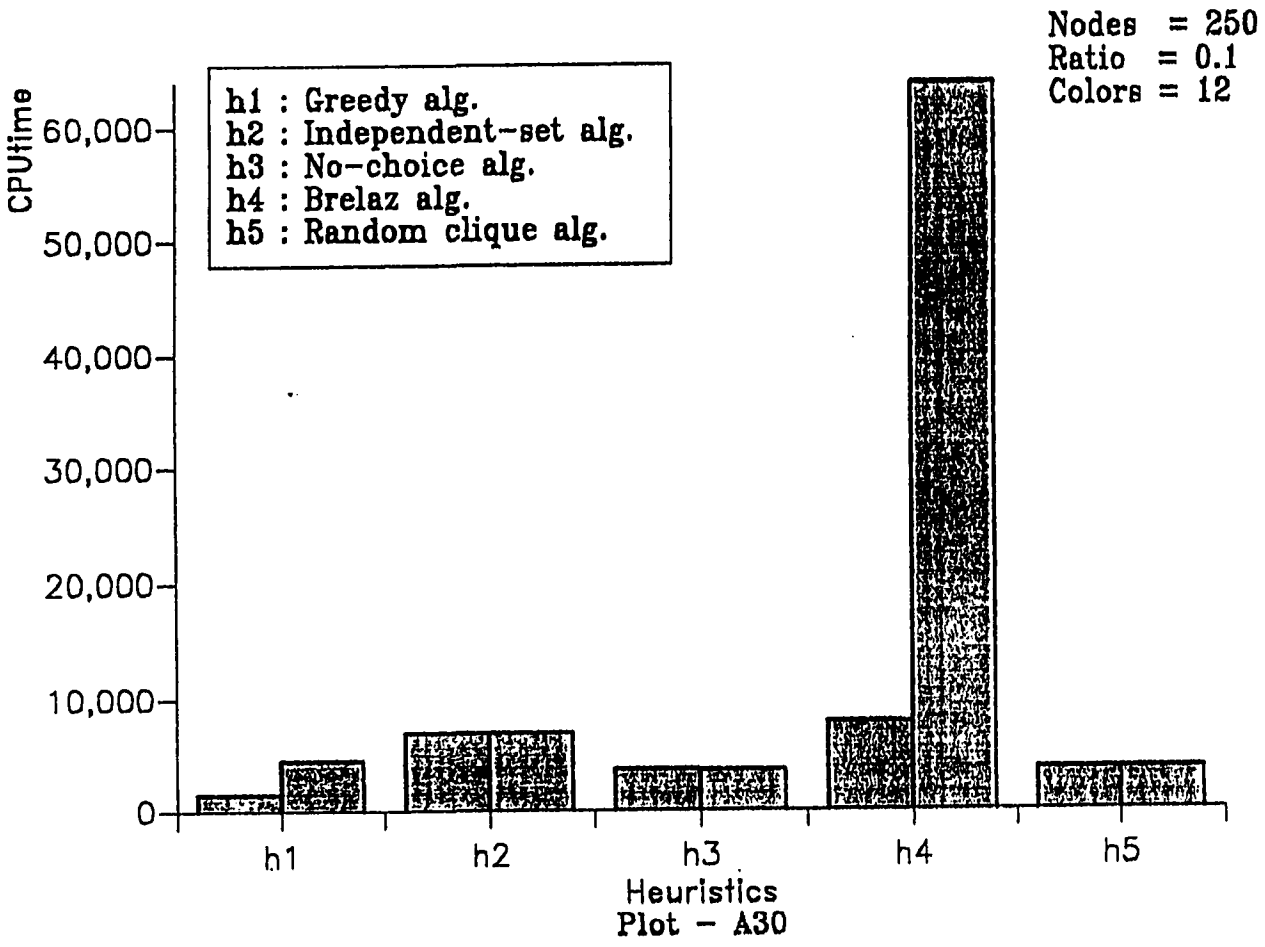
EFFECT OF NODE SPLITTING ON CPU-TIME



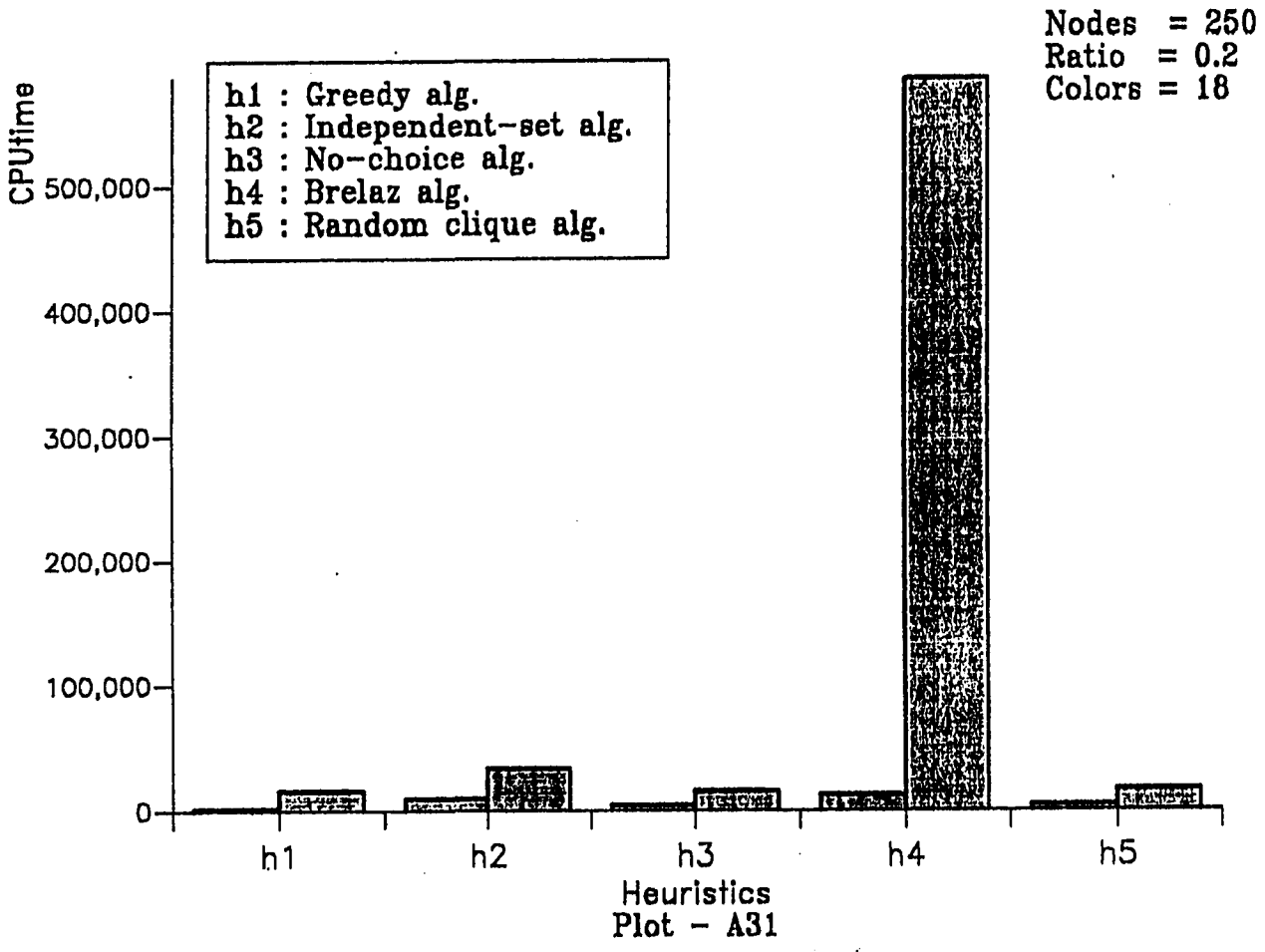
EFFECT OF NODE SPLITTING ON CPU-TIME



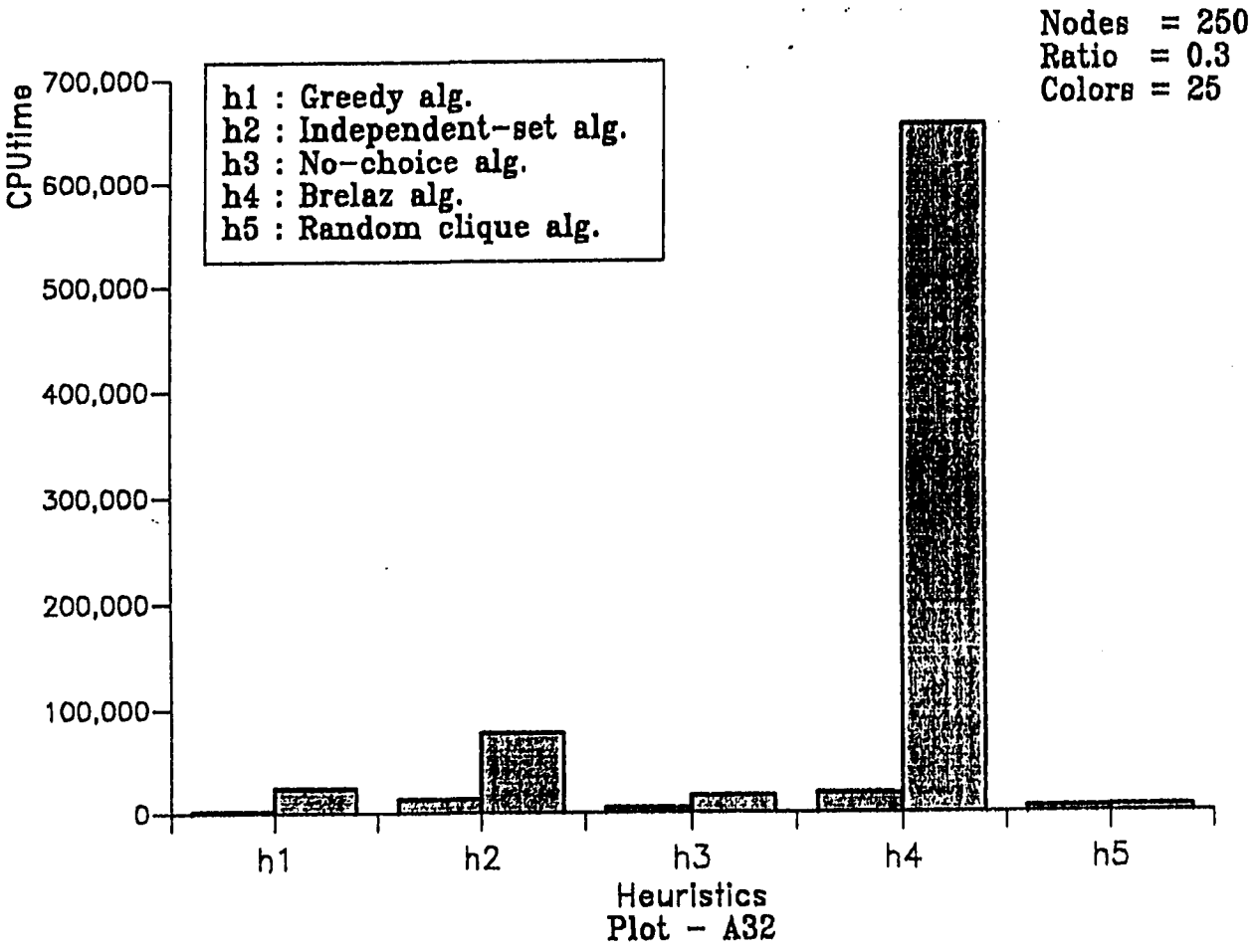
EFFECT OF NODE SPLITTING ON CPU-TIME



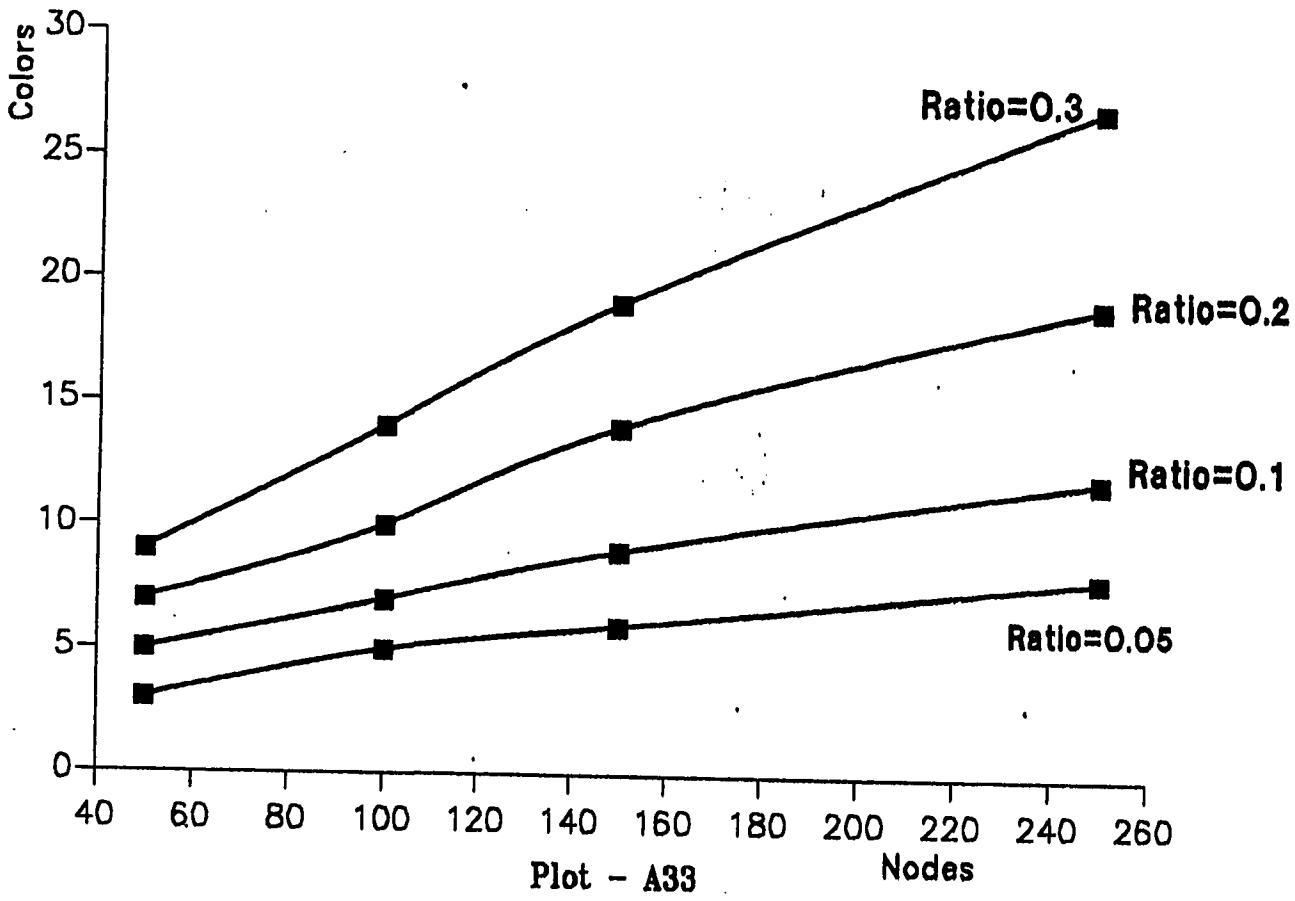
EFFECT OF NODE SPLITTING ON CPU-TIME

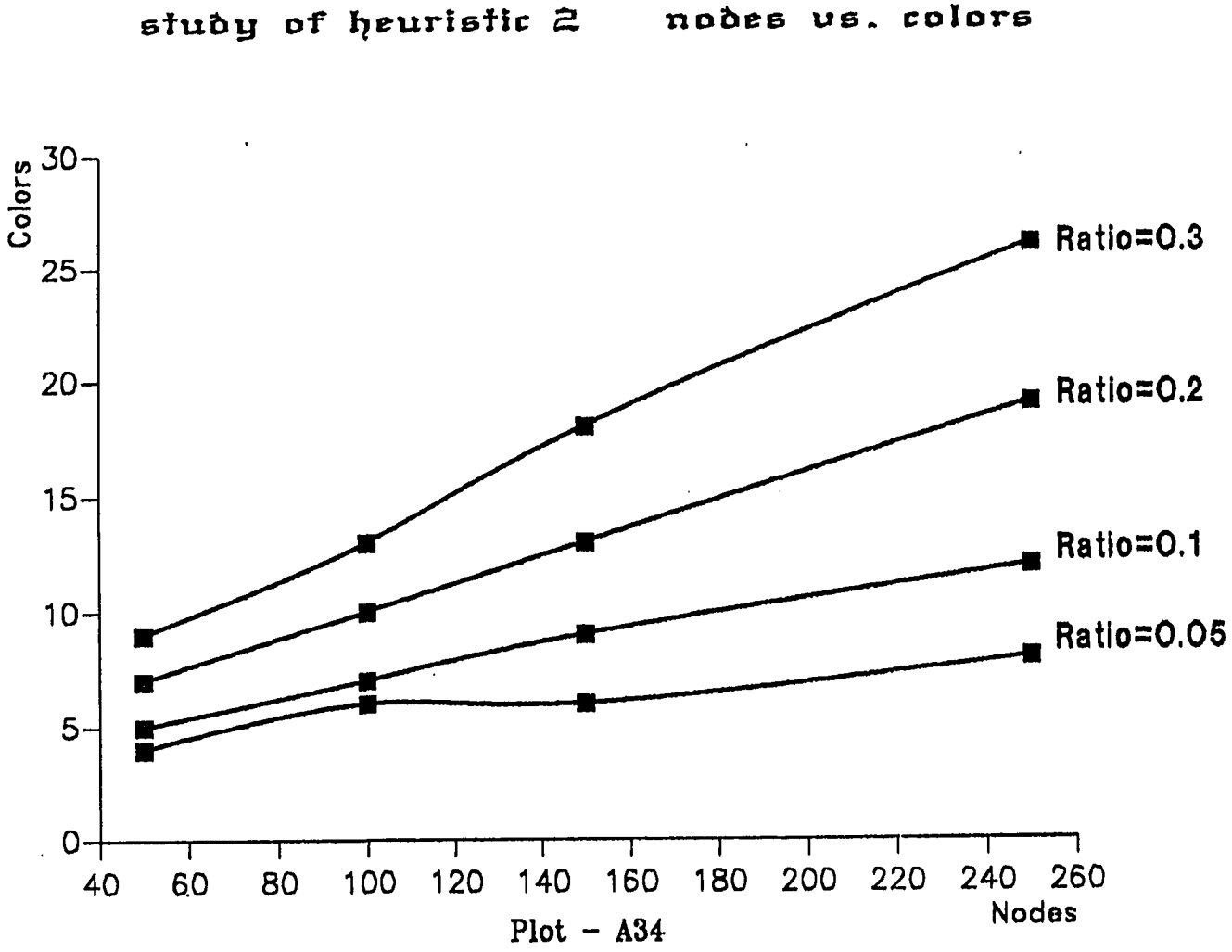


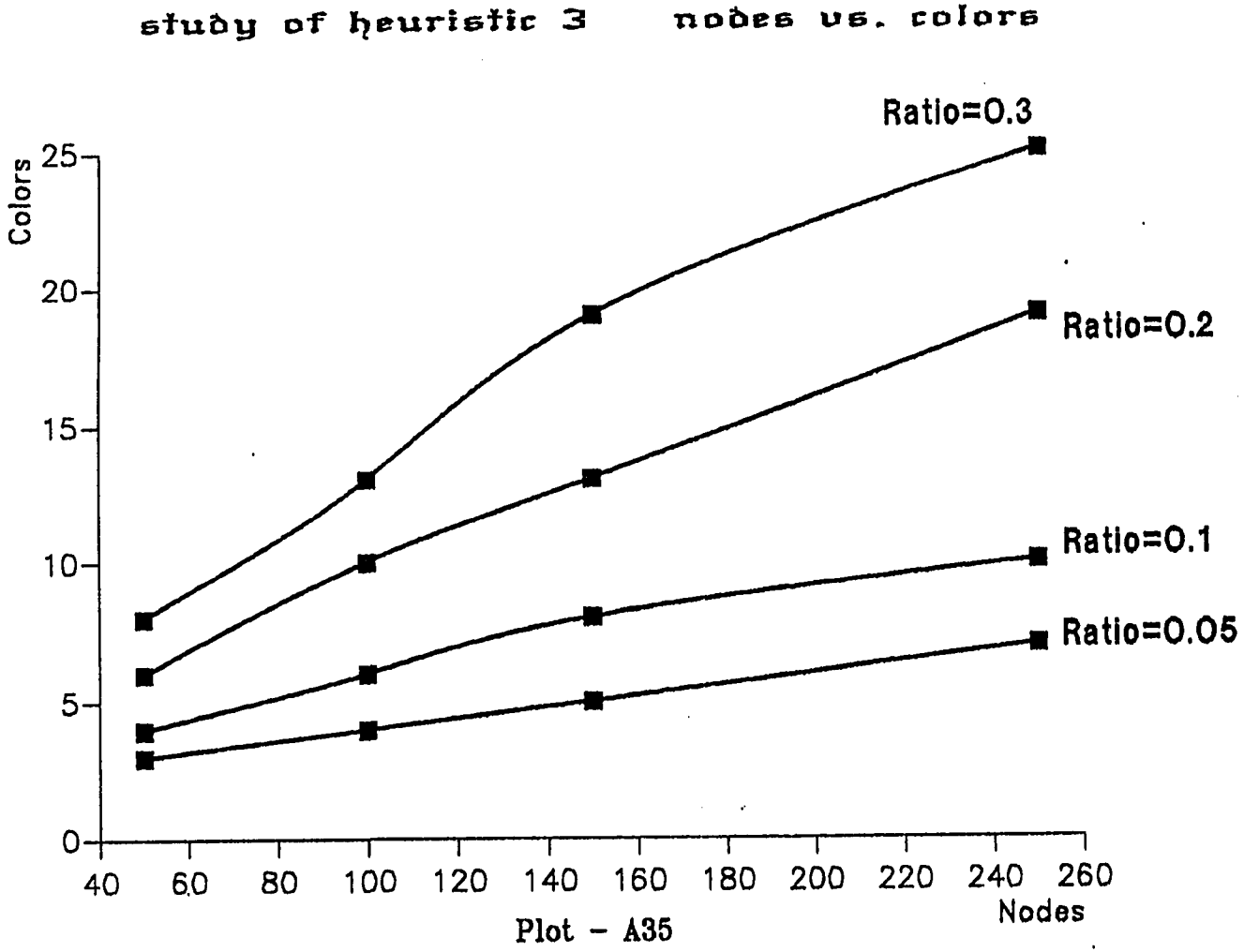
EFFECT OF NODE SPLITTING ON CPU-TIME

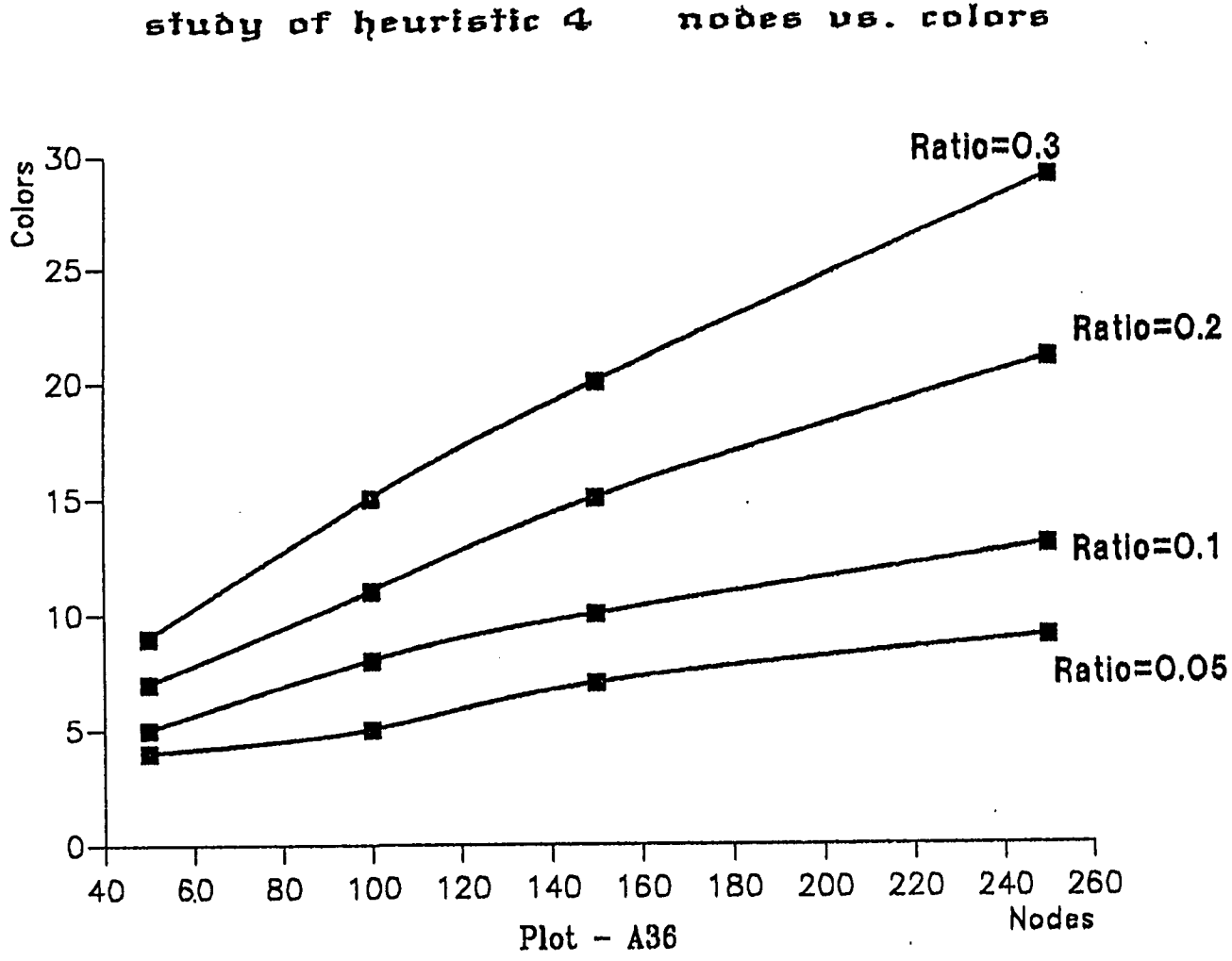


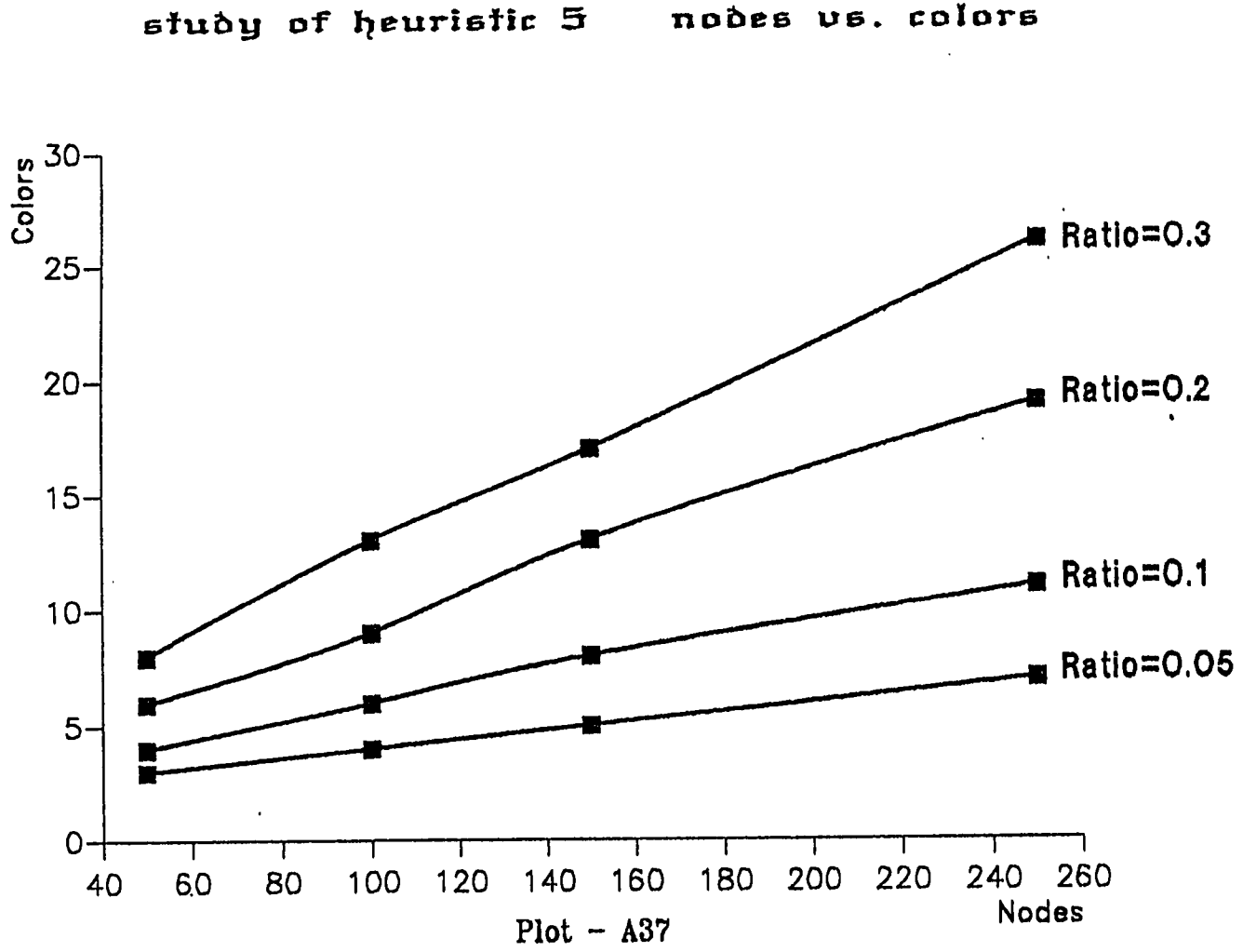
study of heuristic 1 nodes vs. colors

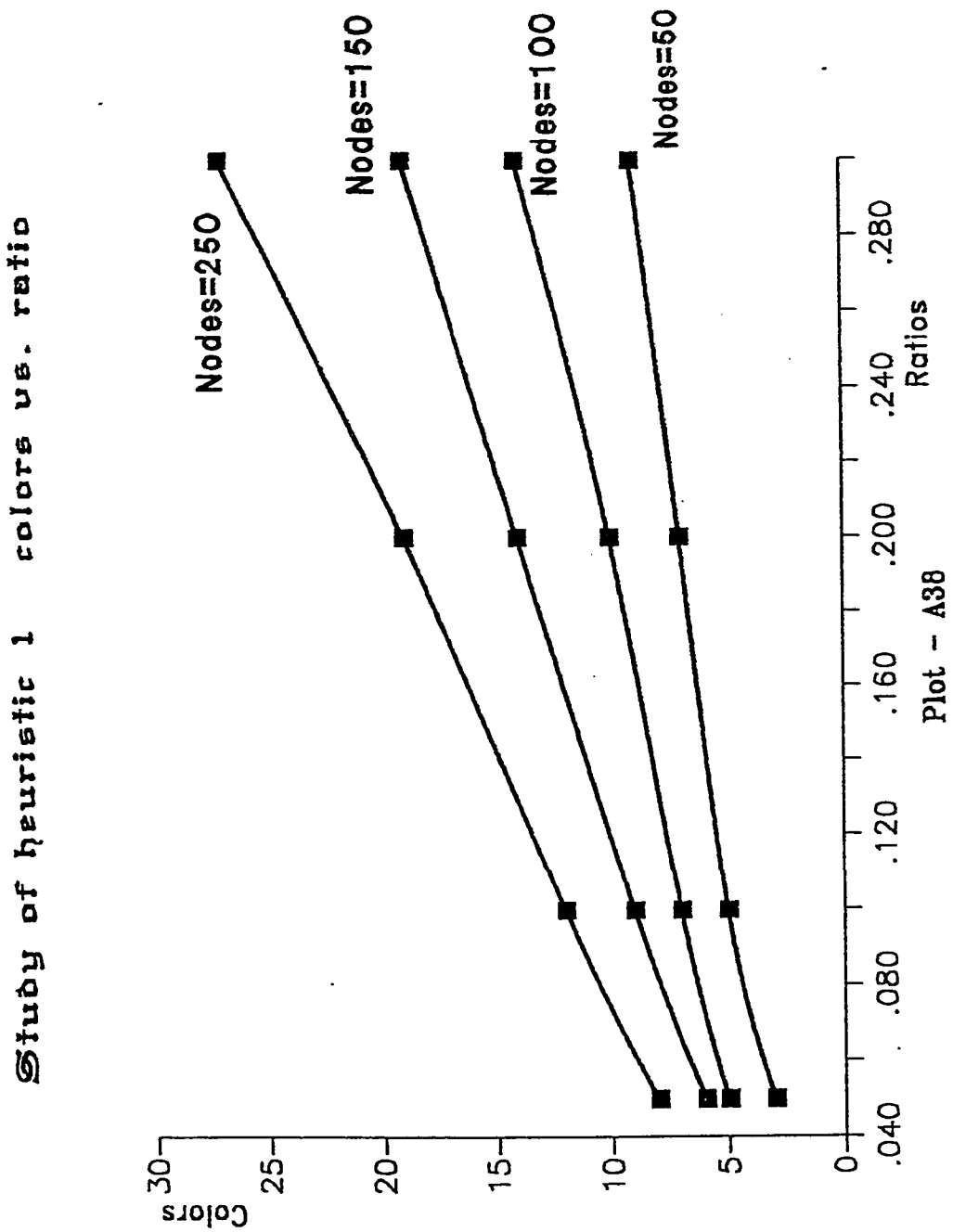


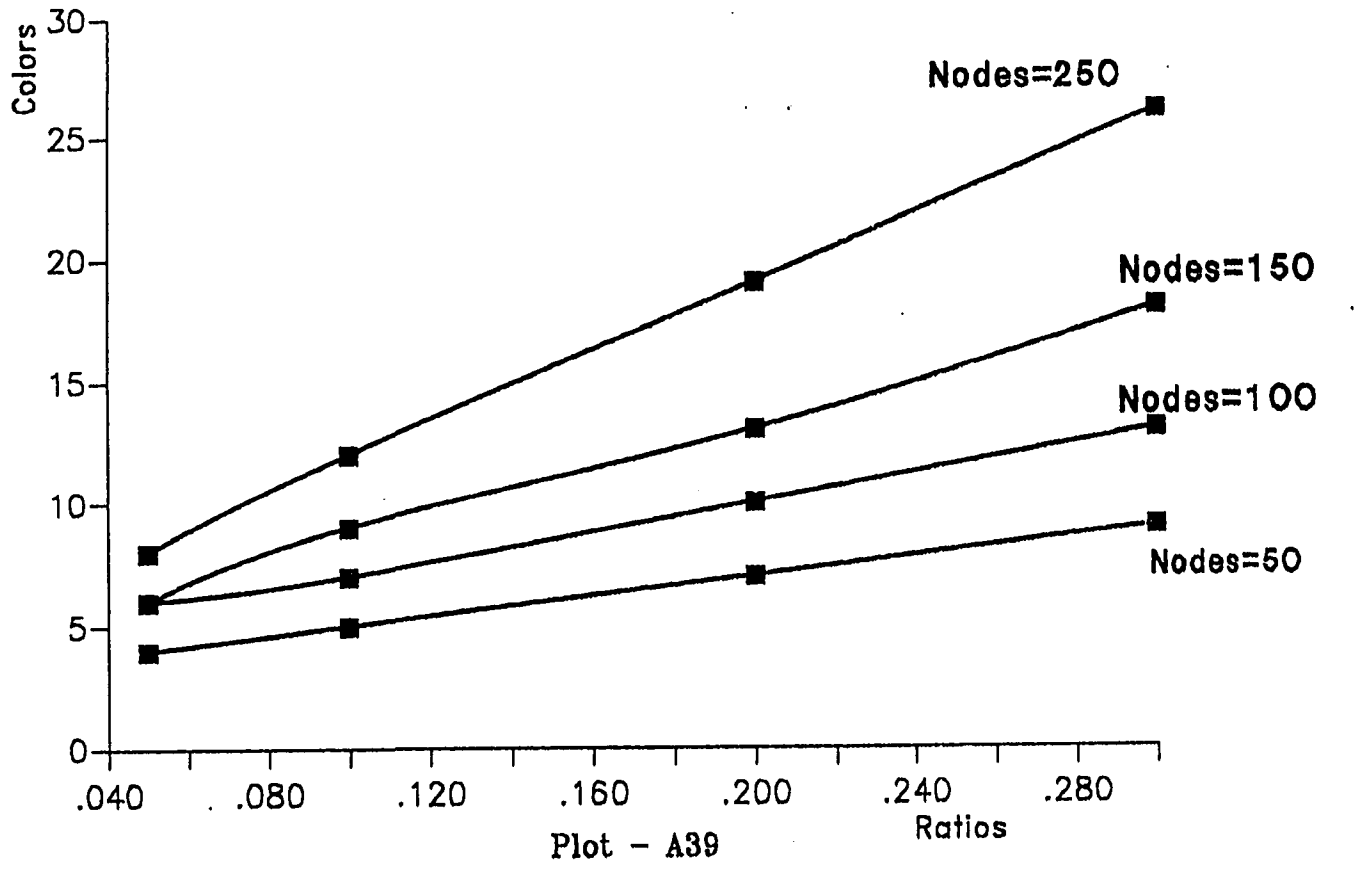

study of heuristic 2 nodes vs. colors
Colors
30
25
20
15
10
5
0
40
60
80
100
120
140
160
180
200
220
240
260
Nodes
Ratio=0.3
Ratio=0.2
Ratio=0.1
Ratio=0.05
Plot - A34

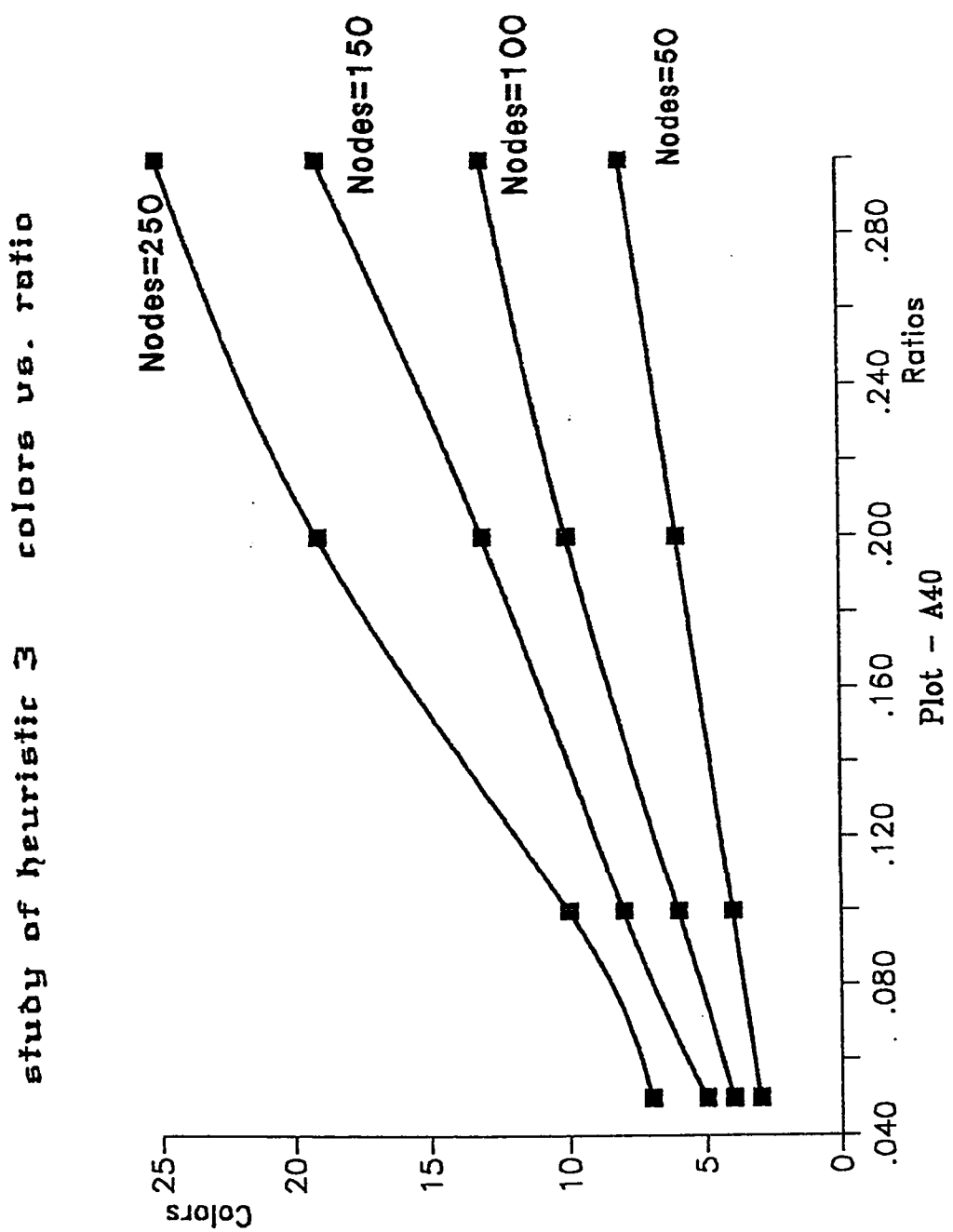

study of heuristic 3 nodes vs. colors
Ratio=0.3
Ratio=0.2
Ratio=0.1
Ratio=0.05
Colors
25
20
15
10
5
0
40
60
80
100
120
140
160
180
200
220
240
260
Nodes
Plot - A35

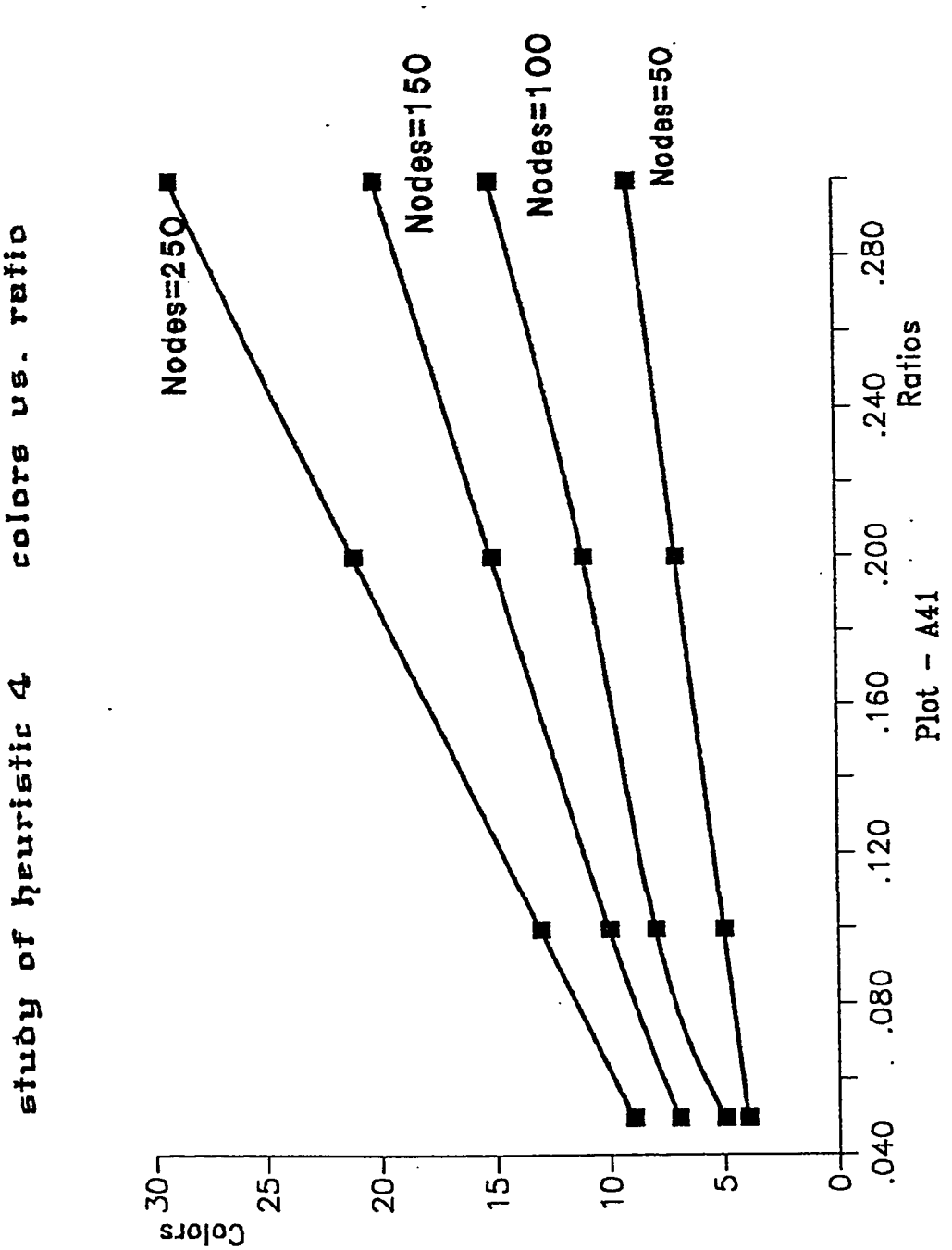

study of heuristic 4 nodes vs. colors
Ratio=0.3
Ratio=0.2
Ratio=0.1
Ratio=0.05
Colors
30
25
20
15
10
5
0
40
60
80
100
120
140
160
180
200
220
240
260
Nodes
Plot - A36

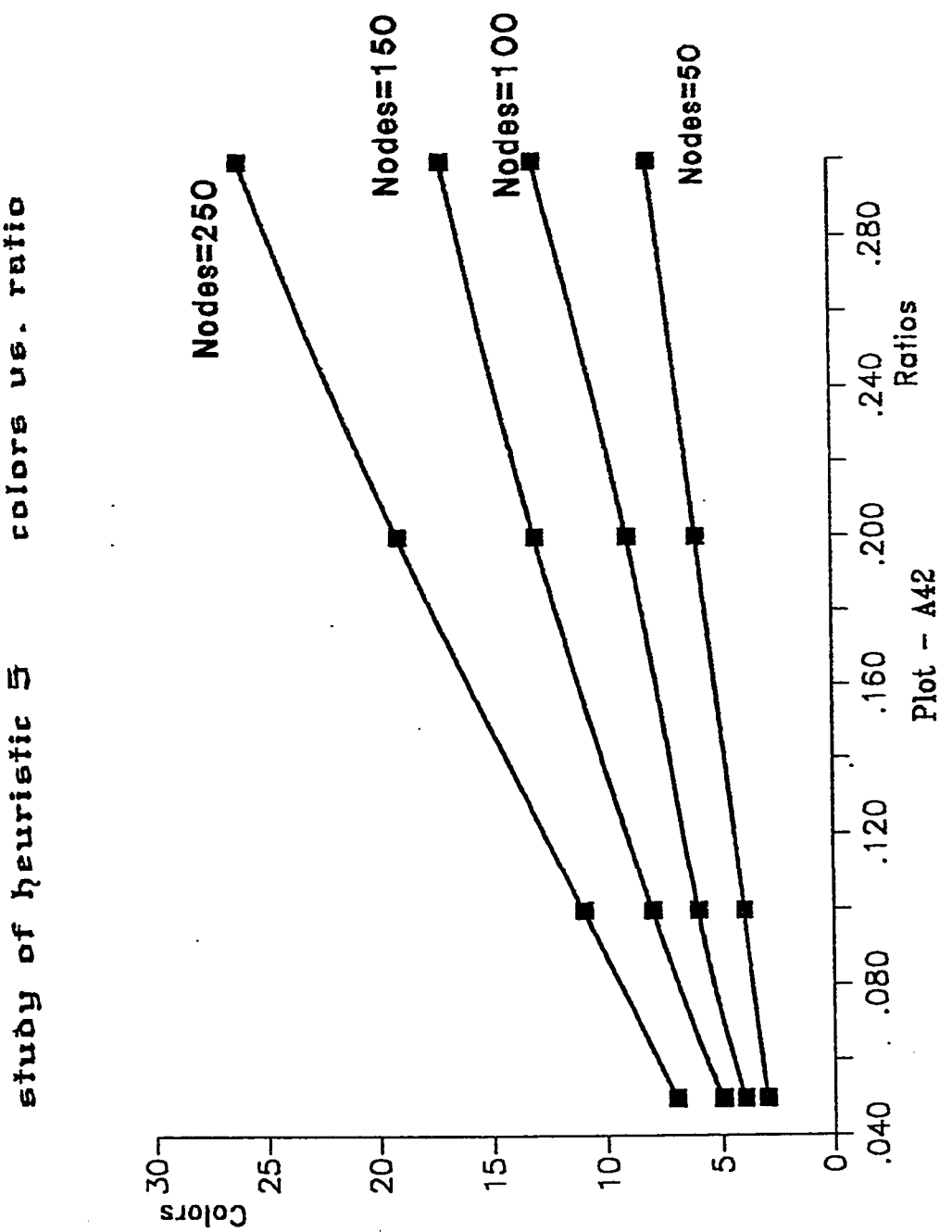

study of heuristic 5 nodes vs. colors
Colors
30
25
20
15
10
5
0
40
60
80
100
120
140
160
180
200
220
240
260
Nodes
Ratio=0.3
Ratio=0.2
Ratio=0.1
Ratio=0.05
Plot - A37


Study of heuristic 1 colors vs. ratio
Nodes=250
Nodes=150
Nodes=100
Nodes=50
Colors
30
25
20
15
10
5
0
.040
.080
.120
.160
.200
.240
.280
Ratios
Plot - A38


Study of heuristic 2 colors vs. ratio
Colors
30
25
20
15
10
5
0
.040
.080
.120
.160
.200
.240
.280
Ratios
Nodes=250
Nodes=150
Nodes=100
Nodes=50
Plot - A39


study of heuristic 3 colors vs. ratio
Nodes=250
Nodes=150
Nodes=100
Nodes=50
Colors
25
20
15
10
5
0
.040
.080
.120
.160
.200
.240
.280
Ratios
Plot - A40


study of heuristic 4 colors vs. ratio
Nodes=250
Nodes=150
Nodes=100
Nodes=50
Colors
30
25
20
15
10
5
0
.040
.080
.120
.160
.200
.240
.280
Ratios
Plot - A41


study of heuristic 5 colors us. ratio
Nodes=250
Nodes=150
Nodes=100
Nodes=50
Colors
30
25
20
15
10
5
0
.040
.080
.120
.160
.200
.240
.280
Ratios
Plot - A42

CHAPTER 7

HYBRID ALGORITHM

This chapter develops a hybrid algorithm extracting the best of all the five heuristics. Towards that, we will utilize the test results from the previous chapter summarized in the ranking file.

7.1 Hybrid Algorithm

From the previous results, summarized in the *color ranking file* and the *CPU time ranking file*, it can be seen that in different intervals, different algorithms have different performance. In each interval, the different algorithms were ranked. This fact will be used to get the best results of each interval, by applying the corresponding algorithm.

The Hybrid algorithm reads the graph to be colored, characterizes it by computing its size and ratio. Size is provided to it by the user and ratio is calculated by counting all the edges in the graph and applying the ratio formula. The ratio calculating formula is:

$$Ratio = edges * 2 / nodes / nodes$$

Now the ranking file is consulted to determine the interval of the given graph,

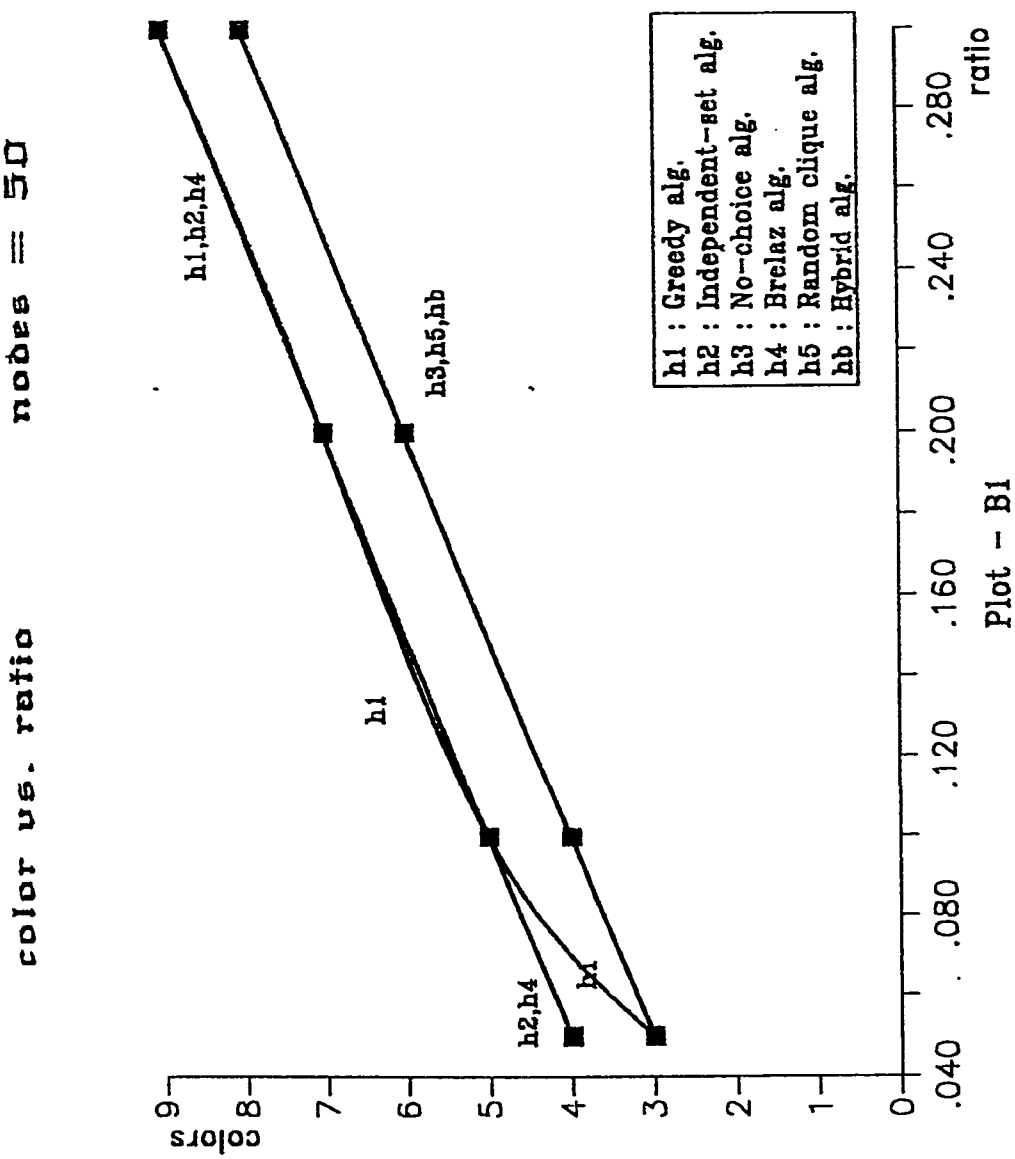
using the calculated size and ratio. Once this interval is allocated the first algorithm in the rank is most likely the best to run. Since this may not always be the best for each instance, the subsequent algorithms in the ranked order are also tried in order for better results.

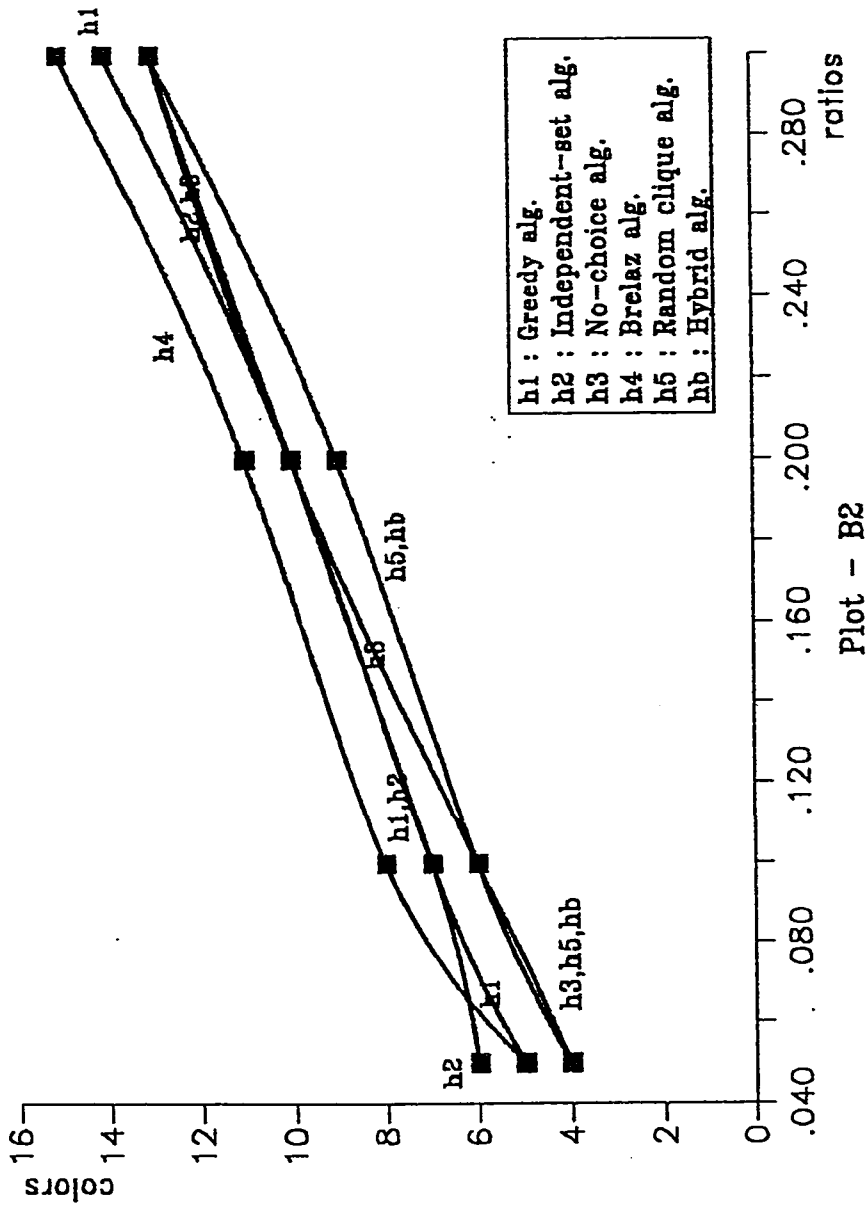
This idea can be used for best coloring using the color ranking file or best CPU time using the CPU time ranking file.

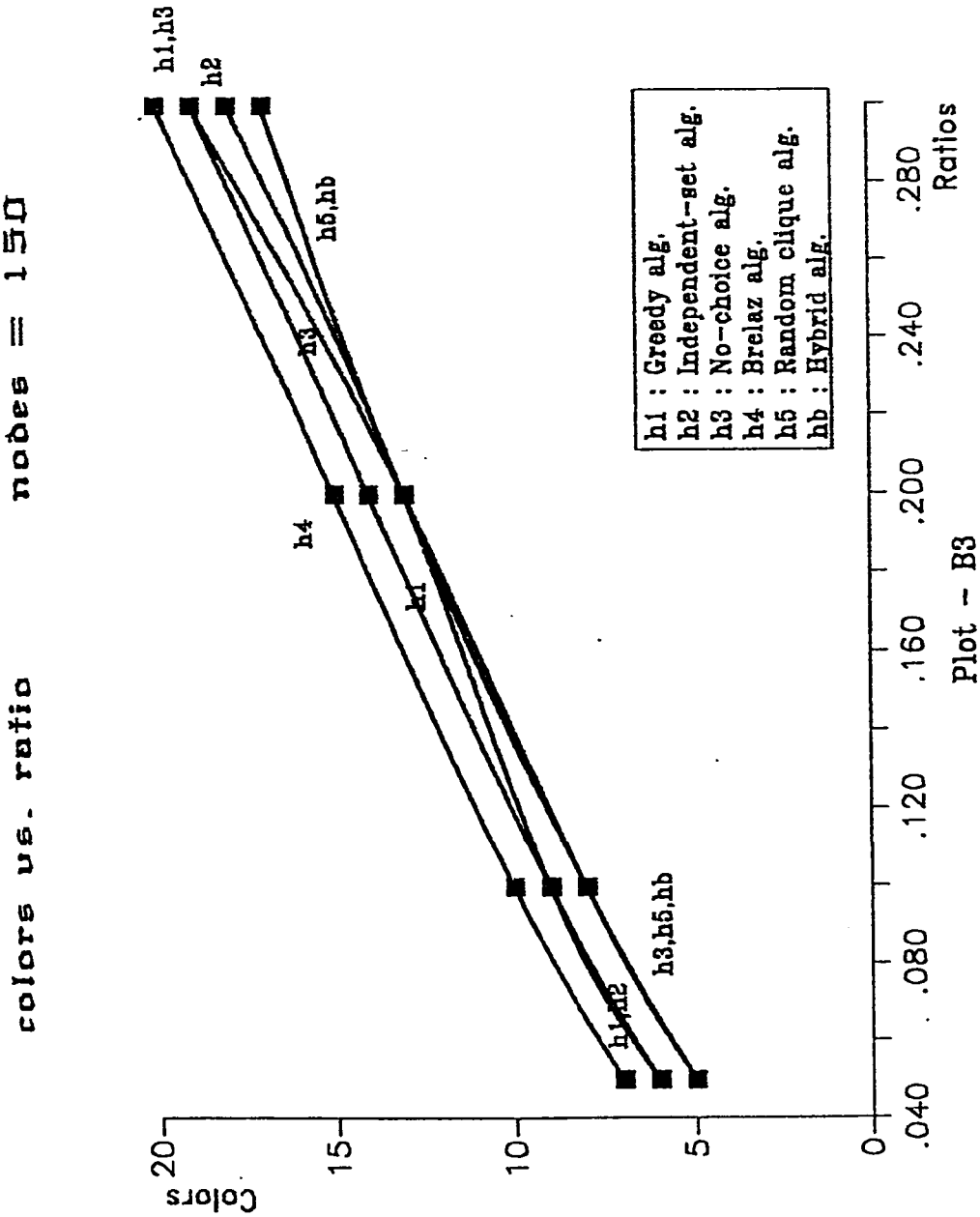
7.2 Test Results

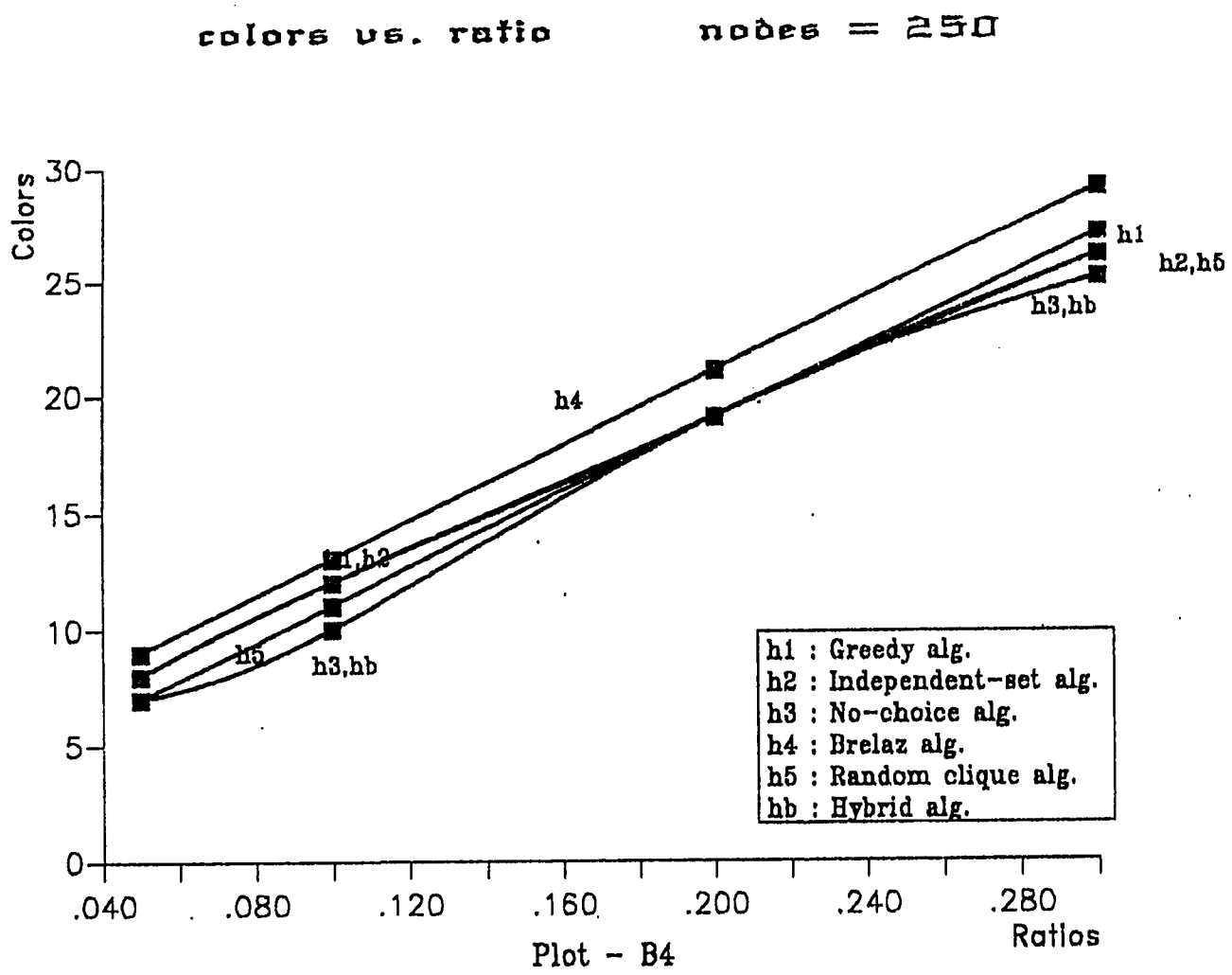
Tests have been made on the Hybrid algorithm. These tests used the same graph sizes and ratios used for the other algorithms. The results show that the Hybrid algorithm achieves the best results of all.

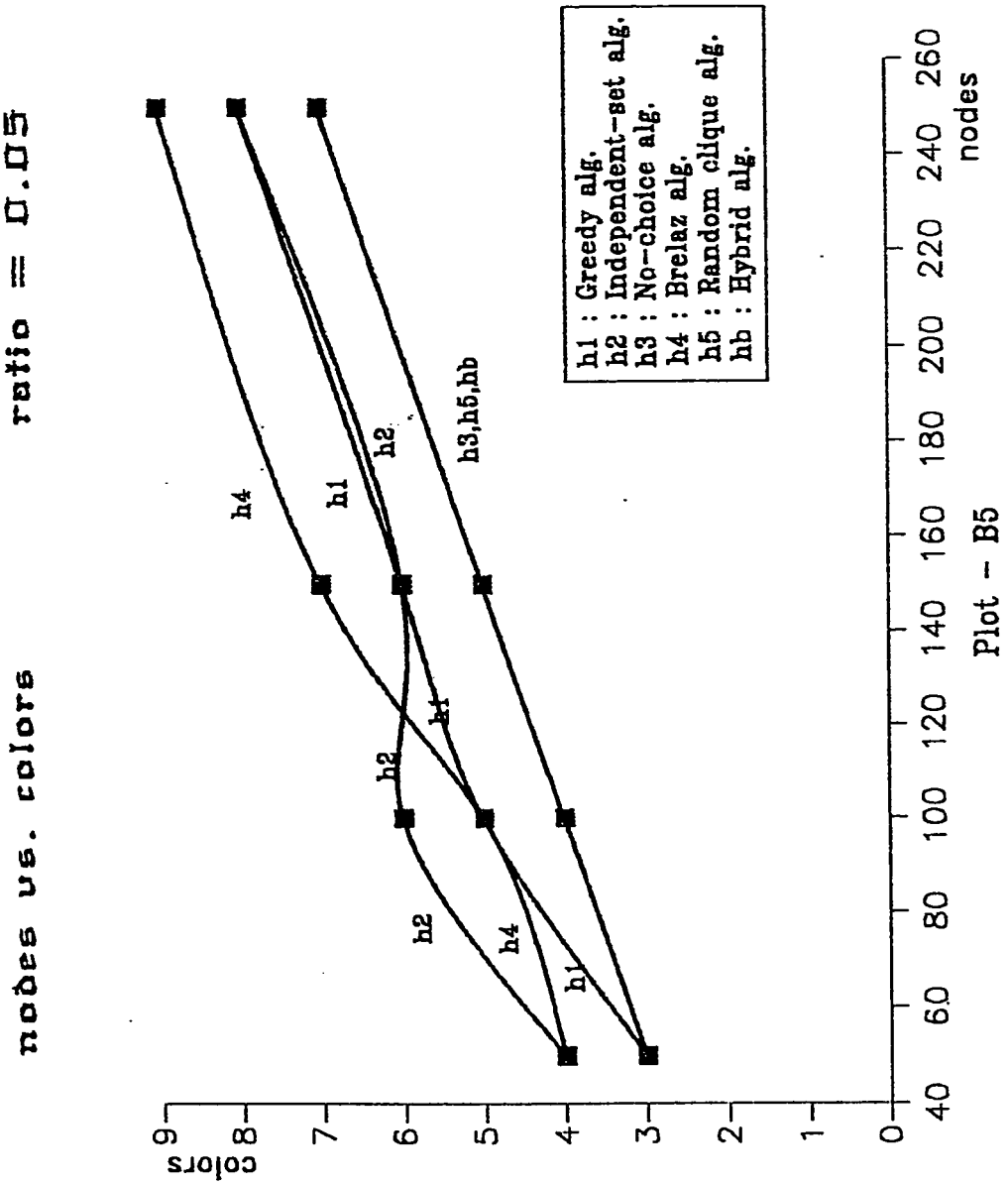
Plots B1-B4 show the effect of ratio on the coloring. Plots B5-B8 show the effect of graph size on the coloring. Plots B9-B12 show the effect of graph size on the CPU time. Plots B13-B16 show the effect of ratio on the coloring. Each plot contains six curves. Five of the curves are the same as the ones shown earlier in the last chapter and the new one is that of the Hybrid algorithm. These plots show how the Hybrid algorithm behaves better than all the other algorithms. Each point in the plot is the average of twenty different simulations of the same graph specification. This means that for very few instances, if any, the Hybrid algorithm might not select the best first, it selects the best algorithm for most graphs of that specifications.

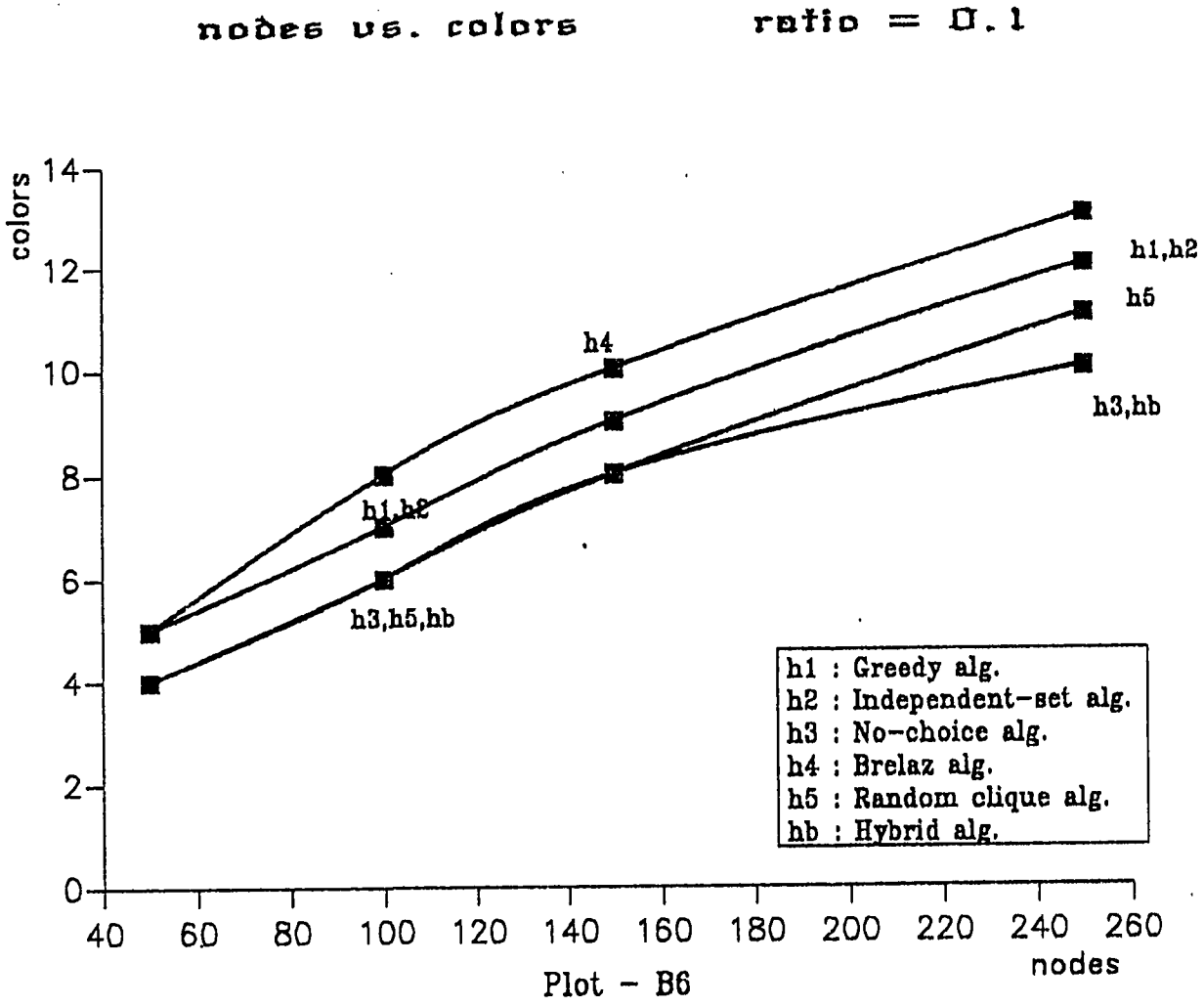

color vs. ratio
nodes = 50
colors
ratio
Plot - B1
h1,h2,h4
h3,h5,hb
h1
h2,h4
h3
h1 : Greedy alg.
h2 : Independent-set alg.
h3 : No-choice alg.
h4 : Brelaz alg.
h5 : Random clique alg.
hb : Hybrid alg.

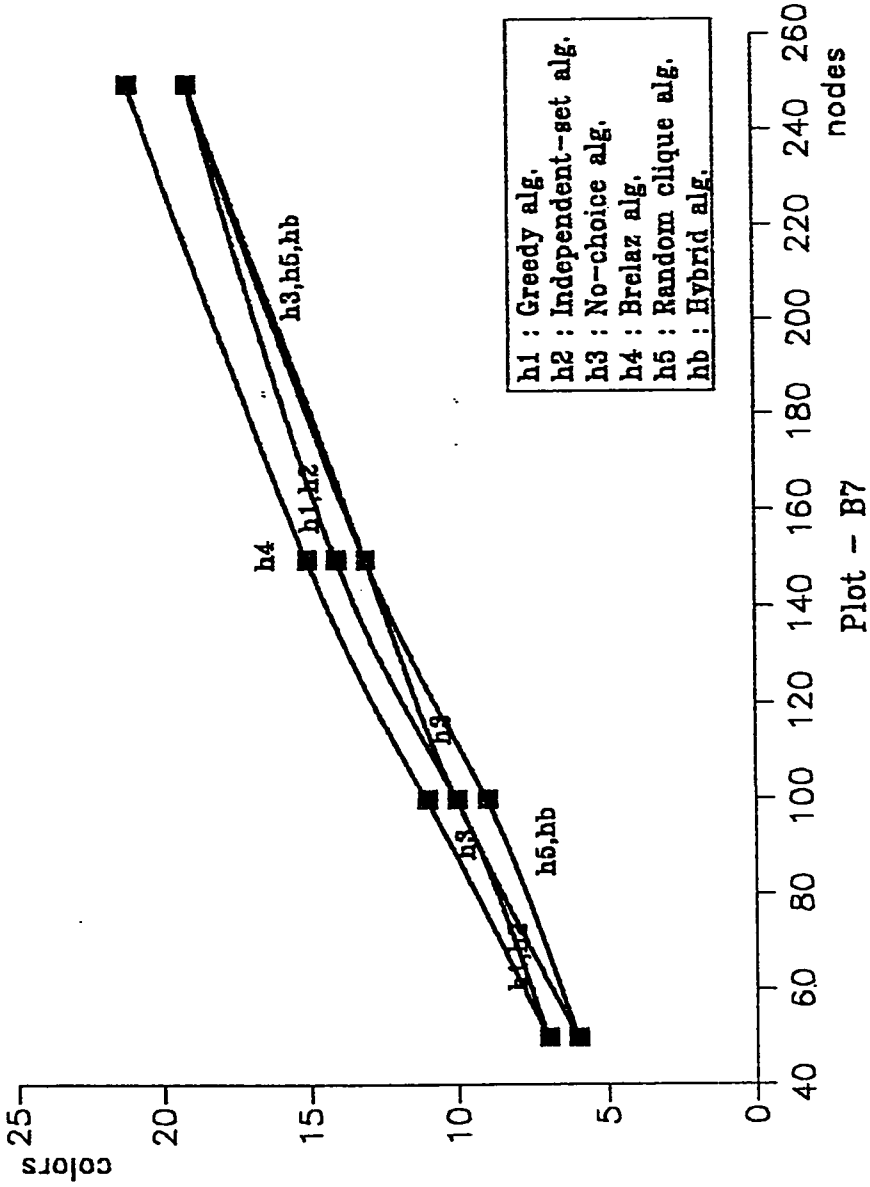

colors vs. ratio
nodes = 100
colors
ratios
Plot - B2
h1 : Greedy alg.
h2 : Independent-set alg.
h3 : No-choice alg.
h4 : Brelaz alg.
h5 : Random clique alg.
hb : Hybrid alg.
h1
h4
h5,hb
h1,h2
h2
h3,h5,hb
0
2
4
6
8
10
12
14
16
.040
.080
.120
.160
.200
.240
.280

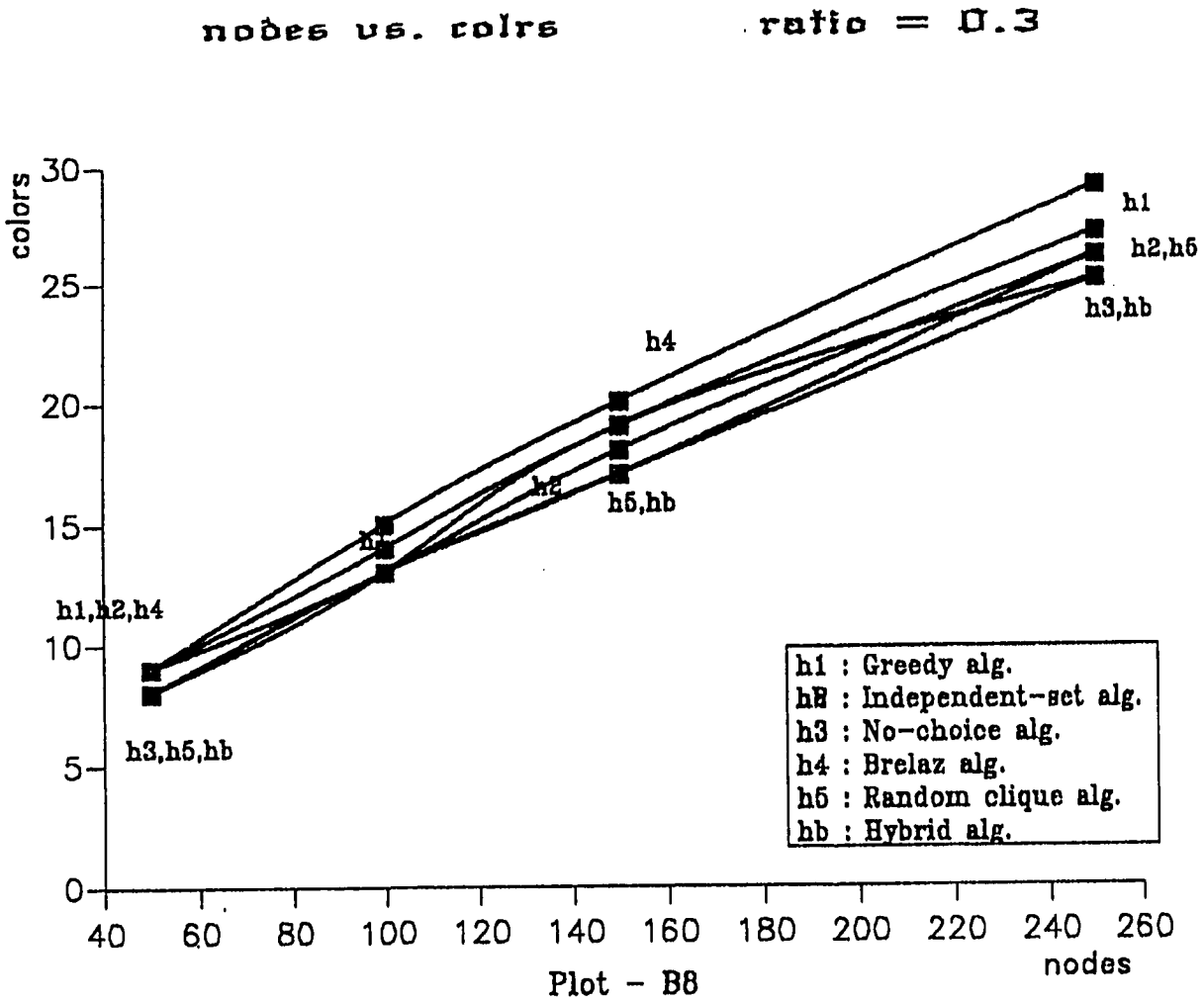

colors vs. ratio
nodes = 150
Colors
20
15
10
5
0
.040
.080
.120
.160
.200
.240
.280
Ratios
Plot - B3
h1,h3
h2
h5,hb
h3
h4
h1
h1,h2
h3,h5,hb
h1 : Greedy alg.
h2 : Independent-set alg.
h3 : No-choice alg.
h4 : Brelaz alg.
h5 : Random clique alg.
hb : Hybrid alg.

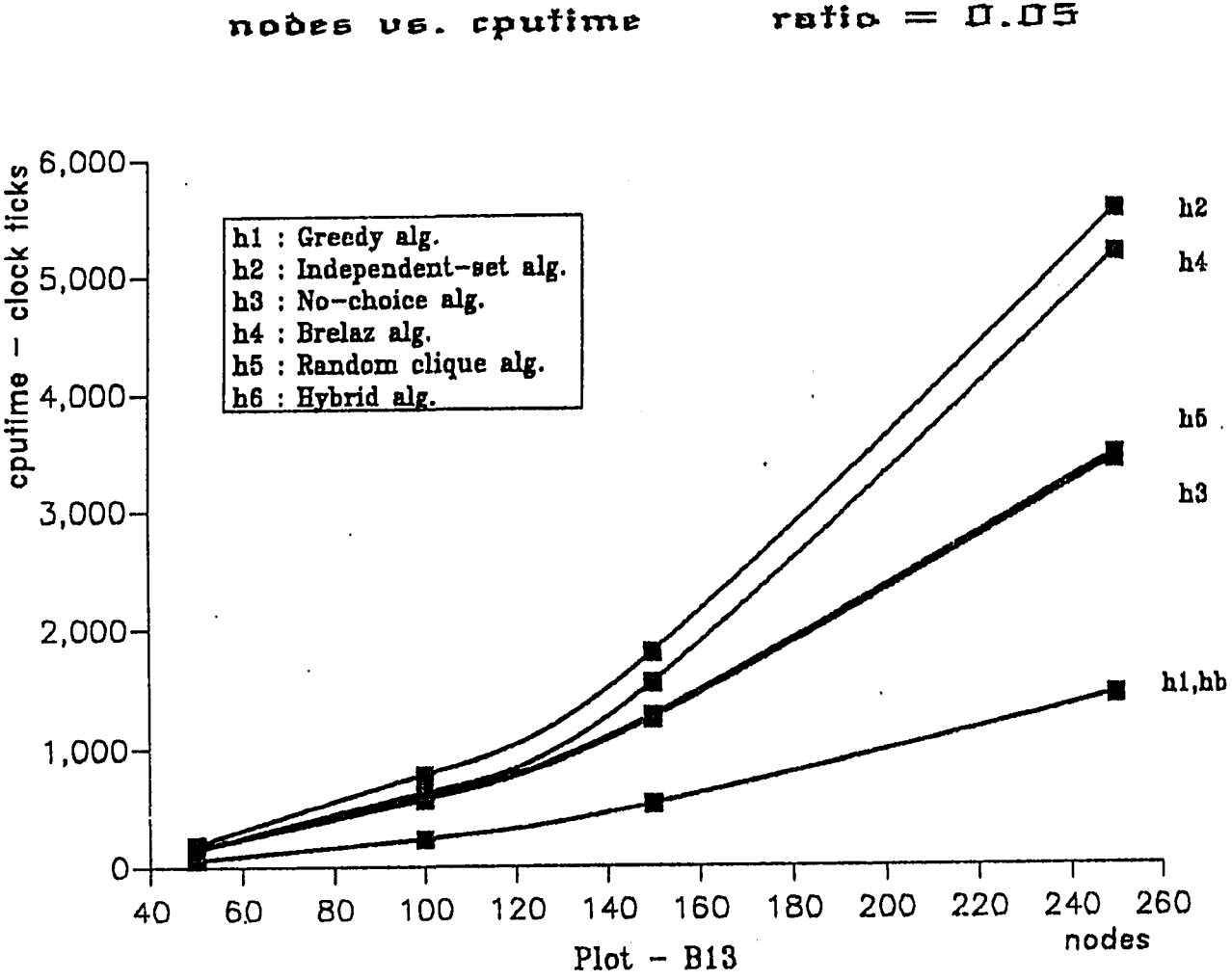

colors vs. ratio
nodes = 250
Colors
30
25
20
15
10
5
0
.040
.080
.120
.160
.200
.240
.280
Ratios
Plot - B4
h1
h2,h5
h3,hb
h4
h1,h2
h5
h3,hb
h1 : Greedy alg.
h2 : Independent-set alg.
h3 : No-choice alg.
h4 : Brelaz alg.
h5 : Random clique alg.
hb : Hybrid alg.

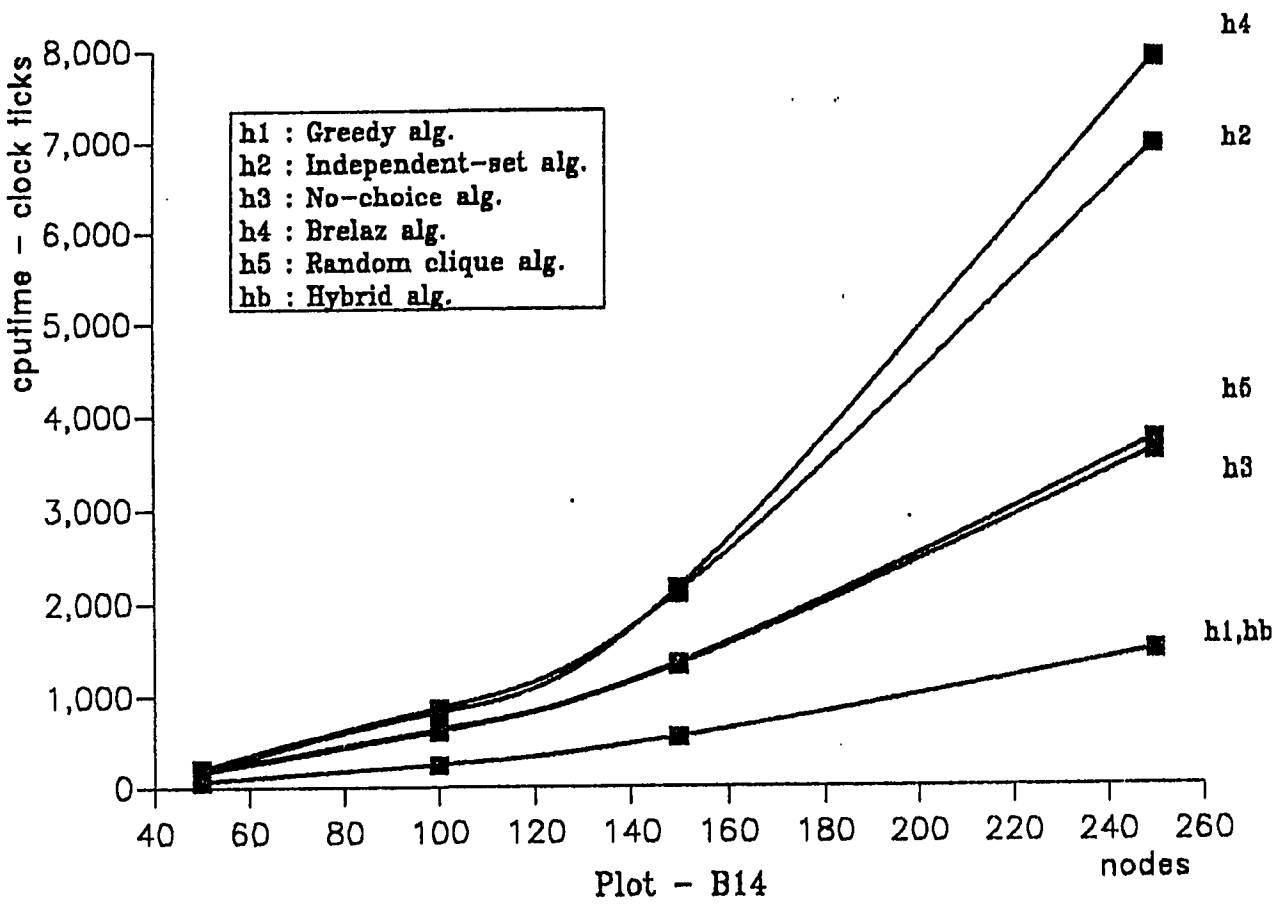

nodes vs. colors
ratio = 0.05
colors
nodes
Plot - B5
h1 : Greedy alg.
h2 : Independent-set alg.
h3 : No-choice alg.
h4 : Brelaz alg.
h5 : Random clique alg.
hb : Hybrid alg.
h1
h2
h3,h5,hb
h4

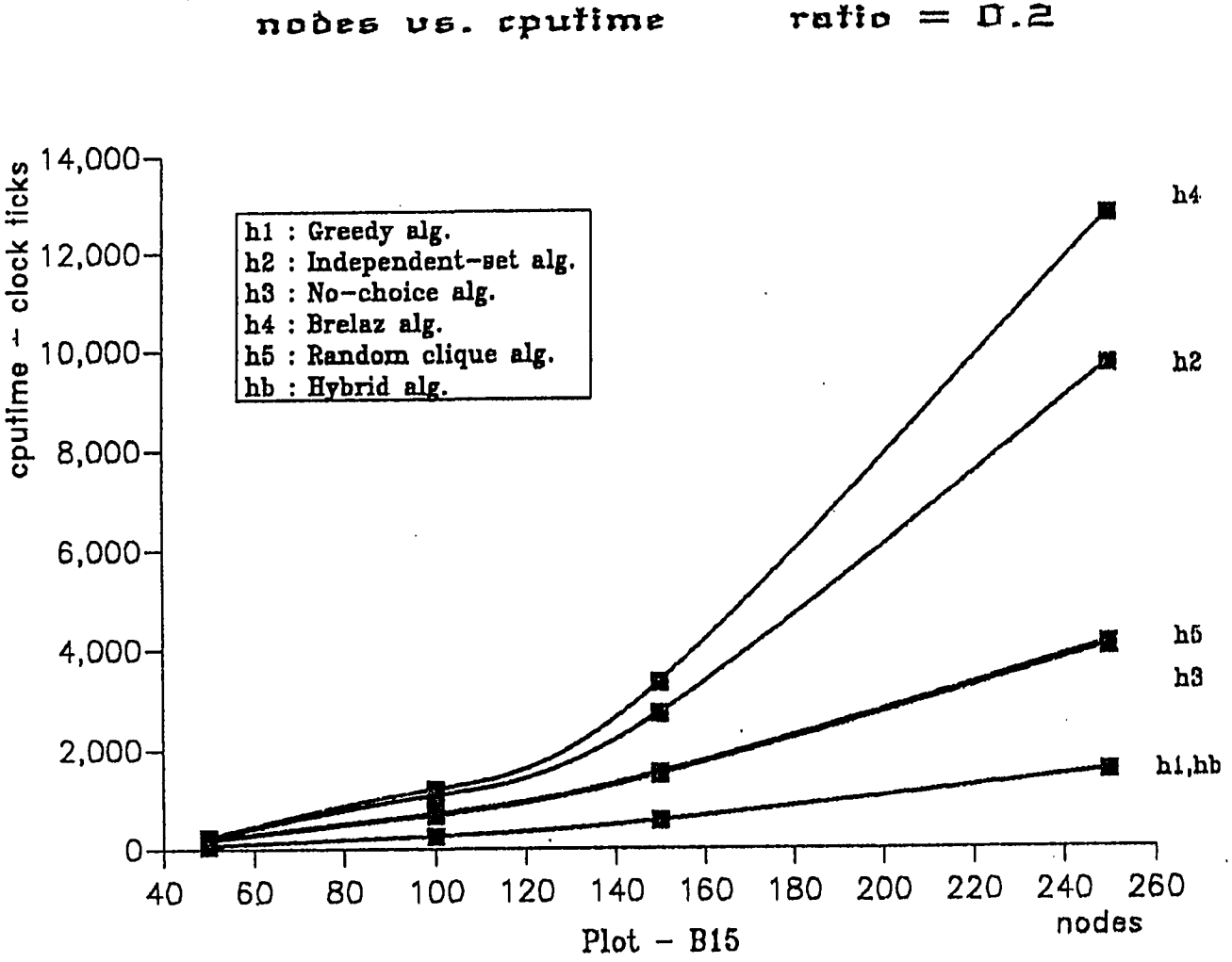

nodes vs. colors
ratio = 0.1
colors
14
12
10
8
6
4
2
0
40
60
80
100
120
140
160
180
200
220
240
260
nodes
h1,h2
h5
h4
h3,hb
h1,h2
h3,h5,hb
h1 : Greedy alg.
h2 : Independent-set alg.
h3 : No-choice alg.
h4 : Brelaz alg.
h5 : Random clique alg.
hb : Hybrid alg.
Plot - B6

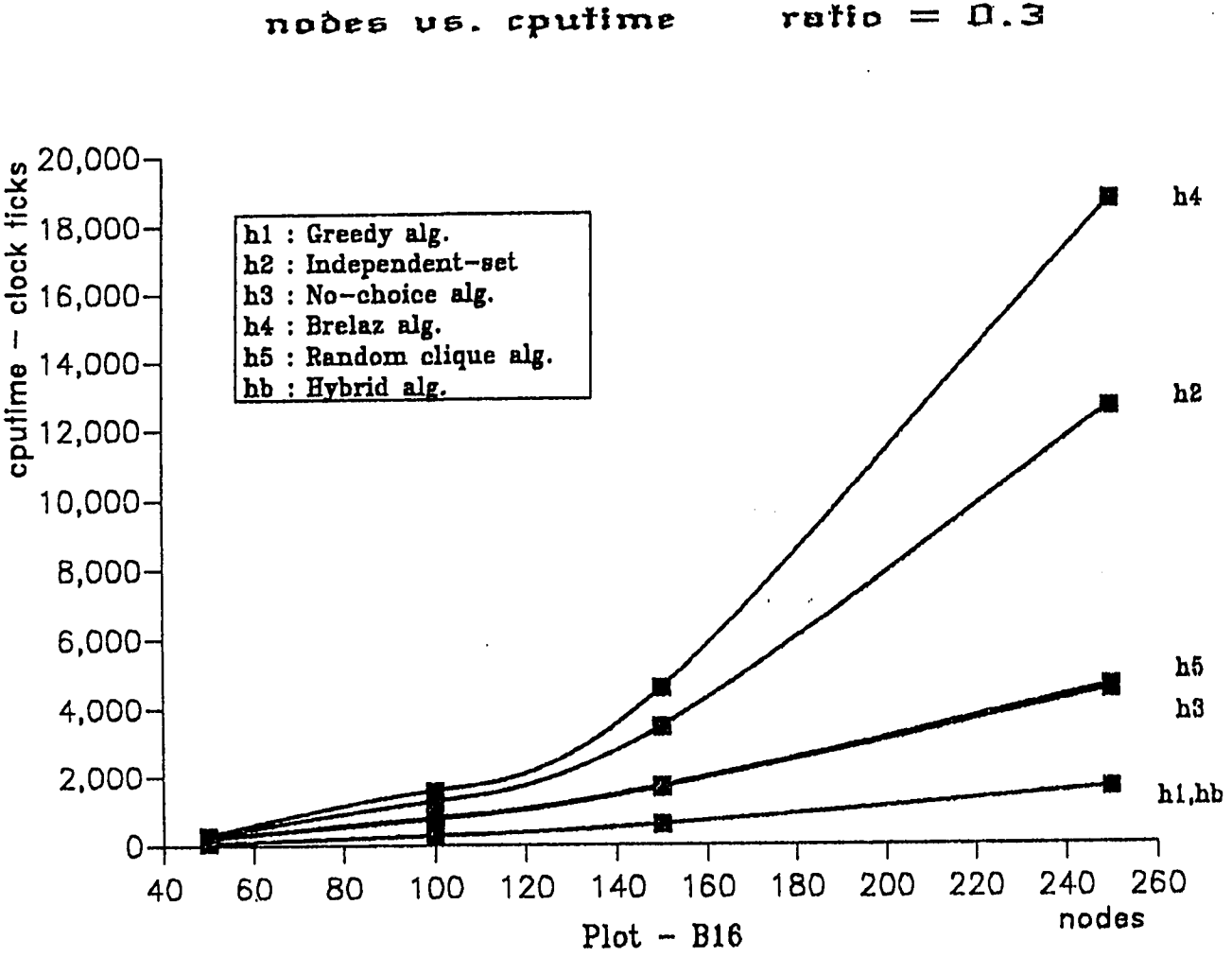

nodes vs. colors
ratio = 0.2
colors
nodes
Plot - B7
h1 : Greedy alg.
h2 : Independent-set alg.
h3 : No-choice alg.
h4 : Brelaz alg.
h5 : Random clique alg.
hb : Hybrid alg.
h4
h1,h2
h3,h5,hb
h3
h5,hb
0
5
10
15
20
25
40
60
80
100
120
140
160
180
200
220
240
260

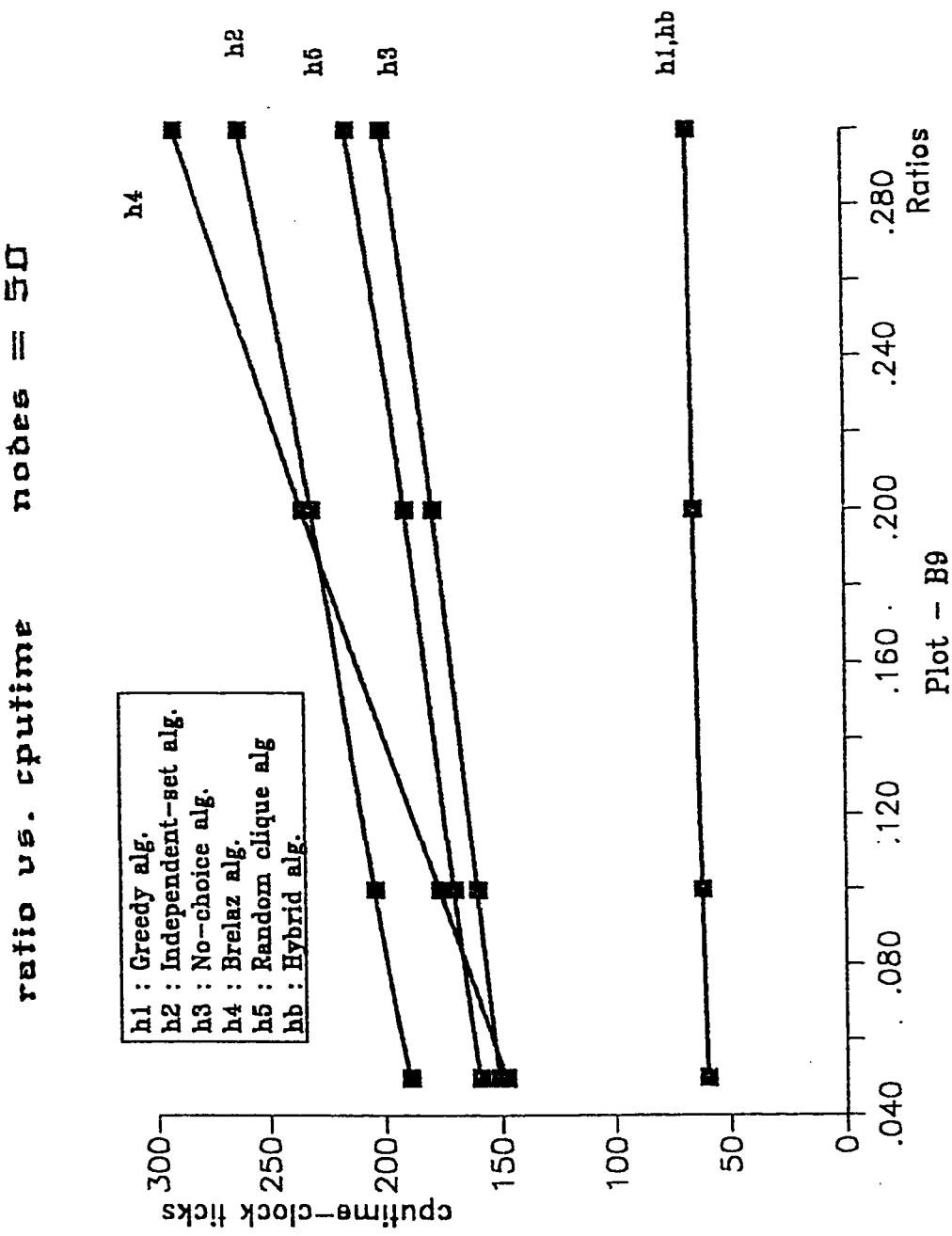

nodes vs. colrs
ratio = 0.3
colors
30
25
20
15
10
5
0
40
60
80
100
120
140
160
180
200
220
240
260
nodes
Plot - B8
h1
h2,h5
h3,hb
h4
h5,hb
h1,h2,h4
h3,h5,hb
h1 : Greedy alg.
h2 : Independent-set alg.
h3 : No-choice alg.
h4 : Brelaz alg.
h5 : Random clique alg.
hb : Hybrid alg.

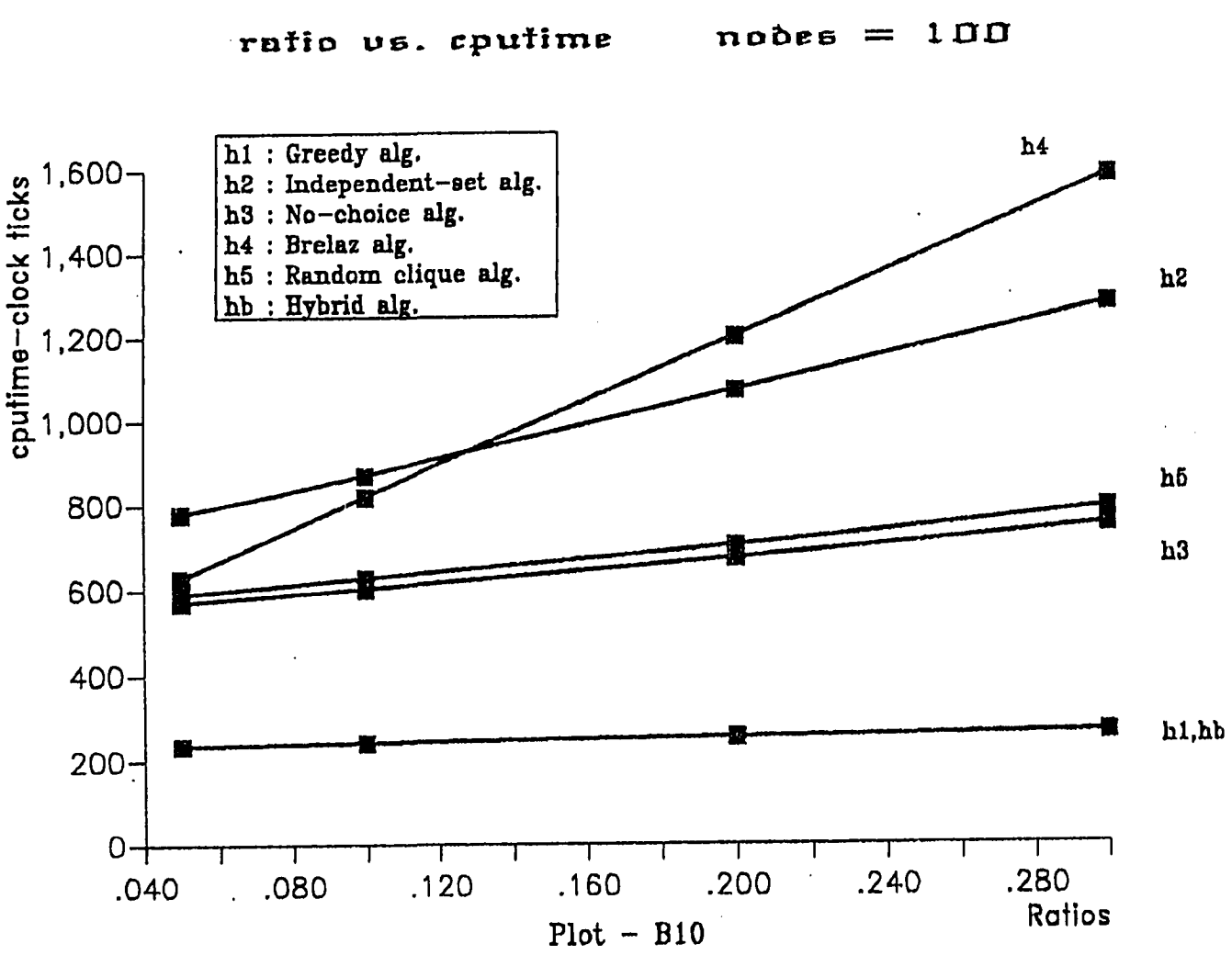

nodes vs. cputime
ratio = 0.05
cputime - clock ticks
6,000
5,000
4,000
3,000
2,000
1,000
0
h1 : Greedy alg.
h2 : Independent-set alg.
h3 : No-choice alg.
h4 : Brelaz alg.
h5 : Random clique alg.
h6 : Hybrid alg.
h2
h4
h5
h3
h1,hb
40
60
80
100
120
140
160
180
200
220
240
260
nodes
Plot - B13

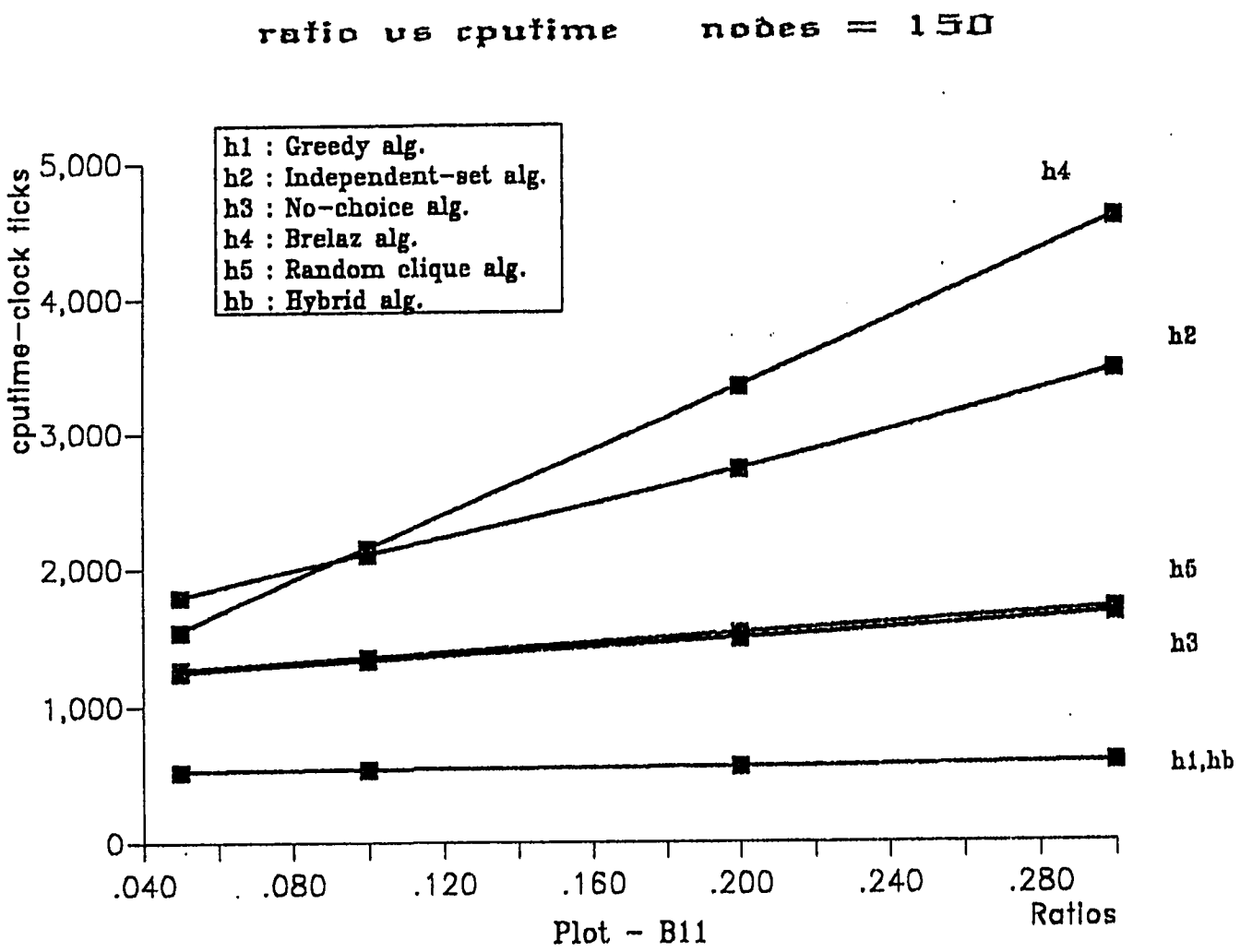

nodes vs. cputime ratio = 0.1
cputime - clock ticks
8,000
7,000
6,000
5,000
4,000
3,000
2,000
1,000
0
h1 : Greedy alg.
h2 : Independent-set alg.
h3 : No-choice alg.
h4 : Brelaz alg.
h5 : Random clique alg.
hb : Hybrid alg.
h4
h2
h5
h3
h1,hb
40 60 80 100 120 140 160 180 200 220 240 260
nodes
Plot - B14

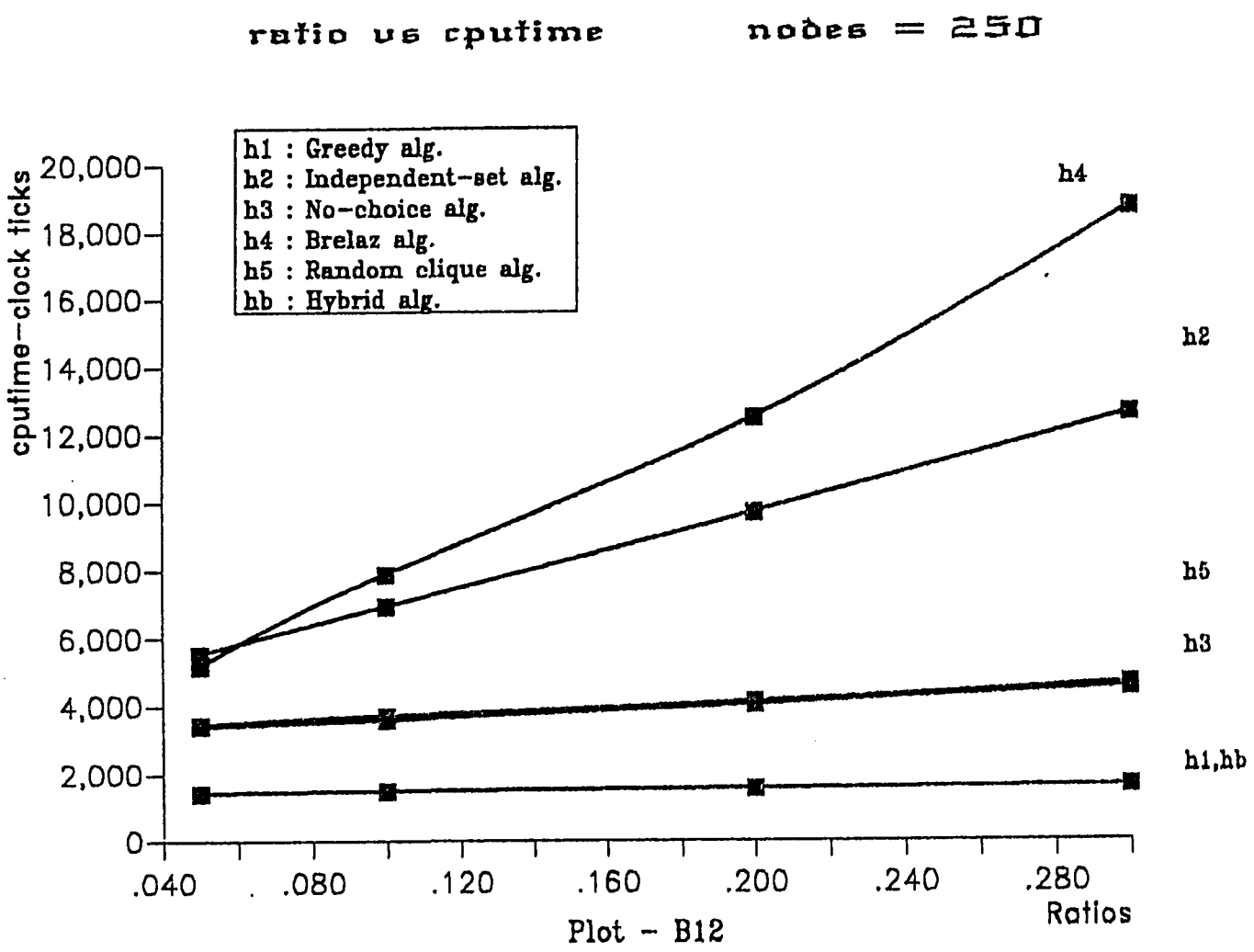

nodes vs. cputime ratio = 0.2
cputime - clock ticks
14,000
12,000
10,000
8,000
6,000
4,000
2,000
0
h1 : Greedy alg.
h2 : Independent-set alg.
h3 : No-choice alg.
h4 : Brelaz alg.
h5 : Random clique alg.
hb : Hybrid alg.
h4
h2
h5
h3
h1,hb
40 60 80 100 120 140 160 180 200 220 240 260
nodes
Plot - B15


nodes vs. cputime ratio = 0.3
cputime - clock ticks
20,000
18,000
16,000
14,000
12,000
10,000
8,000
6,000
4,000
2,000
0
h1 : Greedy alg.
h2 : Independent-set
h3 : No-choice alg.
h4 : Brelaz alg.
h5 : Random clique alg.
hb : Hybrid alg.
h4
h2
h5
h3
h1,hb
40
60
80
100
120
140
160
180
200
220
240
260
nodes
Plot - B16


ratio vs. cputime nodes = 50
h1 : Greedy alg.
h2 : Independent-set alg.
h3 : No-choice alg.
h4 : Brelaz alg.
h5 : Random clique alg
hb : Hybrid alg.
cputime-clock ticks
300
250
200
150
100
50
0
.040
.080
.120
.160
.200
.240
.280
Ratios
Plot - B9
h4
h2
h5
h3
h1,hb


ratio vs. cputime nodes = 100
h1 : Greedy alg.
h2 : Independent-set alg.
h3 : No-choice alg.
h4 : Brelaz alg.
h5 : Random clique alg.
hb : Hybrid alg.
cputime-clock ticks
1,600
1,400
1,200
1,000
800
600
400
200
0
.040
.080
.120
.160
.200
.240
.280
Ratios
Plot - B10
h4
h2
h5
h3
h1,hb


ratio vs cputime nodes = 150
h1 : Greedy alg.
h2 : Independent-set alg.
h3 : No-choice alg.
h4 : Brelaz alg.
h5 : Random clique alg.
hb : Hybrid alg.
cputime-clock ticks
5,000
4,000
3,000
2,000
1,000
0
.040
.080
.120
.160
.200
.240
.280
Ratios
h4
h2
h5
h3
h1,hb
Plot - B11


ratio vs cputime nodes = 250
h1 : Greedy alg.
h2 : Independent-set alg.
h3 : No-choice alg.
h4 : Brelaz alg.
h5 : Random clique alg.
hb : Hybrid alg.
cputime-clock ticks
20,000
18,000
16,000
14,000
12,000
10,000
8,000
6,000
4,000
2,000
0
.040
.080
.120
.160
.200
.240
.280
Ratios
Plot - B12
h4
h2
h5
h3
h1,hb

7.3 Implementation of Hybrid Algorithm

The algorithm is written in pascal. The program is user friendly. It asks the user whether the ranking preferred is best color ranking or best CPU time ranking. It also asks if there is a limit in the number of colors required by the user or a limit in the CPU time required.

The Hybrid algorithm then reads the graph, characterizes by calculating its ratio and searches the ranking file requested. The algorithm numbers are placed in the Rank (I) array. Using the case statement as follows the best algorithm is selected, and the results are printed.

ALGORITHM HYBRID-COLORING

Read Graph

Determine the characteristics of the graph

Using the experimentally determined rank file (color or

CPU time) and the characteristics of the given graph

determine the rank of individual algorithm

(Rank [1-5] contains this information)

Case Rank (I)

Greedy: call Greedy

Independent-set: call Independent-set

No-Choice: call No-choice

Brelaz: call Brelaz

Random-Clique: call Random-clique

End case

Print colors

The rest of the algorithms in the rank are also tested for better results. If no algorithm can give accepted colorings, maximum number of colors is exceeded, node splitting is carried out to get better results.

CHAPTER 8

CONCLUSIONS

Two major tasks were covered in this thesis. The first one is the analysis, design and implementation of an information system for the semester credit system (SCS). The second is study and analysis of different algorithms that solves the examination scheduling problem. Two packages were developed to support the work. The first is registration system built on a Dbase VI system, and the second is a hybrid algorithm that supersedes the other graph coloring algorithms.

The first two chapters are about the analysis and design phase of the new SCS system. The first Chapter introduces the system by giving an informal description of the system and its importance. The second Chapter discusses the current system, the proposed system and the design and implementation of the new system. The phases of requirement analysis, conceptual design, logical design and physical design are covered.

The second part of this thesis was covered in Chapters three to seven. Chapter three introduces the examination scheduling problem as a timetabling problem. It also formulates it as a coloring problem. Chapter four introduces five coloring heuristics to solve the examination scheduling problem. Chapter five adds some enhancements to coloring concepts such as node splitting,

preassignments, priority ordering of nodes and node spilling. Some of these ideas such as node splitting was added to the five heuristics in the development of a testing system that was discussed in Chapter six. Test results were drawn on plots. Plots were analyzed, and intervals where algorithms change their rankings, were detected. Two files of algorithm-rankings were produced. One ranks the algorithms with respect to number of colors required and the others ranks them with CPU time required. In Chapter seven a hybrid algorithm was proposed and implemented. Test results were plotted. Analysis proved the superiority of this new algorithm.

A number of scheduling problems can also use the hybrid algorithm. These include scheduling operation rooms in hospitals, airplanes to gates, etc.

This algorithm also allows for other coloring heuristic algorithms to be part of it if they are analyzed and added to the ranking files.

APPENDIX

Inputs to Current System:

The current implementation of SCS standardizes its inputs by making use of the following forms.

1. *Admission form:* as the student is admitted to school he fills out a form which requires some information about his personal status, such as, name, phone, parents job etc.

2. *Courses offered:* is an output of the scheduling subsystems. It includes the courses offered in a semester, its credits, periods offered, room and the instructor assignment.

3. *Registration form:* A student fills out this form with the courses he intends to take in a semester. The form should be approved by the instructors of the corresponding courses as the student registers. This form includes some information such as course number, time offered, instructor's name.

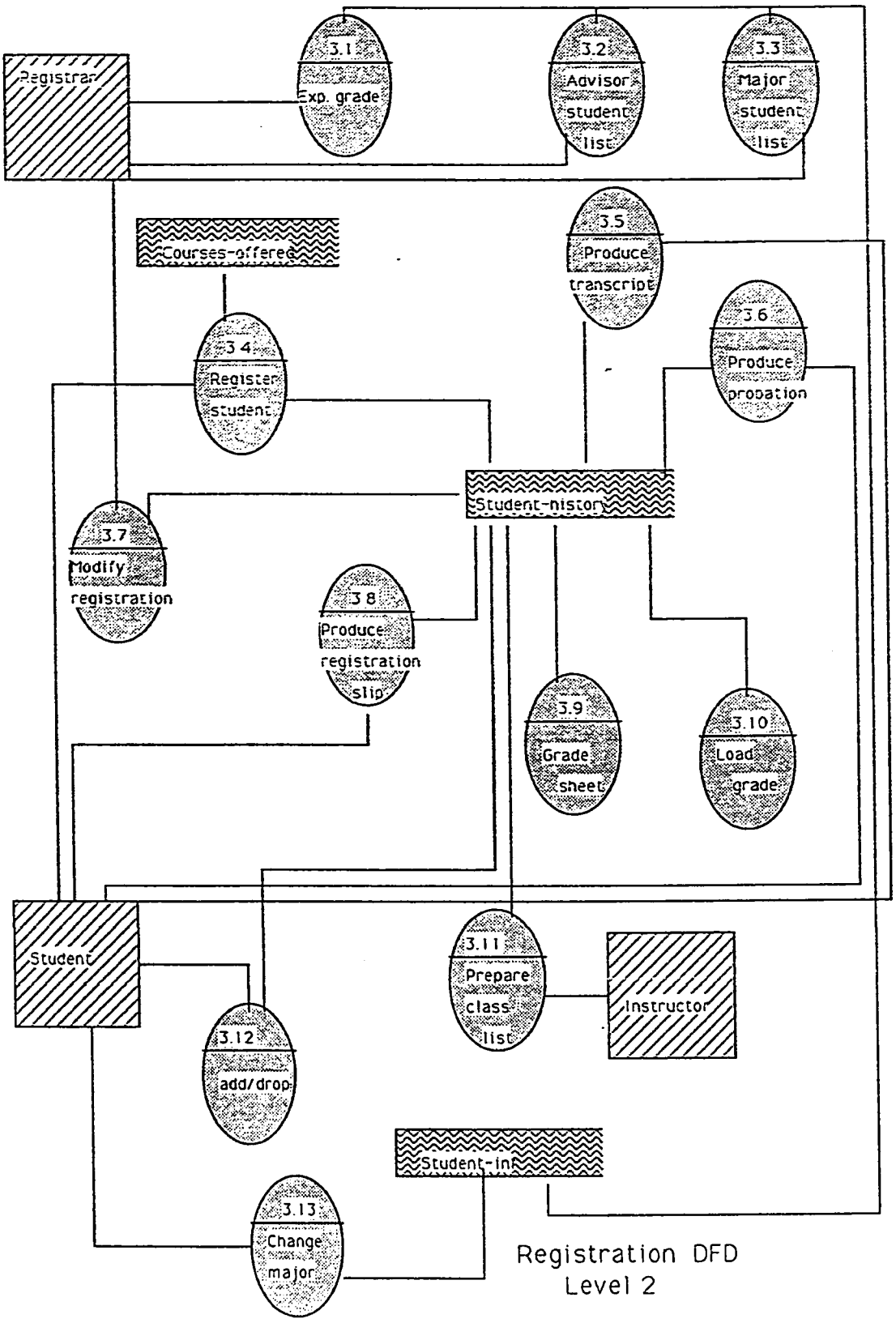
4. *Drop form:* The student fills out this form if he intends to drop a course. The drop form should be approved by the advisor and then by the instructor of the course.

5. *Class grade sheet:* It includes a list of student names in a class, their numbers, midterm test grades, quiz accumulative grades, final test grades,

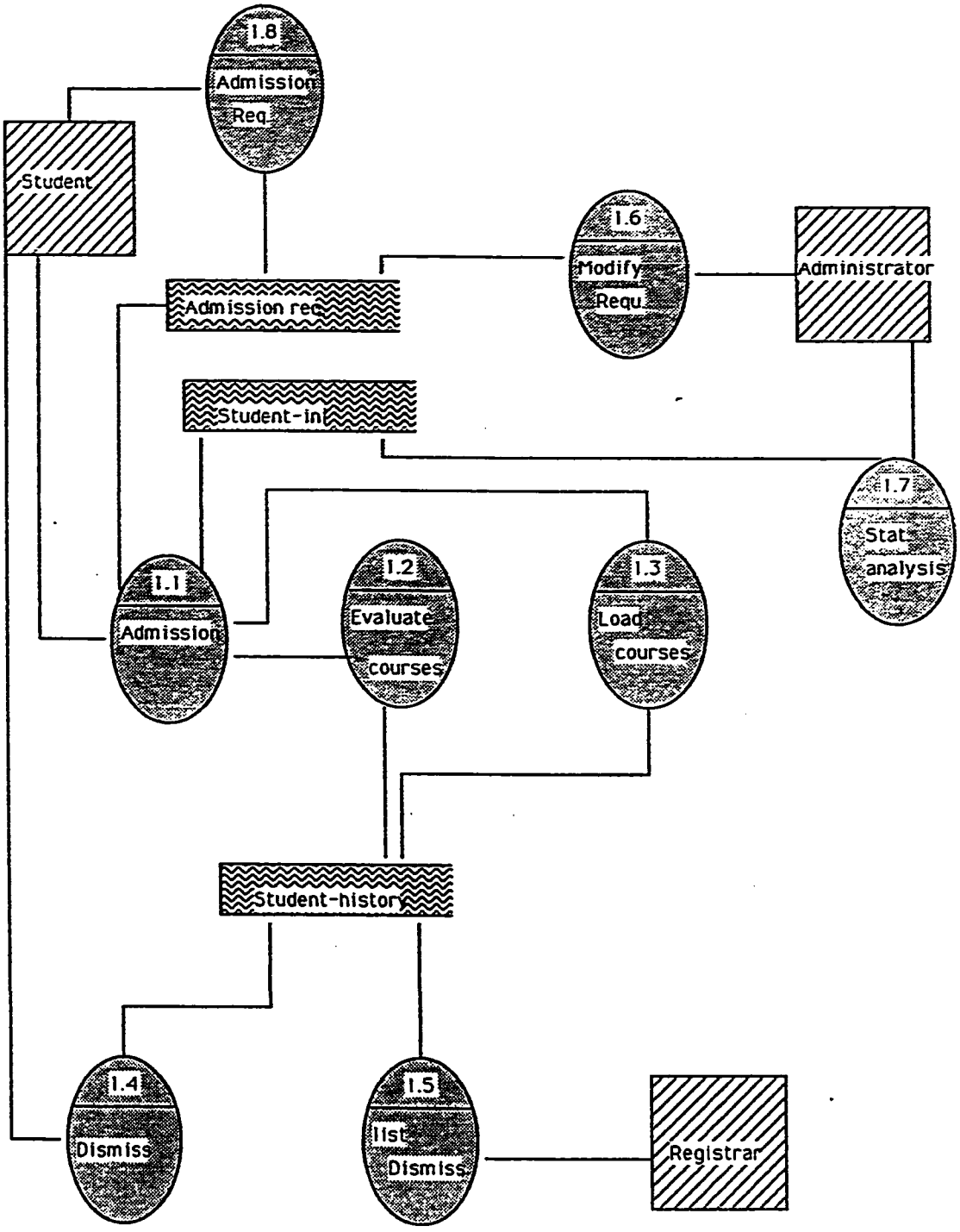
and the total grades.

6. *List of students in a schedule:* It includes the names of the students registered in a certain schedule (set of sessions that are set for newly enrolled students).

7. *A schedule:* Newly enrolled students usually do not understand the SCS. Therefore, the school registers them in groups, each of which is called a schedule.

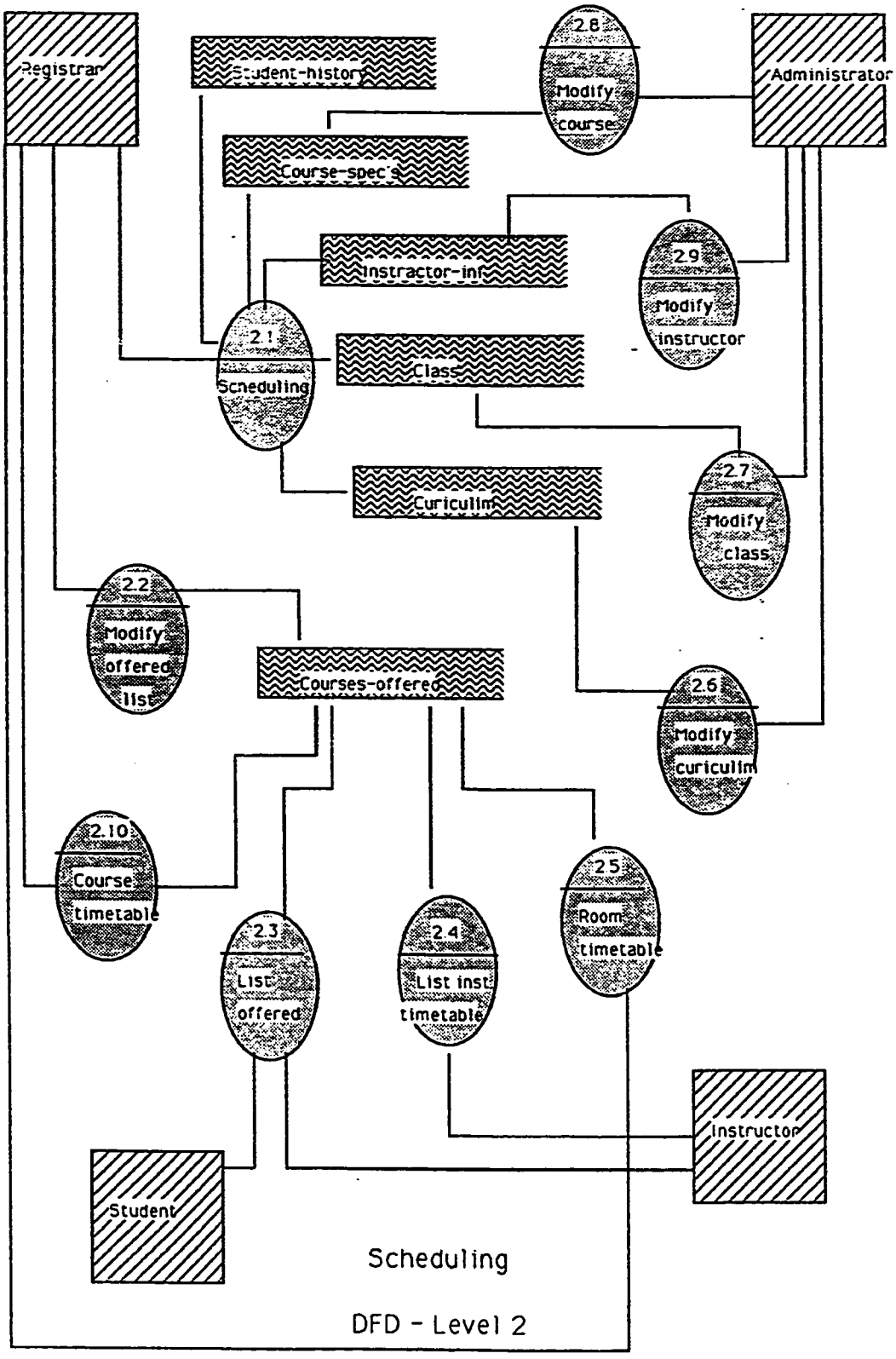


Registration DFD
Level 2



Admission

DFD - Level 2


Registrar
Student-history
2.8
Modify course
Administrator
Course-spec's
Instractor-inf
2.9
Modify instructor
2.1
Scheduling
Class
Curiculim
2.7
Modify class
2.2
Modify offered list
Courses-offered
2.6
Modify curiculim
2.10
Course timetable
2.3
List offered
2.4
List inst timetable
2.5
Room timetable
Instructor
Student
Scheduling
DFD - Level 2

Outputs of Current System:

The current system produces the following reports/documents.

1. *Class roster:* This is prepared for each section, by the instructor as he signs the student's registration form to show his students in order of number. It includes their majors, advisors.

2. *List of advisor's students:* This is prepared every time the advisors's list changes. Advisors are assigned by the registrar. It includes students name, ID , G.P.A., etc.

3. *Transcript:* Prepared by advisor at the end of each semester. It includes, for each student, courses the student took in a semester, credit hours, grades, points, G.P.A., etc. Probation or congratulations or none is given with the transcript.

4. *Grades Analyses sheet:* The system ouputs some analysis about grades, their distribution on this sheet.

5. *Honors List/Probation List:* List of honoured or probation students.

6. *Schedule List:* List of students in a schedule.

INPUTS to new system are:

1. *Admission form:* includes student's name, phone, etc.

2. *Completed courses form:* includes evaluated courses and courses taken in similar school

3. *Registration form:* includes courses registered by the student

4. *Drop form:* includes courses dropped by the student

5. *Grade sheet:* includes the grades of students in a section

6. *Course spec's:* includes the course number, name, credits and prerequisites

7. *Instructor info.:* includes instructor number, name, speciality and date of recruitment

8. *Class info.:* includes section number, course number, instructor number, and date offering

9. *Offered courses changes:* includes course number, section number, instructor number

10. *Change student info:* includes student name, address, enrollment date and status

11. *Change registration sheet:* includes student number, course number and semester number

12. *Add form:* includes student number, course number and semester number

The Output of the new system are:

1. *Course list:* list of all courses
2. *Section students:* list of students in a section of a course
3. *Transcript:* semester and graduation transcript
4. *Grade sheet:* list of the students, their grades and analysis of the sheet
5. *List of major students:* list of all students specialized in the selected major
6. *List advisor students:* list of all students that are advised by the selected advisor
7. *Courses offered list:* list all the courses offered in the current semester
8. *Students list:* list of all students in school
9. *Special status student list:* list of all students that carry the selected status
10. *Instructors list:* list of all instructors and their information
11. *Class information list:* list of all classes and their periods
12. *Course list:* list of all courses and their information

REFERENCES:

[ALMO69] ALMOND, M. "An Algorithm for the Constructing University Time-Table," *Computer, J.*, 8 (1966), 331-340.

[ALMO69] ALMOND, M. "A University Faculty Timetable," *Computer J.*, 12 (1969), 215-217.

[AUST73] AUST, R.J., "The School Timetabling Problem," *Ph.D.Thesis*, School of Applied Science, University of Sussex, 1973.

[AUST76] AUST, R.J., "An Improvement Algorithm for School Timetabling, *Computer J.*, 19 (1976), 339-345.

[BARR65] BARRACLOUGH, E. D., "The Application of a Digital Computer to the Construction of Timetables," *Computer J.*, 8 (1965), 136-146.

[BEAL76] BEALE, E. M. L. and J.J. H. Forest, "Global Optimization Using Special Ordered Sets," *Mathematical Prog.* 19 (1976), 52-69.

[BRAY68] BRAY, T. A. and C. Witzgall, "Algorithm 336 'NETFLOW(H)," *Comm. ACM* 11, 9(1968).

[CHOW90] CHOW, F. C. and HENNESSY, J. L., "The Priority-based coloring approach to register allocation," *ACM sigpla notices Vol.* 12 No. 4, 19(1990), 501-536.

[EVEN76] EVEN, S.A., ITAI and A. SHAMIR, "On the Complexity of

Timetable and Multicommodity Flow Problems," *SIAM Journal on Computing, No.* 4, 12 (1976) 691-703.

[FISH81] FISHER, M. L., "Lagrangean Relaxation Methods for Solving Integer Programming Problems," *Management Sci.*, 27, 1 (1981), 1-1.

[FISH84] FISHER, "School Timetabling - A case in large binary integer linear programming," *Management Sci., vol.* 30, *No.*12, (1984).

[GEOF71] GEOFFRION, A. M. "Duality in Nonlinear Programming: A Simplified Application-Oriented Development," *SIAM Rev.* 13, 1 (1971), 1-37. 1-37.

[KARC88] KARCHMER, M. and NAOR, J., "A fast parallel algorithm to color a graph with colors," *Journal of Algorithms* 9, 83-91 (1986).

[LION66] LIONS, J., "Matrix Reduction Using the Hungarian Method for the Generation of School Timetable," *Comm. ACM*, 9 (1966), 349-354.

[MEHT81] MEHTA, N. K, "The Application of a graph coloring method to an examination scheduling problem," *INTERFACES, Vol.* 11, No.5, 10(1981), 57-65.

[MURP87] MURPHY, C. "Timelog an intelligent spreadsheet for school timetabling" *SIGART Newsletter*, July 1987, No.101.

[SAKH89], SAKHER Registration System exhibited 1989 at Al-Gosaibi

Exhibition.

[TURN88] TURNER, J. S., "Almost all k-colorable graphs are easy to color," *Journal of algorithms*, 9, 63-82 (1988)

[ULPH77] ULPH, A. "Notes an Extension to a Result in School Timetabling", *INFOR. Vol.*15, No. 2, June 1977.

[WERR71] WERRA, D., "Construction of School timetables by flow methods," *INFOR. Vol.* 9, No.1, March 1971, p.12-22.

[WERR85] WERRA, D. ,"An Introduction to timetabling," *European Journal of Operational Research* 19 (1985), 151-162.

[WHIT76] WHITE, D.J., "A Note on Faculty Timetabling," *Oper. Res. Quart.*, 26 (1976), 875-878.

[WHIT79] WHITE,D.J., "Towards the construction of optimal examination schedules," *INFOR. Vol.* 17, No. 3, 9(1979), 219-228.

صفحة ٢
التاريخ: ٨١/١٨/٠١

تقرير عن جميع المواد المعروضة

رقم العرض	رقم المادة	الشعبة	الفصل	المدرس	الغرفة	الايام	الفتر
٤٩	فيز١	ا	١	صالح خليفه الحربي	١٤٠س	حشع	٣
٥٠	احص١	ب	١		٢٦٠س	حشع	٣
٥١	دار٧	ا	١	رائد المهيدب	١٣٠	حشع	٣
٥٢	قمد٢	ا	١	خالد اليحي	١٦٠ي	حشع	٣
٥٣	انجل٢	ا	١		١٢٤٠	حشع	٣
٥٤	انجل٤	ب	١	محمد مبين عثمان	١٢٣٠	حشع	٣
٥٥	فن١	ج	١	منير عبدالفتاح	١٣٠ي	س ن	٣
٥٦	بدن٢	ج	١	ابراهيم احمد العامر	١٣٠ج	س ن	٣
٥٧	قرا٣	ب	١		مسجد	ح ث	٤
٥٨	قرا٤	ا	١	عبدالعزيز الخلوي	١٢٧٠	ن ع	٤
٥٩	وحد١	ا	١	عبدالعزيز الخلوي	١٢٧٠	ح ث	٤
٦٠	فقه٦	ا	١	عبدالمنعم محمود	١٢٦٠	ح ث	٤
٦١	نحو٣	ج	١	السعيد عطا	١٢٢٠	ح ث	٤
٦٢	نحو٥	ا	١	محمد محمد ابوالغيط	١١٥٠	سنع	٤
٦٣	طلع١	ا	١	السعيد عطا	١٢٢٠	س ن	٤
٦٤	نقد٢	ا	١	فراج الزهراني	١١٤٠	ح ث	٤
٦٥	ارخ٢	ا	١	صالح عبدالله الغامدي	١١٢٠	س ن	٤
٦٦	جمع٣	ا	١	مسفر ظافر الشهراني	١٦٠ي	ح ث	٤
٦٧	ريض٤	ب	١	بهاء فوزي	٢٥٠س	٤	٤
٦٨	ريض١٢	ا	١		٢٤٠س	سنع	٤
٦٩	كيم٩	ا	١	حسن بالخيور	١٢٠ي	سنع	٤
٧٠	فيز٢	ا	١٠		٢٢٠س	سنع	٤
٧١	فيز٣	ا	١	حسين محمد	٢٣٠س	سنع	٤
٧٢	فيز١١	ا	١	صالح خليفه الحربي	١٤٠ج	سنع	٤
٧٣	احص١	ج	١		٢٦٠س	سنع	٤
٧٤	حسب١	ا	١	خالد اليحي	١٦٠ي	سنع	٤
٧٥	دار١	ا	١	رائد المهيدب	١١٣٠	سنع	٤
٧٦	انجل٧	ا	١		١٢٤٠	سنع	٤
٧٧	فن٢	ج	١	منير عبدالفتاح	١٣٠ي	ح ث	٤
٧٨	بدن١	ب	١	ابراهيم احمد العامر	١٣٠ج	ح ث	٤
٧٩	بدد٢	د	١	ابراهيم احمد العامر	١٣٠ج	س ن	٤
٨٠	قرا٧	ب	١		مسجد	ن ع	٥
٨١	وحد١	ب	١	عبدالعزيز الخلوي	١٢٧٠	ن ع	٥
٨٢	وحد٣	ا	١	عبدالعزيز الخلوي	١٢٧٠	ح ث	٥
٨٣	فقه٢	ا	١	عبدالمنعم محمود	١٢٦٠	ن ع	٥
٨٤	نحو٢	ا	١	محمد محمد ابوالغيط	١١٥٠	ن ع	٥
٨٥	نحو٦	ا	١	السعيد عطا	١٢٢٠	ن ع	٥
٨٦	ادب٣	ا	١	محمد محمد ابوالغيط	١١٥٠	ن ع	٥
٨٧	نشا١	ج	١	فراج الزهراني	١١٤٠	ن ع	٥
٨٨	جمر١	ا	٢	مسفر ظافر الشهراني	١٦٠ي	ن ع	٥
٨٩	ريض٥	ب	١		٢٤٠س	٤	٥
٩٠	ريض١١	ا	١		٢٦٠س	سنع	٥
٩١	كيم٤	ا	١		١٣٠ي	سحث	٥
٩٢	كيم٨	ا	١	حسن بالخيور	١٢٠ي	سحث	٥
٩٣	دار٦	ا	١	رائد المهيدب	١١٣٠	سحث	٥
٩٤	حاسب٢	ا	١	سعود المهيدب	١١٧٠	سحث	٥
٩٥	فن١	د	١	منير عبدالفتاح	١٣٠ي	س ن	٥
٩٦	فن٢	د	١	منير عبدالفتاح	١٣٠ي	ح ث	٥
٩٧	بدن١	ج	١	ابراهيم احمد العامر	١٣٠ج	ح ث	٥
٩٨	بدن٢	ه	١	ابراهيم احمد العامر	١٣٠ج	ن ع	٥

صفحة 1
التاريخ: ٨١/١٨/٠١

تقرير عن طلاب تخصص

رقم التخصص: ٥

الاسم	الرقم
فيصل خشمان العصيمي	٥٩٧
ماهر احمد طرابزوني	٦٦٤
خليل ابراهيم الدغفق	٦٧٦
خالد مفرح الكلي الغامدي	٦٩٢
طارق صالح الكريديس	٧١٢
محمد علي النهري	٧١٨
عبدالعزيز ابراهيم الغانم	٧٢٢
ايمن حمد الخضير	٧٢٥
فهد سعد القحطاني	٧٣١
ماجد احمد المسعد	٧٤٢
علي محمد الناشري	٧٤٥
امجد احمد الناضخ	٧٥٦
ابراهيم حمد الحسيني	٧٥٩
نورالحي عبدالقيوم	٧٦٤
احمد عبدالمحسن الحاجي	٧٧٥
بندر محمد الربيع	٧٧٩
عبدالرحمن خالد السديري	٧٨١
مشعل وارد الجعيد العتيبي	٧٨٩

حسام محمد عسيري	٨٠٩

صفحة 1
التاريخ: ٨١/١٨/٠١

تقرير عن جميع المواد

الاسم	الرمز	الساعات
القرآن و التفسير	قرا١	٢
القرآن و التفسير	قرا٢	٢
القرآن و التفسير	قرا٣	٢
القرآن و التفسير	قرا٤	٢
تفسر وتلاوه	قرا٥	٢
تفسير وتلاوه	قرا٦	٢
الحديث والثقافة الاسلامية	حديث١	٢
الحديث والثقافة الاسلامية	حدث٢	٢
الحديث والثقافة الاسلامية	حدث٣	٢
الحديث والثقافة الاسلامية	حدث٤	٢
الحديث والثقافة الاسلامية	حدث٥	٢
التوحيد	وحد١	٢
التوحيد المتطلب وحد١	وحد٢	٢
التوحيد المتطلب وحد١	وحد٣	٢
التوحيد المتطلب وحد١	وحد٤	٢
الفقه	فقه١	٢
الفقه المتطلب١	فقه٢	٢
الفقه المتطلب فقه١	فقه٣	٢
الفقه المتطلب فقه١	فقه٤	٢
اصول الفقه	فقه٥	٢
اصول الفقه المتطلب٥	فقه٦	٢
السياسة الشرعية	فقه٧	٤
قواعد اللغة العربية	نحو١	٢
قواعد اللغة العربية	نحو٢	٢
قواعد اللغة العربية	نحو٣	٢
قواعد اللغة العربية	نحو٤	٣
قواعد اللغة العربية	نحو٥	٣
قواعد اللغة العربية	نحو٦	٢
الادب العربي	ادب١	٢
الادب العربي	ادب٢	٢
الادب العربي	ادب٣	٢
الادب العربي	ادب٤	٢
الادب العربي	ادب٥	٢
الانشاء	نشا١	٢
الانشاء	نشا٢	٢
الانشاء	نشا٣	٢
المطلعة	طلع١	٢
المطلعة	طلع٢	٢
بلاغة ونقد	نقد١	٢
بلاغة ونقد المتطلب١	نقد٢	٢
بلاغة ونقد المتطلب٢	نقد٣	٢
تاريخ المملكة العربية السعودية	ارخ١	٣
تاريخ الجزيرة العربية القديم	ارخ٢	٣
تاريخ العرب الحديث	ارخ٣	٣
الحضارة الاسلامية	ارخ٤	٣
التاريخ الحديث	ارخ٥	٤
الجغرفيا الطبلة والخرائط	جغرف١	٣
الجغرافيا البشرية المتطلب جغرف١	جغرف٢	٣
الجغرافيا الاقتصاية المتطلب جغرف١	جغرف٣	٣

التاريخ: ٨١/١٨/٠١

تقرير عن جميع الطلاب

الاســـــــم	الرقـــم
ثاني محمد ابو	٧
منصور فالح البلوي	٧٠
محمد عبدالرحمن العامر	١٨٤
سامي عبدالوهاب المقهوي	١٩١
بدر حسن كردي	٢٤٠
منير عبدالعزيز العوده	٢٤٥
يوسف عبدالعزيز الغربي	٢٥٧
صلاح عبدالعزيز المغربي	٢٦٤
سليمان ابراهيم الفعاتق	٢٨٢
ناصر فهد البيشي	٢٨٤
طارق نورالدين حسن رزق	٢٨٥
محمد فريد شريف	٢٨٧
محمد ابراهيم القطنان	٢٩٣
عبدالله سعود البليهي	٢٩٨
زيد حمود الكحلاني	٣٠٧
شقران سعد عيسي الراجح	٣١٠
فهد بلال النعيمي	٣١٦
عبدالوهاب موسي الزهراني	٣١٧
محمد عبدالرحمن الدباغ	٣٢١
عادل فواز العتيبي	٣٢٤
سعد ناصر العقيل	٣٣٢
رباح زوهير غزاوي	٣٣٦
زياد علي الغامدي	٣٤١
علي حسن عبدالله بوحمد	٣٤٧
عبدالكريم عبدالرحمن البابطين	٣٥٣
حسن مرشد الذبياني	٣٥٥
راشد حمد الدوسري	٣٥٧
حسين علي العباس	٣٥٨
محمد علي محمد القحطاني	٣٦٣
سامي حسين اليامي	٣٦٨
سعد عبدالله الشبيعان	٣٧٢
سيف عبدالعزيز السيف	٣٧٣
كمال عباس البقمي	٣٧٦
فهد الحميدي العنزي	٣٨٠
امين محمد الهاشم	٣٨١
بدر عبدالله السنبل	٣٨٢
سامي فرج الحمد	٣٨٧
سعد احمد العبدالهادي	٣٨٨
خالد علي ال شريف	٣٩٨
محمد جاسم الملا	٤٠٢
ابراهيم يوسف الرويعي	٤٠٣
محمد خشيبان الخشيبان	٤٠٦
عبدالعزيز عبدالله العقلا	٤١١
جمال ناصر الغامدي	٤١٣
سفر بن عويضه العويضه	٤٢٠
يحي احمد الفتحي علي	٤٢٢
نايف محمد العلام	٤٢٦
ابراهيم رميح المزيد	٤٢٨
محمد عبدالله الثميري	٤٣٥
عبدالله نهار الرويلي	٤٥٠