1-1-1978

The applicability of graph theory to large scale school and examination scheduling problems.

Francis J. Vasko

THE APPLICABILITY OF GRAPH THEORY TO LARGE SCALE SCHOOL AND EXAMINATION SCHEDULING PROBLEMS

by

Francis J. Vasko

A THESIS

Presented to the Graduate Committee

of Lehigh University

in Candidacy for the Degree of

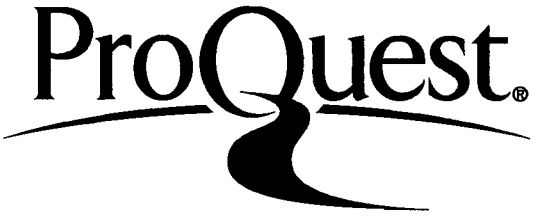
Master of Science

in

The Department of Industrial Engineering

Lehigh University

1978



CERTIFICATE OF APPROVAL

This thesis is accepted and approved in partial fulfillment of the requirements for the degree of Master of Science.

24 April 1978
(Date)

Professor in Charge

Head of the Department

ACKNOWLEDGEMENTS

I would like to express my very sincere thanks to Professor John W. Adams for his valuable advice and counsel in the preparation of this thesis.

TABLE OF CONTENTS

LIST OF TABLES

AN ABSTRACT

of

THE APPLICABILITY OF GRAPH THEORY TO LARGE SCALE SCHOOL AND EXAMINATION SCHEDULING PROBLEMS

by

Francis J. Vasko

In this paper is developed a computer program which uses a graph theoretic approach, developed by Welsh and Powell (10), to construct large scale time-tables. This program is capable of constructing schedules for as many as 960 events. For example, this program will schedule a total of 960 classes or examinations. It is also shown through linear regression analysis, that accurate preliminary approximations can be made for the number of timeslots required to schedule all the events, the maximum number of events scheduled per timeslot, and the execution time of this program on the CDC 6400 computer.

Also, it is shown that this program can be used effectively to schedule classes on the secondary school level and to schedule examinations on the college and university level. Finally, a comparison is made of this method to other methods presently in use and a description is given of how this program may be used if added constraints are imposed.

CHAPTER 1

INTRODUCTION

In today's society there are many scheduling problems which deal with assigning events to certain timeslots with the restriction that certain events may not be assigned to the same timeslot. Two common examples of this type of problem are secondary school class scheduling and university examination scheduling. In the first case, for a given day, there are a number of classes which are to be scheduled into perhaps an eight or nine period day. In other words, so many classes are assigned to meet the first period, the second period, etc. The major restriction being that two or more classes which have a common student or students may not meet in the same timeslot or period. That is, a student cannot take both English and math during the same period. In university examination scheduling the problem is usually a little easier because the exams are scheduled over several days. In other words, the total number of timeslots is not as constrained as for the school scheduling problem.

In this paper I will outline briefly some methods presently used to perform scheduling of this nature. Then I will develop a computer program which will use a graph theoretic algorithm to construct schedules. Finally, I will mention a zero-one integer programming

formulation of this problem.

The purpose of this paper is to explore the usefulness of a graph theoretic algorithm as a means for constructing large scale timetable schedules. The equivalence of examination timetable problems with graph coloring problems was shown by Welsh and Powell (10). However, they stated that the problems remain unsolved when additional restrictions are imposed. The graph theoretic formulation is the following: the events, classes or exams, are represented as vertices of the graph, and a pair of vertices are joined by a line if and only if the corresponding events cannot take place simultaneously. To compute a schedule is equivalent to coloring the graph. That is, color the vertices of the graph in a minimum number of colors, but coloring with different colors any two vertices joined by a line. In this formulation all vertices colored the same color will meet in the same timeslot or period. No conflicts will occur, because vertices joined by lines must be colored different colors, in other words, they must meet in different timeslots or periods.

I will develop a computer program in Fortran IV which will, first of all, for a given number of events, simulate conflicts between the events. Next I will use an approximate coloring algorithm developed by Welsh and

Powell (10) to actually perform the coloring, in other words, to actually perform the scheduling.

The reason I'm using this approximate algorithm instead of an exact algorithm is twofold. First, an exact algorithm would consume much more computer time and, hence, reduce the size of problems which could be dealt with effectively. Secondly, Wood (12) has shown that for problems involving a low probability of conflict (i.e., a probability of conflict less than .25) among the events the method of Welsh and Powell gives very good results. The problems I'm dealing with will all be shown to have a low probability of conflict among the events, hence, justifying this method.

Finally, I will analyze the results to determine the applicability of this technique to large scale scheduling. In particular, its applicability to secondary school scheduling and university examination scheduling. The computer program is capable of scheduling a maximum of about 600 events with the present dimensioned arrays. If the dimensioned arrays are increased to the maximum allowable for the computer I used, then this computer program is capable of scheduling 960 events. The computer I used was Lehigh University's CDC 6400.

There are available commercial data processing type computer programs which assign students to classes

based upon student requests. In this paper I will assume that the classes are already assembled and the problem is to schedule these classes into appropriate timeslots or periods.

The graph theoretic approach that I will be using can also take into account various room and teacher restrictions by incorporating the appropriate lines into the graph. However, my formulation will only include the most important restriction of not scheduling a student for two or more classes or exams during the same time period.

CHAPTER 2

METHODS PRESENTLY USED

In this section I will discuss some commonly used methods for constructing timetables applicable either to secondary school class scheduling or to university examination scheduling.

The methods outlined will be described in terms of secondary school scheduling, however, the same or very similar results hold for university examination scheduling. Differences between the two will be detailed when they arise.

There are three common methods by which these scheduling problems are solved. These methods are: manual scheduling, one-stage, and two-stage computer programming packages.

In manual scheduling the entire problem is done manually by an administrator of the school, usually the principal or assistant principal. This is always a lengthy process and is only possible when at least one of two conditions hold. These conditions are: 1) the school must be small, usually not more than 600 total students, or 2) the curriculum is rigid and offers few or no elective courses. The actual scheduling process usually involves many trial and error attempts at creating a master schedule before a final satisfactory

schedule is completed.

In the case of manual examination scheduling, the same procedures and difficulties are encountered except that the college administrator creating the examination schedule need not worry about scheduling all the exams into an eight or nine period day as the secondary school administrator must. In other words, the total number of timeslots used for examinations extends over several days, therefore, making larger problems susceptible to manual scheduling. However, the work is largely trial and error. Also it is very time consuming, therefore, other methods are being sought.

Commercial computer scheduling packages make wide use of heuristic methods. These programs fall into two general categories. First, there are computer packages which function in two stages.

In the first stage the input consists of student course selections, teacher restrictions, class size restrictions, room restrictions, etc. This input is processed and a typical output would be a list of the courses students actually selected to enroll in. Also as output would be a conflict matrix signifying which courses should not be scheduled concurrently in order to avoid conflicts. As input for the second stage, the administrator in charge now uses the information from the first

stage output to manually decide when to schedule the various course offerings into various periods of the school day. Then this information is used as input for the second phase. The function of the second phase of the computer package is to use the master schedule developed by the administrator to schedule the students into the classes they selected. Typical output from the second stage would be a schedule for each individual student scheduled successfully. Also a list of students who could not be scheduled and the reason why these students could not be scheduled, and the percentage of students successfully scheduled.

Now the administrator must decide how to alleviate the conflicts and thus, increase the number of students scheduled. There are at least two general methods that can be used to reduce the number of conflicts. First, a manual check of which classes seem to be causing numerous conflicts may be rescheduled manually by the administrator. Secondly, the students who have not been scheduled successfully may be asked to select alternate courses.

What seems to be done in practice is that the second phase of the package is run several times. After each run the administrator in charge makes manual adjustments of the first kind outlined above, and then runs the entire second phase again. This revision procedure

continues until either no further changes can be made or until an acceptable level of students have been scheduled successfully. Then the students who still have not been scheduled successfully are dealt with on an individual basis. Usually the students who have not been scheduled successfully must make changes in their course selections in order to obtain a feasible schedule.

The second general type of computer package is a one-stage package. This package uses as input student course selections, teacher restrictions, class size restrictions, room restrictions, etc. The difference between the one-stage and two-stage packages is that the one-stage package will do all the scheduling itself. In other words, the output consists of a master schedule which has assigned to each period of the school day a number of classes. In the two-stage approach this was done manually by an administrator. Also as output the administrator receives a schedule for each individual student scheduled successfully, a list of students who could not be scheduled and the reason why these students could not be scheduled, and the percentage of students successfully scheduled. Now the administrator must make manual revisions in the master schedule based upon a study of where the majority of conflicts are. These revisions, as in the two-stage package, involve either re-

scheduling classes or having students select alternate courses. Usually classes are rescheduled manually and the entire program is rerun. This is done several times until either no further changes can be made or until an acceptable level of students are scheduled successfully. Then the students who have not been scheduled successfully are required to make changes in their course selections.

When using computer packages to schedule university examinations, the packages and procedures are similar to secondary school scheduling. Except when students are not scheduled successfully, in other words, when a student has two or more exams scheduled in the same timeslot, then special provisions are made for that student to take the exam usually during a later timeslot.

The major disadvantage of the methods outlined above is the amount of time required. Scheduling done totally without the aid of a computer yields very limited results in terms of school size or curriculum flexibility which can be dealt with in this manner. Even the computer scheduling packages require a certain amount of manual assistance, as well as a good degree of compromise. Also considerable computer time and resources are required by these packages. All in all, the large timetable scheduling problem, as viewed as either secondary school schedul-

ing or as university examination scheduling, appears to be a very difficult problem to deal with and to solve.

CHAPTER 3

DISCUSSION OF PROGRAM COLOR

Program COLOR, written in Fortran IV, is on file at Lehigh University's computer center.

Program COLOR uses a graph theoretic coloring algorithm to schedule events. This program is capable of scheduling large scale timetable problems. The output is in terms of the secondary school scheduling problem, but can easily be interpreted for other timetable problems by merely substituting appropriate terms for the terms class and period. For example, in dealing with university examination scheduling one would use the term examination instead of class and use the term timeslot instead of period. Except for output terminology, which is specific to secondary school scheduling, the program is very general and capable of handling any large-scale timetable problem.

This program is capable of scheduling 600 events, that is, 600 classes or examinations into periods or timeslots. To schedule more than 600 classes or examinations would require changing the array dimensions. The maximum allowable number is approximately 960 events. The restriction is due to the core capacity of the CDC 6400.

The reason this program is capable of handling such large scale problems is that the conflict array elements are stored in bit positions instead of using an entire word to store each element of the conflict array. This is possible because the elements of the conflict array are all either zeros or ones. The use of bit positions in place of words is accomplished through the use of three subroutines. These subroutines are ITRANS, PUTBIT, and GETBIT. In subroutine ITRANS a standard two dimensional array position is translated into the appropriate bit position. In subroutine PUTBIT a bit value is assigned and in subroutine GETBIT a desired bit value is retrieved. Since the length of a computer word in the CDC 6400 is 60 bits, by using the above method I can store a conflict array 60 times larger than would be normally possible.

The minimum hardware configuration is the CDC 6400 central processor, card reader, and printer. The maximum core memory requirement is 54,000 octals to schedule up to 600 events. In order to schedule 960 events 120,000 words of core memory, given in octals, are required. There are no error indications for either the operator or the users.

In the next section dealing with the analysis of the results I will show that there exists a strong linear

tendency between the number of events to be scheduled and the amount of system seconds required to execute the program. The relationship is approximately given by:

System Seconds = .0846 (number of events) - 6.4973

provided that the number of events to be scheduled is at least 60.

The following is a description of the input file. The input data required for execution of this program is read from one card. The format is 2I3, F10.2. The first field is the total number of events to be scheduled. The second field is the number of repetitions of the schedule that are to be computed. The third field is the probability of conflict given as a decimal.

The output file is described below. For each repetition the number of the repetition, the number of total classes to be scheduled and the conflict probability are printed. Next, all the classes meeting in the first period or timeslot are listed, and the total number of classes meeting in period one is printed. This is done for each succeeding period until no more periods are needed.

The operation and results of this program have been checked manually for some of the results. In all cases, the results were correct and the program appears to be operating correctly.

This program consists of the following eight major parts:

1) read the data and select a random number seed
2) generate and insert entries into the conflict array in a symmetric manner
3) set the values of the diagonal of the conflict array equal to zero
4) calculate the degree of each vertex
5) rearrange the conflict array rows based upon descending order of the degrees of the vertices corresponding to the rows
6) initialize the arrays used in actually computing the coloring of the network as given in its equivalent form in the conflict array
7) compute the coloring, in other words, construct the schedule
8) print the schedule.

The first thing the program does is read the data from one data card. Next it initializes the random number seed. This is done because I will use the CDC 6400's random number generator to generate entries for the conflict array. Although the values of the conflict array are stored in single bit positions the array can be thought of as a two dimensional array with both dimensions being the total number of events to be scheduled. For

example, if there are 500 events to be scheduled, then the conflict array can be thought of as a 500 by 500 dimensioned array. The (i,j) entry in this array is one if events i and j conflict and 0 if they do not conflict. The values for the entries are generated using the CDC 6400's random number generator based on a given probability of conflict. As stated earlier, the probability of conflict is read from the data card. For example, if the probability of conflict were .10, then the probability that any given class conflicted with any other class would be .10. This would be reflected in the values assigned in the conflict array. Thus, the reason for several repetitions would be to study the results of several randomly generated schedules. For a user to utilize this program to construct a schedule empirical data would have to be input in order to construct the conflict array. The random number generator is only employed when simulated schedules are constructed.

In the next part of the program entries are generated and inserted into the conflict array in a symmetric manner. The reason for the symmetry is that class i conflicts with class j if and only if class j conflicts with class i. This is accomplished using the CDC 6400's random number generator, and the subroutines ITRANS and PUTBIT.

Next, the values of the diagonal of the conflict array are set equal to zero. This is done because no class can conflict with itself. This is accomplished using the subroutines ITRANS and PUTBIT.

The next step is to calculate the degree of each vertex. That is, to determine how many lines join each vertex. This is done for vertex i by merely counting the number of ones that appear in the i^{th} row of the conflict array.

After the degree of each vertex is determined, then the program proceeds to rearrange the rows of the conflict array based upon descending order of the degrees of the vertices corresponding to the rows. This is accomplished through the use of a sorting routine.

Now the arrays used in actually computing the coloring of the network as given in its equivalent form in the conflict array are initialized.

Once the work arrays are initialized the program computes the coloring. The algorithm used was first suggested by Welsh and Powell (10). The method is as follows: the vertices are initially arranged in descending order of their degrees. Color the first vertex with color one and scan the list of vertices downwards coloring with one any vertex which does not conflict with another vertex that has already been colored with one. Starting from

the top of the list, color the first uncolored vertex by color two and again scan the list downward coloring with two any uncolored vertex which is not in conflict with another vertex that has already been colored with two. Proceed in the same way with colors three, four, etc. until all the vertices have been colored.

The final section prints the color partition; in other words, it prints the schedule.

CHAPTER 4

ANALYSIS OF RESULTS

The purpose of this section is to analyze the results obtained from executions of program COLOR. Details of the function and purpose of this computer program are given in the previous chapter. For all tables in this chapter, I will use the general term of event instead of using the terms class or examination, and I will use the more general term timeslot instead of period. These terms will be used unless I am referring to a particular application in which I will use the appropriate terminology. The two specific application areas I will analyze are secondary school scheduling and university examination scheduling.

The computer program COLOR generates a graph and then colors it. It does not actually construct a graph, but instead constructs its equivalent conflict array. This array has the same number of rows as columns, and that number is the total number of events to be scheduled. For example, if 300 events are to be scheduled, then the conflict array would be a two dimensional array with 300 rows and 300 columns. The (i,j) position of the array, that is, the i^{th} row and j^{th} column has value one if and only if events i and j conflict. If events i

and j do not conflict, then the (i,j) position of the array has the value zero. Thus the conflict array consists of only zeros and ones. The actual compact storage of this array is explained in the previous chapter. The manner in which the entires for the conflict array are generated is through the use of a random number generator. For example, if the probability of conflict is chosen to be .10, then the random number generator will complete the conflict array in such a manner that, on the average, for any event i the probability of another event j, distinct from i, of conflicting with i is .10 or 10%. For a user to utilize this program to construct a schedule empirical data would have to be input in order to construct the conflict array. The random number generator is only employed when simulated schedules are constructed. I will now explain how I arrived at the various levels of probability of conflict for the two application areas.

The various values I used for total number of events and the values for probability of conflict are based on the following analysis. The assumptions made in this analysis are based on my personal experience as a secondary school teacher and on my experience as a graduate student and teaching assistant.

I will first consider secondary school scheduling.

From my own experiences I have been able to determine values for the total number of classes that are to be scheduled on a given day. I have made the following assumptions:

1) the average class size is thirty students
2) each student is scheduled for seven classes; this includes five major subjects, two minor subjects, and lunch is scheduled manually.
3) study periods are not scheduled, but assigned later since the students are not grouped in any special manner for study periods.
4) the school day is 9 periods long.

From the above assumptions I deduced that the following number of classes would have to be scheduled per day for the given school populations:

300 students - 60 classes

500 to 800 students - 120 to 180 classes

1000 to 1300 students - 240 to 300 classes

1500 students - 360 classes

Observe that to use this computer program for school scheduling it would be executed five times, that is, it would determine separately a schedule for each day of the week.

I will now explain how I determined the various levels of probability of conflict as applied to secondary

school scheduling. I considered four situations and based my analysis on the ones with the highest probability of conflict. The four situations are:

1) one group of students per class - no electives
2) two groups of students per class - few electives
3) three groups of students per class - moderate electives
4) four groups of students per class - many electives

What one group of students per class means is that, on the average, students are grouped according to ability level and a given number of students constitute a class which meets with all the same students for each of the seven scheduled classes per day. In this situation there are no elective courses. The situation of two groups of students per class arises when the students are allowed to take electives. This situation means that, on the average, in a given class there are basically two different groups of students, that is, aside from the one class in common the two groups of students are taking totally different courses. The same situation occurs for three groups of students and four groups of students, the only difference being that the number of groups increases as the number of different elective courses selected increases. The probability of conflict of a given class with any other class can easily be computed for each of

the four situations. The results of these computations are given in Table 37. As can be seen from the results in Table 37 in each case, except for 60 classes, the maximum probability of conflict is approximately .10. For the case of 60 classes the maximum probability of conflict is .20.

In the university exam scheduling problem the probability of conflict intuitively should be higher than in the secondary school situation. This is because the students that meet for any given class take a wide variety of other courses. This is particularly true in large introductory classes where the students taking the class constitute a wide range of interests. Therefore, the probability of conflict between two classes will most likely be higher than those outlined for secondary school classes. However, the diversity would have to be rather extensive to exceed a probability of .20. For example, if the total number of exams to be scheduled are 360 and if on the average in a given exam there are 14 distinct groups of students each taking 6 exams in all, then the probability of conflict would be approximately .20. Also from inspecting exam schedules it appears that for a college or university with 4000 to 5000 students the number of exams scheduled ranges from 300 to 400 exams.

I executed program COLOR for 60 events, 120 events, 180 events, 240 events, 300 events, and 360 events. The probabilities of conflict that I used are .10, .15, and .20. These values were used because of the information gained through the analyses given above for typical school scheduling situations and for typical university examination scheduling situations. Also I did not perform runs with a larger number of events, because I felt that the cost, for all the repetitions for each probability of conflict, to compute a schedule for a large number of events would not justify the limited additional information gained. The number of repetitions performed for each combination of probability of conflict and total number of events is in Table 38. For 60 events, 300 events, and 360 events I performed only five repetitions for each probability of conflict, whereas, for 120 events and 240 events I performed thirty repetitions for each probability of conflict. Also, for each probability of conflict associated with 180 events, I performed fifteen repetitions. Again, due to cost considerations I performed thirty repetitions for only 120 and 240 events. However, in Tables 34, 35, and 36 I compare for a given probability of conflict level, the standard deviation of the number of events scheduled per timeslot for the various total number of events. From studying the statistics

given in these tables, there appears to be no significant differences. This appears to indicate that the results gained from performing only five repetitions are probability as valid as the results gained from performing thirty repetitions. Thus, only a small number of repetitions is needed to obtain valid conclusions from this program.

In Tables 19 and 20 results concerning the number of system seconds needed to execute program COLOR are summarized. From Table 19 it can be seen that, for a given number of events, the probability of conflict has no significant affect on the execution time of the program. This is intuitively plausible because as the probability of conflict increases less events will be scheduled per timeslot, but simply more timeslots will be needed. In any case the same number of total events will be scheduled. Using the average system seconds required to execute program COLOR, I performed a linear regression analysis comparing the number of events to the number of system seconds. The correlation coefficient I obtained for this analysis was .976, in other words, the data fits a linear relationship quite well. The equation for this relationship is:

(1) system seconds = .0846 (number of events) - 6.4973, provided the number of events is at least 60. Equation

(1) enables one to approximately determine the number of system seconds required to execute program COLOR given only the desired number of events to be scheduled. For example, if I desired to determine the number of system seconds required to execute program COLOR for say 600 events I would merely use equation (1) and compute the answer to be approximately 75 system seconds. The amount of computer time needed to execute program COLOR for even very large scheduling problems is extremely reasonable and economical on the CDC 6400.

This method of scheduling does not attempt to balance in any manner the number of events scheduled per timeslot. In Tables 1 to 18, as well as in Tables 25 to 33, one can compare the number of events scheduled per timeslot. In all cases, the number of events scheduled per timeslot decreases gradually as the number of the timeslot increases, except in the last few timeslots when the number of events scheduled per timeslot decreases rapidly. In the case of secondary school scheduling, two techniques can be employed to remedy the sharp decrease in number of events scheduled during the last few timeslots. First, a few events may be rescheduled manually to increase the balance in classes shceduled per period. Secondly, study periods which are not formally scheduled using this program may be manually scheduled during the

periods when few classes are scheduled. The scheduling of study periods may also be done using a simple computer program to fill in vacant classrooms with students who are not scheduled for any classes. In examination scheduling the problem of balance is not that important because there are usually enough physical facilities to accommodate all the exams scheduled for any given timeslot. However, if problems arise minor manual rescheduling may be necessitated. In general the standard deviation of the number of events scheduled per timeslot for all cases was relatively small. Also checking the maximum number of events scheduled per timeslot, usually occurring in the first timeslot, these values were all well within the known physical classroom capacities for either secondary schools or universities. In other words, there appears no need to be concerned that this program will schedule too many events into a given timeslot. To further substantiate this claim, I performed two linear regression analyses. The first linear regression analysis compared the probability of conflict to the number of classes scheduled in the first timeslot. I performed this analysis for each of the various total number of events to be scheduled. The results are:

1) for 60 events, the correlation coefficient equals -.971 and number of events scheduled in first

timeslot = -.7 (prob. of conflict) +25.5

2) for 120 events, the correlation coefficient equals -.993 and number of events scheduled in first timeslot = -1. (prob. of conflict) +34.667

3) for 180 events, the correlation coefficient equals -.995 and number of events scheduled in first timeslot = 1. (prob. of conflict) +39.667

4) for 240 events, the correlation coefficient equals -.991 and number of events scheduled in first timeslot = -1.3 (prob. of conflict) +43.5

5) for 300 events, the correlation coefficient equals -.976 and number of events scheduled in first timeslot = -1.3 (prob. of conflict) +44.167

6) for 360 events, the correlation coefficient equals -.994 and number of events scheduled in first timeslot = -1.5 (prob. of conflict) +49.5.

The second linear regression analysis compared the total number of events to the number of events scheduled in the first timeslot. I performed this analysis for three probabilities of conflict, namely, .10, .15, and .20. The results are:

1) for probability of conflict .10, the correlation coefficient equals .973 and number of events scheduled in first timeslot = .0495 (number of events) +17.933

2) for probability of conflict .15, the correlation coefficient equals .954 and

number of events scheduled in first timeslot

= .0352 (number of events) +13.6

3) for probability of conflict .20, the correlation coefficient equals .981 and

number of events scheduled in first timeslot

= .0257 (number of events) +11.267.

Judging from the correlation coefficients it appears that all the above equations should yield good approximations to the maximum number of events scheduled in any given timeslot. For example, suppose I wanted to determine the maximum number of events scheduled per timeslot for 500 events and probability of conflict .15, then using the appropriate equation above the answer is computed to be approximately 31 events.

The results concerning total timeslots required to schedule with no conflicts are given in Tables 21, 22, and 24. The applicability of this program, based upon these results, to either examination scheduling or to secondary school scheduling will be discussed in the next paragraphs. From Table 21 three facts are immediately evident. First of all, for a given number of events, the number of timeslots required increases as the probability of conflict increases. Secondly, the standard deviation

for all the results is very small. Thirdly, for a given probability of conflict the number of timeslots required increases as the number of events increases. This last result might not be intuitively appealing to some observers. One might reason that if the algorithm is working properly, then for a given probability of conflict the required number of timeslots should be about the same, regardless of the number of events to be scheduled. However, the subtle fact is that as the number of events increases any event has a larger number of possible events to conflict with even though the probability of conflict is the same. This results in a more intricate interaction of events in terms of conflicts, hence, leading to the larger number of required timeslots. I performed a linear regression analysis comparing the number of events to the number of timeslots required for scheduling the events. I performed this analysis for three probabilities of conflict, namely, .10, .15, and .20. In all cases the correlation coefficients were extremely close to one, indicating a strong linear relationship. The results are:

1) for probability of conflict .10 the correlation coefficient is .999 and number of timeslots = .0326 (number of events) + 4.02
2) for probability of conflict .15 the correlation co-

efficient is .998 and number of timeslots = .0442 (number of events) + 4.76

3) for probability of conflict .20 the correlation coefficient is .997 and number of timeslots = .0564 (number of events) + 5.2,

provided the number of events lie between 60 and 960. Thus, for a given probability of conflict the required number of timeslots can be determined for any number of events lying between 60 and 960. For example, assume we are interested in determining the required number of timeslots to schedule 800 events with a probability of conflict .10, then the answer is computed by the use of the appropriate equation to be 30 timeslots.

The average number of days that exams are scheduled in colleges or universities ranges from six to nine days with from three to five exam periods scheduled per day. That is, the number of timeslots used to give examinations ranges from 18 to 45. The above information is based on a survey of 20 colleges and universities of varying student population sizes. Also I talked with the registrars' offices of four local colleges or universities with student populations ranging from 1200 to 6000 students. The number of examinations given varied with the various curriculums, but no college or university surveyed scheduled more than 425 examinations. Although

I have no conclusive evidence, the fact that program COLOR is capable of scheduling a maximum of 960 events within the number of timeslots mentioned above, leads me to believe that this program could effectively schedule examinations for the vast majority of colleges and universities. Certainly it should be able to schedule examinations for any college or university with a student population of 15000 or less.

For the various actual school scheduling situations and student populations outlined at the beginning of this chapter, Table 37 gives the estimated probabilities of conflict. Table 23 gives the percentage of classes which must be rescheduled based on a nine period school day when the probabilities of conflict are .10, .15, and .20. From Table 37 one can see that a realistic probability of conflict value for 60, 120, and 180 classes is .20, .10, and .10 respectively. Now from Table 23 one can observe that for 60, 120, and 180 classes with a probability of conflict .20, .10, and .10 respectively, the percentage of classes which must be rescheduled are 0%, 0%, 1.11% respectively. Hence, this program appears to be functioning adequately for the schools with student populations represented by 60, 120, and 180 classes. I performed a linear regression analysis for schedules with 240, 300, and 360 events comparing the probability of

conflict to the percentage of classes which must be rescheduled. The correlation coefficients for 240, 300, and 360 classes were .999, .991, and .995 respectively. In other words, the data fits linear curves very well. The actual equations are:

1) for 240 events

   percentage rescheduled = 304.2 (prob. of conflict) -22.43

2) for 300 events

   percentage rescheduled - 280 (prob. of conflict) -10.22

3) for 360 events

   percentage rescheduled = 297.2 (prob. of conflict) -5.78.

I used the information gained from the above linear regression analysis and the information in Table 37 to calculate realistic values for the percentage of classes that must be rescheduled for 240, 300, and 360 total classes. The percentage of classes that must be rescheduled when scheduling 240, 300, and 360 classes are 2%, 12%, 14% respectively. Now in the case where there are 240, 300, or 360 classes we see that some rescheduling is necessary, but the amount of rescheduling that must be done is small and should be easily handled manually. Also, the value for the probability of conflict for each

school size was taken to be the maximum value calculated based on a curriculum with a large number of electives, therefore, for schools with fewer elective course offerings the percentage of classes that would need to be rescheduled would be very small. Hence, it appears that this program will efficiently schedule classes for a wide variety of school sizes and curriculums.

TABLE 1

THE NUMBER OF EVENTS SCHEDULED PER TIMESLOT
FOR 60 EVENTS AND PROBABILITY OF CONFLICT .10

Timeslot	Average # of Events	Standard Deviation	Minimum	Maximum
1	19	2.098	15	21
2	17	2.145	14	19
3	12	1.483	9	13
4	8	1.844	5	10
5	4	1.049	3	6
6	1	.447	0	1

TABLE 2

THE NUMBER OF EVENTS SCHEDULED PER TIMESLOT
FOR 60 EVENTS AND PROBABILITY OF CONFLICT .15

Timeslot	Average # of Events	Standard Deviation	Minimum	Maximum
1	14	1.095	13	16
2	14	.775	13	15
3	12	.632	11	13
4	9	1.095	8	11
5	6	.447	6	7
6	4	1.0	2	5
7	1	.447	1	2

TABLE 3

THE NUMBER OF EVENTS SCHEDULED PER TIMESLOT
FOR 60 EVENTS AND PROBABILITY OF CONFLICT .20

Timeslot	Average # of Events	Standard Deviation	Minimum	Maximum
1	12	.775	11	13
2	11	.775	10	12
3	10	.775	9	11
4	9	1.844	7	11
5	8	1.342	7	10
6	5	1.342	3	7
7	2	1.0	1	4
8	1	.775	0	2
9	1	.775	0	2

TABLE 4

THE NUMBER OF EVENTS SCHEDULED PER TIMESLOT
FOR 120 EVENTS AND PROBABILITY OF CONFLICT .10

Timeslot	Average # of Events	Standard Deviation	Minimum	Maximum
1	25	2.145	21	30
2	23	1.732	20	26
3	20	2.145	17	25
4	18	1.517	15	22
5	15	2.121	9	18
6	10	1.517	7	13
7	6	1.703	3	9
8	2	1.095	0	4
9	0	.477	0	1

TABLE 5

THE NUMBER OF EVENTS SCHEDULED PER TIMESLOT
FOR 120 EVENTS AND PROBABILITY OF CONFLICT .15

Timeslot	Average # of Events	Standard Deviation	Minimum	Maximum
1	19	1.817	17	24
2	18	1.703	14	21
3	17	1.304	15	20
4	16	1.342	13	18
5	14	1.449	11	17
6	13	1.844	9	17
7	10	1.183	8	14
8	7	1.612	4	10
9	4	1.449	1	7
10	2	1.095	0	3
11	0	.632	0	2
12	0	.173	0	1

TABLE 6

THE NUMBER OF EVENTS SCHEDULED PER TIMESLOT
FOR 120 EVENTS AND PROBABILITY OF CONFLICT .20

Timeslot	Average # of Events	Standard Deviation	Minimum	Maximum
1	15	1.304	13	18
2	15	1.643	11	18
3	14	1.949	11	18
4	13	1.378	10	16
5	12	1.449	9	15
6	11	1.549	8	15
7	10	1.612	7	14
8	9	1.265	7	12
9	8	1.095	5	10
10	6	1.049	3	7
11	4	1.265	2	6
12	1	.837	0	3
13	0	.447	0	2

TABLE 7

THE NUMBER OF EVENTS SCHEDULED PER TIMESLOT
FOR 180 EVENTS AND PROBABILITY OF CONFLICT .10

Timeslot	Average # of Events	Standard Deviation	Minimum	Maximum
1	28	2.966	25	37
2	27	1.732	24	30
3	24	2.408	20	28
4	25	2.191	22	27
5	21	1.897	18	23
6	19	1.549	16	21
7	15	1.673	12	18
8	12	1.924	9	15
9	6	1.844	3	10
10	2	1.304	0	4
11	0	.245	0	1

TABLE 8

THE NUMBER OF EVENTS SCHEDULED PER TIMESLOT
FOR 180 EVENTS AND PROBABILITY OF CONFLICT .15

Timeslot	Average # of Events	Standard Deviation	Minimum	Maximum
1	21	1.643	18	24
2	21	1.703	19	25
3	19	1.483	17	22
4	19	1.703	16	22
5	18	1.732	15	21
6	17	1.581	15	21
7	15	.775	14	16
8	14	1.897	9	17
9	12	1.265	8	13
10	10	1.483	7	13
11	7	1.265	6	10
12	4	1.581	1	8
13	2	1.304	0	5
14	0	.4	0	1

TABLE 9

THE NUMBER OF EVENTS SCHEDULED PER TIMESLOT
FOR 180 EVENTS AND PROBABILITY OF CONFLICT .20

Timeslot	Average # of Events	Standard Deviation	Minimum	Maximum
1	16	2.258	15	19
2	17	1.095	15	19
3	16	1.049	14	17
4	15	1.449	14	19
5	16	1.732	13	19
6	14	1.449	12	17
7	14	1.225	11	16
8	13	1.612	9	15
9	11	1.414	8	13
10	11	1.304	9	13
11	10	1.897	8	16
12	8	1.414	6	12
13	7	1.517	5	10
14	5	1.225	3	7
15	3	1.304	1	6
16	1	.949	0	3
17	0	.4	0	1

TABLE 10

THE NUMBER OF EVENTS SCHEDULED PER TIMESLOT
FOR 240 EVENTS AND PROBABILITY OF CONFLICT .10

Timeslot	Average # of Events	Standard Deviation	Minimum	Maximum
1	31	2.280	27	36
2	30	2.280	26	35
3	29	1.949	25	34
4	27	1.870	24	32
5	25	2.191	21	29
6	24	2.470	19	28
7	21	2.121	16	27
8	18	1.870	14	23
9	15	2.214	10	22
10	10	1.761	7	14
11	6	1.871	1	10
12	2	1.483	0	5
13	0	1.118	0	2

TABLE 11

THE NUMBER OF EVENTS SCHEDULED PER TIMESLOT
FOR 240 EVENTS AND PROBABILITY OF CONFLICT .15

Timeslot	Average # of Events	Standard Deviation	Minimum	Maximum
1	23	1.673	20	27
2	22	1.612	19	25
3	21	1.897	17	25
4	21	1.517	19	25
5	20	1.673	17	24
6	20	1.581	17	23
7	19	1.673	16	22
8	17	1.549	15	20
9	16	1.817	12	20
10	15	1.265	11	17
11	13	1.581	11	16
12	11	1.673	8	15
13	9	1.732	6	12
14	6	1.612	3	10
15	3	1.549	1	6
16	1	2.569	0	4
17	0	.173	0	1

TABLE 12

THE NUMBER OF EVENTS SCHEDULED PER TIMESLOT
FOR 240 EVENTS AND PROBABILITY OF CONFLICT .20

Timeslot	Average # of Events	Standard Deviation	Minimum	Maximum
1	18	1.414	16	21
2	18	1.549	16	21
3	17	1.549	15	20
4	17	1.483	14	20
5	17	1.673	13	21
6	17	1.581	14	20
7	16	1.703	13	19
8	15	1.643	12	19
9	14	1.483	12	17
10	14	1.414	12	18
11	13	1.449	11	16
12	12	1.517	10	15
13	12	1.643	9	15
14	11	1.183	9	13
15	10	1.265	7	13
16	8	1.732	5	12
17	6	1.732	2	9
18	4	1.612	0	6
19	1	1.049	0	4
20	0	.632	0	2

TABLE 13

THE NUMBER OF EVENTS SCHEDULED PER TIMESLOT
FOR 300 EVENTS AND PROBABILITY OF CONFLICT .10

Timeslot	Average # of Events	Standard Deviation	Minimum	Maximum
1	32	2.098	30	35
2	33	2.280	30	36
3	31	2.490	27	34
4	30	1.183	29	32
5	28	2.145	26	31
6	27	1.183	26	29
7	25	2.933	22	29
8	24	2.490	21	28
9	21	2.324	18	23
10	18	2.236	15	20
11	15	.775	14	16
12	9	1.612	6	10
13	6	1.483	4	8
14	2	1.183	0	3

TABLE 14

THE NUMBER OF EVENTS SCHEDULED PER TIMESLOT
FOR 300 EVENTS AND PROBABILITY OF CONFLICT .15

Timeslot	Average # of Events	Standard Deviation	Minimum	Maximum
1	23	.775	22	24
2	25	1.0	23	26
3	23	1.095	21	24
4	24	.632	23	25
5	22	.447	22	23
6	21	1.844	19	23
7	22	.447	21	22
8	20	1.265	19	22
9	18	.775	17	19
10	18	1.897	16	21
11	17	2.236	15	21
12	17	1.342	14	18
13	14	1.342	12	16
14	12	.775	11	13
15	11	1.183	9	12
16	7	2.324	4	11
17	4	1.483	2	6
18	2	1.342	0	4
19	0	.447	0	1

TABLE 15

THE NUMBER OF EVENTS SCHEDULED PER TIMESLOT
FOR 300 EVENTS AND PROBABILITY OF CONFLICT .20

Timeslot	Average # of Events	Standard Deviation	Minimum	Maximum
1	19	1.096	18	21
2	20	.633	19	21
3	19	.633	18	20
4	19	1.732	17	22
5	18	.447	18	19
6	17	1.0	15	18
7	18	.775	17	19
8	17	1.844	14	19
9	17	1.844	15	20
10	15	1.549	12	16
11	16	1.732	14	19
12	15	.775	14	16
13	14	.775	13	15
14	14	1.673	11	16
15	12	1.265	11	14
16	13	1.183	12	15
17	9	.775	8	10
18	9	.775	8	10
19	7	1.342	6	10
20	6	1.096	4	7
21	3	.775	2	4
22	1	.633	0	2
23	0	.447	0	1

TABLE 16

THE NUMBER OF EVENTS SCHEDULED PER TIMESLOT
FOR 360 EVENTS AND PROBABILITY OF CONFLICT .10

Timeslot	Average # of Events	Standard Deviation	Minimum	Maximum
1	35	.894	34	36
2	35	1.483	33	37
3	33	3.0	30	37
4	33	3.0	29	37
5	32	1.844	30	35
6	30	3.066	27	35
7	28	2.236	26	31
8	25	.775	24	26
9	25	1.732	22	27
10	22	1.414	20	24
11	21	.447	20	21
12	16	1.844	14	19
13	11	2.569	8	15
14	8	1.613	6	10
15	4	1.897	1	6
16	1	1.0	0	2
17	0	.775	0	1

TABLE 17

THE NUMBER OF EVENTS SCHEDULED PER TIMESLOT
FOR 360 EVENTS AND PROBABILITY OF CONFLICT .15

Timeslot	Average # of Events	Standard Deviation	Minimum	Maximum
1	26	2.236	24	30
2	25	1.613	23	27
3	26	1.949	22	27
4	25	1.414	23	27
5	24	1.183	23	26
6	22	2.145	19	25
7	23	1.265	21	25
8	23	.894	22	24
9	21	1.414	19	23
10	19	.775	18	20
11	20	1.096	18	21
12	18	1.0	16	19
13	17	1.0	16	18
14	14	1.483	13	17
15	14	1.342	12	16
16	13	1.342	12	16
17	11	2.145	7	13
18	10	2.049	6	12
19	6	1.844	3	8
20	3	1.414	1	5
21	1	.775	0	2

TABLE 18

THE NUMBER OF EVENTS SCHEDULED PER TIMESLOT
FOR 360 EVENTS AND PROBABILITY OF CONFLICT .20

Timeslot	Average # of Events	Standard Deviation	Minimum	Maximum
1	20	1.183	19	22
2	20	.447	20	21
3	19	.775	18	20
4	18	.775	17	19
5	18	1.183	17	20
6	19	.775	19	21
7	19	.775	18	20
8	18	1.549	17	21
9	17	1.183	15	18
10	17	1.096	16	19
11	18	.775	17	19
12	17	1.096	16	19
13	16	.633	15	17
14	17	.775	16	18
15	15	1.183	13	16
16	14	1.0	12	15
17	13	.775	13	15
18	13	1.183	12	15
19	11	1.483	10	13
20	10	.775	9	11
21	9	2.049	6	11
22	9	1.183	7	10
23	6	1.483	5	9
24	4	.447	3	4
25	1	.775	0	2
26	0	.447	0	1

TABLE 19

AVERAGE SYSTEM SECONDS NEEDED TO EXECUTE PROGRAM COLOR

Number of Events	Prob. .10	Prob. .15	Prob. .20	Average System Seconds	Standard Deviation	Minimum	Maximum
60	1.26	1.24	1.24	1.25	.010	1.24	1.26
120	2.91	2.95	2.96	2.94	.022	2.91	2.96
180	6.83	6.87	6.95	6.88	.050	6.83	6.95
240	11.50	11.68	11.56	11.58	.075	11.50	11.68
300	18.30	18.56	18.56	18.47	.125	18.30	18.56
360	26.38	26.56	26.68	26.54	.123	26.38	26.68

TABLE 20

PERCENTAGE INCREASE OF SYSTEM SECONDS TO EXECUTE PROGRAM COLOR

Probability of Conflict	% Increase 60 to 120	% Increase 120 to 180	% Increase 180 to 240	% Increase 240 to 300	% Increase 300 to 360
.10	131	135	68	59	44
.15	138	133	70	59	43
.20	139	135	66	61	44

TABLE 21

TOTAL TIMESLOTS REQUIRED TO SCHEDULE WITH NO CONFLICTS

Number of Events	Probability of Conflict	Average Number of Timeslots Needed	Standard Deviation	Minimum	Maximum
60	.10	5.8	.400	5	6
60	.15	7.0	.000	7	7
60	.20	8.0	.894	7	9
120	.10	8.0	.608	7	9
120	.15	10.23	.671	9	12
120	.20	12.07	.574	11	13
180	.10	9.93	.447	9	11
180	.15	13.07	.574	12	14
180	.20	16.0	.520	15	17
240	.10	12.13	.500	11	13
240	.15	15.53	.566	15	17
240	.20	19.03	.755	17	20
300	.10	13.8	.400	13	14
300	.15	18.0	.632	17	19
300	.20	22.2	.748	21	23
360	.10	15.6	.800	15	17
360	.15	20.4	.490	20	21
360	.20	25.0	.632	24	26

TABLE 22

COMPARISON OF AVERAGE TOTAL TIMESLOTS REQUIRED
AS THE PROBABILITY OF CONFLICT VARIES

Number of Events	Prob. .10	Prob. .15	Prob. .20	% Increase from .10 to .15	% Increase from .15 to .20
60	5.8	7.0	8.0	21	14
120	8.0	10.23	12.07	28	18
180	9.93	13.07	16.0	32	22
240	12.13	15.57	19.03	28	23
300	13.8	18.0	22.2	30	23
360	15.6	20.4	25.0	31	23

TABLE 23

PERCENTAGE OF CLASSES WHICH MUST BE RESCHEDULED
MANUALLY BASED ON A NINE PERIOD DAY

Probability of Conflict	60 Classes	120 Classes	180 Classes	240 Classes	300 Classes	360 Classes
.10	0.	0.	1.11	7.50	16.67	23.06
.15	0.	1.67	12.78	24.78	34.00	40.56
.20	0.	9.17	25.00	37.92	44.67	52.78

TABLE 24

COMPARISON OF TOTAL TIMESLOTS REQUIRED AS TOTAL EVENTS INCREASE

Probability of Conflict	60 Events	120 Events	180 Events	240 Events	300 Events	360 Events
.10	5.8	8.0	9.93	12.13	13.8	15.6
.15	7.0	10.23	13.07	15.53	18.0	20.4
.20	8.0	12.07	16.0	19.03	22.2	25.0

Probability of Conflict	% Increase from 60 to 120	% Increase from 120 to 180	% Increase from 180 to 240	% Increase from 240 to 300	% Increase from 300 to 360
.10	38	24	22	14	13
.15	46	28	19	16	13
.20	51	33	19	17	13

TABLE 25

COMPARISON OF EVENT DISTRIBUTION PER TIMESLOT FOR 60 EVENTS

Events	Average # of Events for Prob. .10	Average # of Events for Prob. .15	Average # of Events for Prob. .20
1	19	14	12
2	17	14	11
3	12	12	10
4	8	9	9
5	4	6	8
6	1	4	5
7		1	2
8			1
9			1

TABLE 26

COMPARISON OF EVENT DISTRIBUTION PER TIMESLOT FOR 120 EVENTS

Timeslot	Average # of Events for Prob. .10	Average # of Events for Prob. .15	Average # of Events for Prob. .20
1	25	19	15
2	23	18	15
3	20	17	14
4	18	16	13
5	15	14	12
6	10	13	11
7	6	10	10
8	2	7	9
9		4	8
10		2	6
11			4
12			1

TABLE 27

COMPARISON OF EVENT DISTRIBUTION PER TIMESLOT FOR 180 EVENTS

Timeslot	Average # of Events for Prob. .10	Average # of Events for Prob. .15	Average # of Events for Prob. .20
1	28	21	16
2	27	21	17
3	24	19	16
4	25	19	15
5	21	18	16
6	19	17	14
7	15	15	14
8	12	14	13
9	6	12	11
10	2	10	11
11		7	10
12		4	8
13		2	7
14			5
15			3
16			1

TABLE 28

COMPARISON OF EVENT DISTRIBUTION PER TIMESLOT FOR 240 EVENTS

Timeslot	Average # of Events for Prob. .10	Average # of Events for Prob. .15	Average # of Events for Prob. .20
1	31	23	18
2	30	21	18
3	29	21	17
4	27	21	17
5	25	20	17
6	24	20	17
7	21	19	16
8	18	17	15
9	15	16	14
10	10	15	14
11	6	13	13
12	2	11	12
13		9	12
14		6	11
15		3	10
16		1	8
17			6
18			4
19			1

TABLE 29

COMPARISON OF EVENT DISTRIBUTION PER TIMESLOT FOR 300 EVENTS

Timeslot	Average # of Events for Prob. .10	Average # of Events for Prob. .15	Average # of Events for Prob. .20
1	32	23	19
2	33	25	20
3	31	23	19
4	30	24	19
5	28	22	18
6	27	21	17
7	25	22	18
8	24	20	17
9	21	18	17
10	18	18	15
11	15	17	16
12	9	17	15
13	6	14	14
14	2	12	14
15		11	12
16		7	13
17		4	9
18		2	9

(cont'd)

TABLE 29 (cont'd)

COMPARISON OF EVENT DISTRIBUTION PER TIMESLOT FOR 300 EVENTS

Timeslot	Average # of Events for Prob. .10	Average # of Events for Prob. .15	Average # of Events for Prob. .20
19			7
20			6
21			3
22			1

TABLE 30

COMPARISON OF EVENT DISTRIBUTION PER TIMESLOT FOR 360 EVENTS

Timeslot	Average # of Events for Prob. .10	Average # of Events for Prob. .5	Average # of Events for Prob. .20
1	35	26	20
2	35	25	20
3	33	26	19
4	33	25	18
5	32	24	18
6	30	22	19
7	28	23	19
8	25	23	18
9	25	21	17
10	22	19	17
11	21	20	18
12	16	18	17
13	11	17	16
14	8	14	17
15	4	14	15
16	1	13	14
17		11	13
18		10	13

(cont'd)

TABLE 30 (cont'd)

COMPARISON OF EVENT DISTRIBUTION PER TIMESLOT FOR 360 EVENTS

Timeslot	Average # of Events for Prob. .10	Average # of Events for Prob. .5	Average # of Events for Prob. .20
19		6	11
20		3	10
21		1	9
22			9
23			6
24			4
25			1

TABLE 31

COMPARISON OF NUMBER OF EVENTS MEETING PER TIMESLOT FOR PROBABILITY OF CONFLICT .10

Timeslot	60 Events	120 Events	180 Events	240 Events	300 Events	360 Events
1	19	25	28	31	32	35
2	17	23	27	30	33	35
3	12	20	24	29	31	33
4	8	18	25	27	30	33
5	4	15	21	25	28	32
6	1	10	19	24	27	30
7		6	15	21	25	28
8		2	12	18	24	25
9			6	15	21	25
10			2	10	18	22
11				6	15	21
12				2	9	16
13					6	11
14					2	8
15						4
16						1

TABLE 32

COMPARISON OF NUMBER OF EVENTS MEETING PER TIMESLOT FOR PROBABILITY OF CONFLICT .15

Timeslot	60 Events	120 Events	180 Events	240 Events	300 Events	360 Events
1	14	19	21	23	23	26
2	14	18	21	21	25	25
3	12	17	19	21	23	26
4	9	16	19	21	24	25
5	6	14	18	20	22	24
6	4	13	17	20	21	22
7	1	10	15	19	22	23
8		7	14	17	20	23
9		4	12	16	18	21
10		2	10	15	18	19
11			7	13	17	20
12			4	11	17	18
13			2	9	14	17
14				6	12	14
15				3	11	14
16				1	7	13
17					4	11
18					2	10

(cont'd)

TABLE 32 (cont'd)

COMPARISON OF NUMBER OF EVENTS MEETING PER TIMESLOT FOR PROBABILITY OF CONFLICT .15

Timeslot	60 Events	120 Events	180 Events	240 Events	300 Events	360 Events
19						6
20						3
21						1

TABLE 33

COMPARISON OF NUMBER OF EVENTS MEETING PER TIMESLOT FOR PROBABILITY OF CONFLICT .20

Timeslot	60 Events	120 Events	180 Events	240 Events	300 Events	360 Events
1	12	15	16	18	19	20
2	11	15	17	18	20	20
3	10	14	16	17	19	19
4	9	13	15	17	19	18
5	8	12	16	17	18	18
6	5	11	14	17	17	19
7	2	10	14	16	18	19
8	1	9	13	15	17	18
9	1	8	11	14	17	17
10		6	11	14	15	17
11		4	10	13	16	18
12		1	8	12	15	17
13			7	12	14	16
14			5	11	14	17
15			3	10	12	15
16			1	8	13	14
17				6	9	13
18				4	9	13

(Cont'd)

TABLE 33 (cont'd

COMPARISON OF NUMBER OF EVENTS MEETING PER TIMESLOT FOR PROBABILITY OF CONFLICT .20

Timeslot	60 Events	120 Events	180 Events	240 Events	300 Events	360 Events
19				1	7	11
20					6	10
21					3	9
22					1	9
23						6
24						4
25						1

TABLE 34

COMPARISON OF STANDARD DEVIATION OF THE NUMBER OF EVENTS SCHEDULED PER TIMESLOT FOR PROBABILITY OF CONFLICT .10

Timeslot	60 Events	120 Events	180 Events	240 Events	300 Events	360 Events
1	2.098	2.145	2.966	2.280	2.098	.894
2	2.145	1.732	1.732	2.280	2.280	1.483
3	1.483	2.145	2.191	1.949	2.490	3.0
4	1.844	1.517	1.897	1.870	1.183	3.0
5	1.049	2.121	1.549	2.191	2.145	1.844
6	.447	1.517	1.673	2.470	1.183	3.066

TABLE 35

COMPARISON OF STANDARD DEVIATION OF THE NUMBER OF EVENTS SCHEDULED PER TIMESLOT FOR PROBABILITY OF CONFLICT .15

Timeslot	60 Events	120 Events	180 Events	240 Events	300 Events	360 Events
1	1.095	1.817	1.643	1.673	.775	2.236
2	.775	1.703	1.703	1.612	1.0	1.613
3	.632	1.304	1.483	1.897	1.095	1.949
4	1.095	1.342	1.703	1.517	.632	1.414
5	.447	1.449	1.732	1.673	.447	1.183
6	1.0	1.844	1.581	1.581	1.844	2.145
7	.447	1.183	.775	1.673	.447	1.265

TABLE 36

COMPARISON OF STANDARD DEVIATION OF THE NUMBER OF EVENTS SCHEDULED PER TIMESLOT FOR PROBABILITY OF CONFLICT .20

Timeslot	60 Events	120 Events	180 Events	240 Events	300 Events	360 Events
1	.775	1.304	2.258	1.414	1.096	1.183
2	.775	1.643	1.095	1.549	.633	.447
3	.775	1.949	1.049	1.549	.633	.775
4	1.844	1.378	1.449	1.483	1.732	.775
5	1.342	1.449	1.732	1.673	.447	1.183
6	1.342	1.549	1.449	1.581	1.0	.775
7	1.0	1.612	1.225	1.703	.775	.775
8	.775	1.265	1.612	1.643	1.844	1.549
9	.775	1.095	1.414	1.483	1.549	1.183

TABLE 37

DETERMINATION OF VALUES FOR THE PROBABILITY OF CONFLICT APPLICABLE TO SECONDARY SCHOOL SCHEDULING

	60 classes	120 classes	180 classes	240 classes	300 classes	360 classes
no electives	.10	.05	.03	.03	.02	.02
few electives	.20	.10	.07	.05	.04	.03
moderate electives	NA*	NA	.10	.08	.06	.05
many electives	NA	NA	NA	NA	.08	.07

*NA means that the given situation is not applicable because the school size is too small.

TABLE 38

NUMBER OF REPETITIONS PERFORMED FOR EACH COMBINATION OF PROBABILITY OF CONFLICT AND TOTAL NUMBER OF EVENTS

Number of Events	Probability of Conflict .10	.15	.20
60	5	5	5
120	30	30	30
180	15	15	15
240	30	30	30
300	5	5	5
360	5	5	5

CHAPTER 5

AN INTEGER PROGRAMMING FORMULATION

I will now show how a graph coloring problem can be formulated as a zero-one integer programming problem. In particular, how a timetable scheduling problem can be formulated as an integer programming problem. The formulation used is from Christofides (3).

Let G denote the graph describing a scheduling problem and R be an upper bound on the minimum number of colors needed to color this graph. Upper bounds may be found in Chapter 3 of Christofides (3). Let $C = (C_{ij})$ be an array of numbers allocating vertices to colors so that

C_{ij} = 1 if vertex X_i is of color j.
= 0 otherwise.

Now let $A = (a_{ij})$ be an array of numbers defined by:
a_{ij} = 1 if there exists a line connecting vertex

X_i to vertex X_j

a_{ij} = 0 if there does not exist a line connecting vertex X_i to vertex X_j.

Since no class or exam can conflict with itself $a_{ii} = 0$ for all i. Also $a_{ij} = a_{ji}$ since, if X_i conflicts with X_j, then X_j conflicts with X_i.

The following two conditions ensure a feasible coloring of the vertices of G.

$$(1) \quad \sum_{j=1}^{R} C_{ij} = 1 \quad \text{(for all } i = 1, 2, \ldots, M)$$

$$(2) \quad L\,(1 - C_{ij}) - \sum_{K=1}^{M} a_{iK} C_{Kj} \geq 0$$

(for all $i = 1, 2, \ldots, M$ and $j = 1, 2, \ldots, R$)

Where M is the total number of events to be scheduled.

Condition (1) ensures that a vertex can be colored with one and only one color. In condition (2) L is a very large positive integer (any number greater than n will suffice). The first term of (2) is zero, provided that vertex X_i is colored with color j. In other words, the first term of (2) is zero, if C_{ij} is equal to one. The second term of (2) must also then be zero in order to satisfy the inequality; since both a_{iK} and C_{Kj} are non-negative. Hence, condition (2) ensures that if vertex X_i is of color j then no vertex joined to j is of the same color. If vertex X_i is not of color j, that is, if C_{ij} is equal to zero, then the first term of condition (2) becomes L and since the second term of condition (2) cannot possibly attain the value L (since the maximum value it can attain is M-1), any number of vertices X_K joined to vertex X_i can be colored with color j and the

inequality would still be satisfied. Observe that if X_K' and X_K'' are both joined to vertex X_i and also joined to each other, then condition (2) written in terms of X_i will not prevent X_K' and X_K'' from being colored with the same color j. However, condition (2) written in terms of X_K' (or X_K'') will guarantee that the color of these two vertices is not the same.

Now let us associate with each color j a penalty P_j, the penalties being chosen so that:

$$P_{j+1} > P_j \cdot h$$

where P_1 is one and h is an upper bound on the largest number of vertices that can be colored with any one color, due to the structural nature of the graph. The value of h may be taken to be M.

The problem of coloring the vertices of a graph with the minimum number of colors can thus, be formulated as a zero-one integer programming problem expressed as:

$$(3)\quad \text{Minimize } Z = \sum_{j=1}^{R} \sum_{i=1}^{M} P_j C_{ij}$$

subject to conditions (1) and (2).

The minimization of (3) ensures that the j+1 color is never used for coloring the vertices, if the colors 1 to j are sufficient for a feasible coloring. Hence, this formulation guarantees an optimum solution

to this graph coloring problem. In other words, this formulation guarantees a schedule which will require the minimum number of periods or timeslots.

However, the difficulty with this approach is that for even the smallest realistic problems the number of resulting constraints and variables make the problem numerically unsolvable even on the fastest present day computers. For example, suppose we wanted to schedule 100 events and we knew an upper limit on the number of required timeslots was 10, then condition (1) would yield 100 constraints and condition (2) would yield 1000 constraints. This number of constraints plus the enormous number of variables involved causes this problem to be numerically infeasible using the given integer programming approach.

CHAPTER 6

CONCLUSIONS AND SUGGESTED FURTHER STUDY

The present methods employed for scheduling either classes in secondary schools or examinations in universities appear to be adequate to perform the task. However, these methods are usually time consuming both in terms of personnel and computer usage.

The approach described in this paper is capable of handling large scale scheduling problems with a minimum of actual computer and personnel time needed. The linear regression analyses in Chapter 4 enables the user to make estimates concerning important parameters for a scheduling problem. In particular, a user can predict the total number of timeslots needed, the maximum number of events scheduled per timeslot, and the execution time, and thus, the cost of executing the program. This information is very useful in performing preliminary analysis of what the structure of the schedule will be.

The only difficulty with this method, in its present form, is that it is not geared to take into account additional constraints. It appears plausible that such constraints as teacher preference, teacher availability, etc. should be able to be incorporated into a graph theoretic approach. However, it is not clear to me, at

present, how the additional complexity of the graph will affect the efficiency of the computer code. In any case, further research into refinements of the program COLOR are needed to make it more attractive as a commercial programming package. Also an input program would have to be used to construct the conflict array for an actual scheduling situation instead of a simulated conflict array presently used.

BIBLIOGRAPHY

1. Berge, C., The Theory of Graphs and Its Applications. John Wiley and Sons, Inc., New York, 1962.

2. Broder, S., Final Exam Scheduling. Communications of ACM 7, 494-498 (1964).

3. Christofides, N., Graph Theory An Algorithmic Approach. Academic Press, New York, 1975.

4. Corneil, D.G., and Graham, B., An Algorithm for Determining the Chromatic Number of a Graph. SIAM J. on Computing 2, 4, 311-318 (Dec. 1973).

5. Lions, J., The Ontario School Scheduling Program. Computer J. 10, 14-21 (1967-68).

6. Neufeld, G.A., and J. Tartar, Graph Coloring Conditions for the Existence of Solutions to the Timetable Problem. Communications of ACM 7, 450-453 (1974).

7. Ore, O., Graphs and Their Uses. The L.W. Singer Company, New York, 1963.

8. Salazar, A., and R.V. Oakford, A Graph Formulation of a School Scheduling Algorithm. Communications of ACM 7, 696-698 (1974).

9. Smith, O., The Maintenance of the Opportunity List for Class-Teacher Timetable Problems. Communications of ACM 18, 203-208 (1975).

10. Welsh, D.J.A., and M.B. Powell, An Upper Bound for the Chromatic Number of a Graph and Its Application to Timetabling Problems. Computer J. 10, 85-86 (1967-68).

11. Wood, D.C., A System for Computing University Examination Timetables. The Computer J. 11, 41 (1968).

12. _____, A Technique for Coloring a Graph Applicable to Large Scale Timetabling Problems. The Computer J. 12, 317 (1969).

VITA

Francis J. Vasko, son of Mr. and Mrs. Frank B. Vasko was born on March 23, 1952 in Hellertown, Pennsylvania. In 1966, he completed the eighth grade at St. Theresa's Parochial School. Then he attended Hellertown-Lower Saucon Junior-Senior High School graduating in 1970.

From September 1970 to May 1974 he attended Kutztown State College. While there he received the George B. Hancher Award in both his junior and senior years. He graduated in May 1974 with a B.S. in Secondary Education-Mathematics.

From 1974 to 1976 he pursued courses at Lehigh University leading to a M.S. in mathematics. For the 1974-1975 academic year he held a fellow-in-teaching position at the university. During the 1975-1976, 1976-1977 academic years he taught mathematics at Northampton Junior High School. He was a teaching assistant at Lehigh University in the Engineering College for the academic year 1977-1978. During that time, he pursued courses leading to a M.S. in Industrial Engineering.