COMPUTER SCIENCES DEPARTMENT

UNIVERSITY OF WISCONSIN

1210 W. Dayton
Madison, Wisconsin

SCHEDULER - A PROGRAM THAT USES INSTRUCTOR AND STUDENT PREFERENCES TO FORM TIMETABLES

Eric M. Timmreck

Computer Sciences Technical Report #3

June 1967

ABSTRACT

A computer program which forms timetables from instructors' choices of times and students' choices of courses has been used by the Computer Sciences Department at the University of Wisconsin. Instructors are asked to choose up to four very highly acceptable times, up to four highly acceptable times, and up to four other acceptable times. These time periods are chosen in pairs, triples, etc., if the instructor is teaching two, three, etc., courses. Each student is asked to list the courses he expects to take and a corresponding probability or weight for each. A hill-climbing algorithm uses an evaluation function to move courses into various time periods, attempting to give each instructor as high a preference as possible while at the same time trying to avoid having any pair of any student's courses fall within the same time period. Results to date have been quite satisfactory.

This work was supported by NSF Grant No. GP 6070.

I. Introduction

A computer program has been written to assist the Computer Sciences Department at the University of Wisconsin in scheduling its courses. This was done in order to insure, as far as possible, that graduate students would not be prohibited from taking desired courses because they conflicted with other desired courses. On the other hand, it was desired to give due consideration to the faculty members concerning the times at which they would like to teach. The SCHEDULER program, therefore, balances instructor preferences for time periods against student preferences for courses in searching for a satisfactory timetable.

There have been a number of other efforts concerned with scheduling of courses and of examinations. Gotlieb has proposed a method which would have primary applicability to high schools or grammar schools [8]. Others have developed this method ([5], [11], [6]), but as far as I know it has not been applied to real situations. Appleby, Blake, and Newman [3] also report some successes in this area.

Almond [2], Faulkner [7], and Macon and Walker [12] report on methods which are concerned with university scheduling and sectioning of courses. They are basically concerned with eliminating student conflicts. A powerful system called GASP (Generalized Academic Simulation Programs) has been developed at M.I.T. [1]. Additional methods concerned primarily with examination scheduling are reported by Broder [4] and by Hall and Acton [9].

A program which is somewhat similar to the present effort has been written by Schultz, Brooks, and Schwartz [13] to schedule sessions at conferences. It allows the authors of the papers which have to be presented to choose one primary and one secondary category into which they believe their papers should fall. The attendees of the conference are allowed to choose five sessions, ranked from 1st to 5th, which they would like to attend. The papers are divided into sessions of 8 to 10 papers each. Then the choices of the attendees are used to schedule the sessions.

Most of the methods mentioned are concerned primarily with student conflicts; instructor availability is generally handled by simply listing the time periods for which each instructor is available. The method presented here is new in that it allows both ranked professor preferences of times and weighted student preferences of courses. It is the balancing of these two criteria against each other that allows the searching algorithm to work.

II. The SCHEDULER Program

The SCHEDULER program, written in Fortran 63 for the CDC 1604, consists of seven subroutines. An executive routine moves the courses. Another routine determines at each stage which course to move next. And others are concerned with input, initialization, evaluation of timetables encountered, storing of good timetables, and output. The program can handle up to 750 students, 210 courses, and 50 time periods using a 32K memory.

1. Instructor preferences

Each instructor is sent a list of courses which he is to teach together with a list of standard times during which courses are normally offered. He is then asked to choose up to four 1st choices (very highly acceptable times), up to four 2nd choices (highly acceptable times), and up to four 3rd choices (acceptable times). Time periods other than those listed may be chosen.

If an instructor is to teach more than one course, then his choices are to be considered as pairs, triples, etc., instead of single times. For example, a certain instructor's top preference might be: Course 101 at MWF 1100-1150 _and_ Course 508 at MWF 0955-1045; and another of his preferences might be: Course 101 at MWF 1100-1150 _and_ Course 508 at 1320-1410.

No professor will be assigned any time period which he did not list as one of his preferences. This insures that all times assigned will be acceptable, though not necessarily highly or very highly acceptable.

2. Student preferences

Each student is sent a list of the courses to be offered. He is asked to list the ones he expects to take, together with the probability with which he expects to take each one. If, for example, a student is sure of taking Course A and Course B and will also take either Course C or Course D, then he might assign a probability of 1 to Course A and to Course B and a probability of .5 to Course C and to Course D. Also, if an instructor deems it highly important that a particular student be able to take a particular course, the student's weight for that course can be increased before running the program. (For example, it could be multiplied by 4.)

In cases where a course is divided into lecture sections and quiz sections, the quiz sections should not be included in the input. The student is allowed to choose from the lecture sections only; and it is presumed that he will have room in his schedule for at least one of the quiz sections offered. A student who requests a sectioned course but does not specify which section he wants is placed in the section with the fewest students. If the separate sections of a course contain significantly different material, they should be input as separate courses.

3. Evaluation function

Every timetable which SCHEDULER encounters in its process of searching is evaluated by the following expression called the 'conflict ratio sum':

$$\frac{\sum_{\text{courses}} \text{preference level}}{\#\ \text{courses}} \times \text{FACTOR} + \frac{\#\ \text{students} + \#\text{conflicts}}{\#\ \text{students}}$$

where

a) The preference level for a course will be 1, 2, or 3 depending whether its current time period was a 1st, 2nd, or 3rd choice.

b) Number of courses (counting lecture sections separately) is input by the user.

c) The FACTOR is a user-determined number which indicates how much weight is to be given to the instructors as opposed to the students. The value .2 has been found to weigh instructors and students more or less equally. The larger the factor, the more weight given to the instructors.

d) Number of students is input by the user.

e) The number of conflicts is computed from

$$\sum_{\text{STUDENTS}} \sum_{A,B\in S} \text{Weight (Course A)} \times \text{Weight (Course B)}$$

where S is the set of pairs A, B such that A and B are courses which conflict for a particular student and $A \neq B$, thus using, as a measure of each conflict, the product of the weights which the student has assigned to the two courses.

The program attempts to minimize the conflict ratio sum. Its value will be 1.2 (for a FACTOR of .2) if all instructors have a first choice and there are no student conflicts. Although there are probably many other expressions which would give a reasonable evaluation of a particular timetable, this one has proved to be quite satisfactory.

4. Data

Besides the data previously mentioned (courses to be offered and who is to teach them, instructor preferences for times, student preferences and weights for courses, a 'factor' to be used in the conflict ratio sum), there is some other data which the user must prepare. He must decide whether or not he wants the program to output a trace of its operations; he must form a table indicating which time periods overlap with each other or are too close together to allow sufficient passing time; he must decide upon upper and lower bounds for both the individual weights which the students assign to the courses and the sum of the course weights for any particular student. (These last data are used to insure that the students' weights are not unreasonable either individually or in their sum.) The user must also decide on a maximum number of tries to be allowed, that is, a maximum number of moves of courses from one time period to another in search of a better timetable. In addition, the user is also given some leeway in determining how the searching algorithm is to work; he can vary the parameter (henceforth called TRIALTYP) which determines at any point which courses is the candidate to be moved to another time period. The algorithm determines a time period which it will try to move courses out of; the user-determined parameter mentioned then determines which course is to be moved as follows:

<u>TRIALTYP</u>

1 Move the course contributing the second highest number of conflicts.

2 Move the course contributing the highest number of conflicts.

3 Move the course with the second highest number of students.

4 Move the course with the highest number of students.

5 Move the course with the lowest number of students.

On all trials to date, the value 2 has done quite well; the values 5 and 4 have also done well; and the others have done only a mediocre job.

More specifics on how to collect and set up the data and on suggested values for parameters are contained in [14].

5. Method

The basic searching algorithm used is as follows. Assume that each instructor has his top first choice (actually the first choice which happens to head the list of first choices). Compute the conflict ratio sum. Also compute the number of conflicts corresponding to each time period and the number of conflicts corresponding to each course. Determine which time period has the greatest number of student conflicts corresponding to it. Choose the first course to be moved from that time period on the basis of TRIALTYP (including courses which are in time periods which overlap or are immediately adjacent to the time period in question). Move this course to the next time period indicated in its instructor's preferences (unless it is a pre-fixed course, in which case there is only one preference listed). Also move any other courses taught by the same instructor to corresponding time periods. Then compute a new conflict ratio sum and compare it with the old. If there was improvement, start the entire procedure over again using the new assignment of times to courses as a starting point. If there was no improvment, return to the old starting point and use TRIALTYP to choose another course to be moved, skipping over the one which has just failed.

If all courses contributing conflicts to the time period from which courses are being moved have been tried without improvement, move to the time period with the next highest number of conflicts and start trying to remove courses from it.

If all time periods have been tried and there has been some improvement since the courses were last at top first preferences, they are put back at the top first preferences and the algorithm is restarted from a slightly different starting point. But if there has not been improvement, all courses are moved to the preference one below the preference last tried (cycling if at the botton of the list) and the algorithm is restarted.

Whenever an attempt is made to move a course, the next time period down the list of instructor preferences beyond the preference last tried is used, whether there was improvement or not at the last try. This avoids repeatedly attempting the same move.

A special storage area is used to store good timetables. The conflict ratio sum of each timetable encountered is compared with the highest conflict ratio sum representing any timetable in the special storage area. If the former is lower, its timetable replaces the worst timetable stored. Thus this special storage area always contains the best five timetables encountered. A check is made to insure that no two of these are identical.

After the maximum number of moves for courses (determined by the user) has been reached, the five stored schedules are operated upon further. In each schedule, for each course whose instructor has a preference below level 1 or which has student conflicts, an attempt is made to move this course to all other instructor-listed time preferences on the same or higher levels. If any of these moves produces a timetable with a conflict ratio sum lower than the current one, the old timetable is replaced by the new (provided it is not the same as one of the other four timetables stored).

Other details of the searching algorithm are shown in the flow chart (Fig. 1).

6. Output

The best five schedules for each setting of TRIALTYP and FACTOR are printed out. Each timetable lists for each course the course name, the instructor, the instructor's preference level, the time chosen by the program, a code number which also identifies the time, the number of students, and the number of conflicts. The output also contains a list for each timetable of which students have which conflicts. (See Fig. 2.)

The number of students for each course may be non-integral because of the non-integral nature of the weights which the students are allowed to choose. For example, if a student assigns .5 as a weight for course G, then he is counted as one half of a student for course G.

Conflicts, as mentioned before, are also computed from these weights. For example, if a student has a conflict between Course C (weight .4) and Course D (weight .9), there will be .36 of a conflict (the product of the weights) assigned to each of these courses.

III. Analysis of Results

The SCHEDULER program has been used twice by the Computer Sciences Department at the University of Wisconsin. Two results for each of the two runs are shown in Table 1. The October 1966 run actually used a program which was a forerunner to the one described here. Its results, although quite satisfactory, indicated a number of desirable refinements, which were then made. The number of conflicts in any particular timetable, for example, was previously just tallied up without using student weights. This explains why Table 1 shows such a discrepancy between numbers of conflicts for the two semesters. For the data used in the March 1967 run, the program took about 2 minutes to run from an object deck. Run time is heavily dependent upon the number of courses, the number of students, and the maximum number of iterations allowed.

The results produced by SCHEDULER were interpreted with a few limitations in mind. In both cases in which the program was used, the semester for which the timetables were produced was four to six months away from the time of the run. As a result of this, the list of courses and instructors was tentative. Also, the student choices and weights were of questionable reliability. Then too, the students contacted were primarily presently enrolled graduate students; thus courses usually taken by entering graduate students and by undergraduates were not properly represented in terms of which students would like to take them and what conflicts might therefore arise. But these are problems of data collection rather than limitations of the method itself.

IV. Applications and Implications

There are a number of ways in which SCHEDULER might be used. A single department might use it to assist in scheduling its courses, as has been done; or groups of departments such as mathematics, computer sciences, and statistics or botany, zoology, and chemistry might use it. It might be followed by another program which accepts the suggested timetables as input and then makes room assignments. Or it might be tied in with a student advising program in such a way that the output of the student advising program becomes input to SCHEDULER.

The problem attacked here is basically that of performing a mapping from a set of courses into a set of times - but with the difficulty that a change in one assignment can severely affect the value of the timetable as a whole. There are similar problems in which the assignments from the one set into the other are more independent, such as the assignment of prospective employees to job interviews discussed by Holt and Huber [10]. The problem of matching young people for dates might also fall into this category. It is not clear how useful the techniques of SCHEDULER might be in helping to solve these related problems, but the matter does seem to be worth further study.

V. Summary

A program has been written to assist in the formation of university timetables. Its main feature is that it takes into account both the students' preferences of courses and the instructors' preferences of time periods. Because of the large time gap between the setting up of a timetable for a semester and the semester itself, there were problems concerning the accuracy of the input data. Nevertheless, the program was very helpful to the Computer Sciences Department of the University of Wisconsin in scheduling the courses for the two semesters for which it has been run to date.

ACKNOWLEDGEMENT. I would like to thank Dr. J. B. Rosen and Dr. Leonard Uhr for their encouragement and assistance on this project.

October 1966 (for spring, 1967)

26 courses	TRIALTYP 2
75 students	FACTOR .2
50 iterations	
34 conflicts at top 1st preferences	

Two resulting timetables:

	Student Conflicts		Courses at	
		level 1	level 2	level 3
a)	5	20	6	0
b)	4	16	8	2

March 1967 (for fall, 1967)

32 courses	TRIALTYP 2
51 students	FACTOR .3
75 iterations	
13.455 conflicts at top 1st preferences	

Two resulting timetables:

	Student Conflicts		Courses at	
		level 1	level 2	level 3
a)	2.1	27	5	0
b)	.64	29	2	1

Table 1. Some results produced by SCHEDULER.

KK Indexes through the courses which are the candidates to be moved from a given time period

N Indexes through the time periods from which the courses are to be moved

NN Indicates which time period is to be considered first the next time the search is restarted at top preferences.

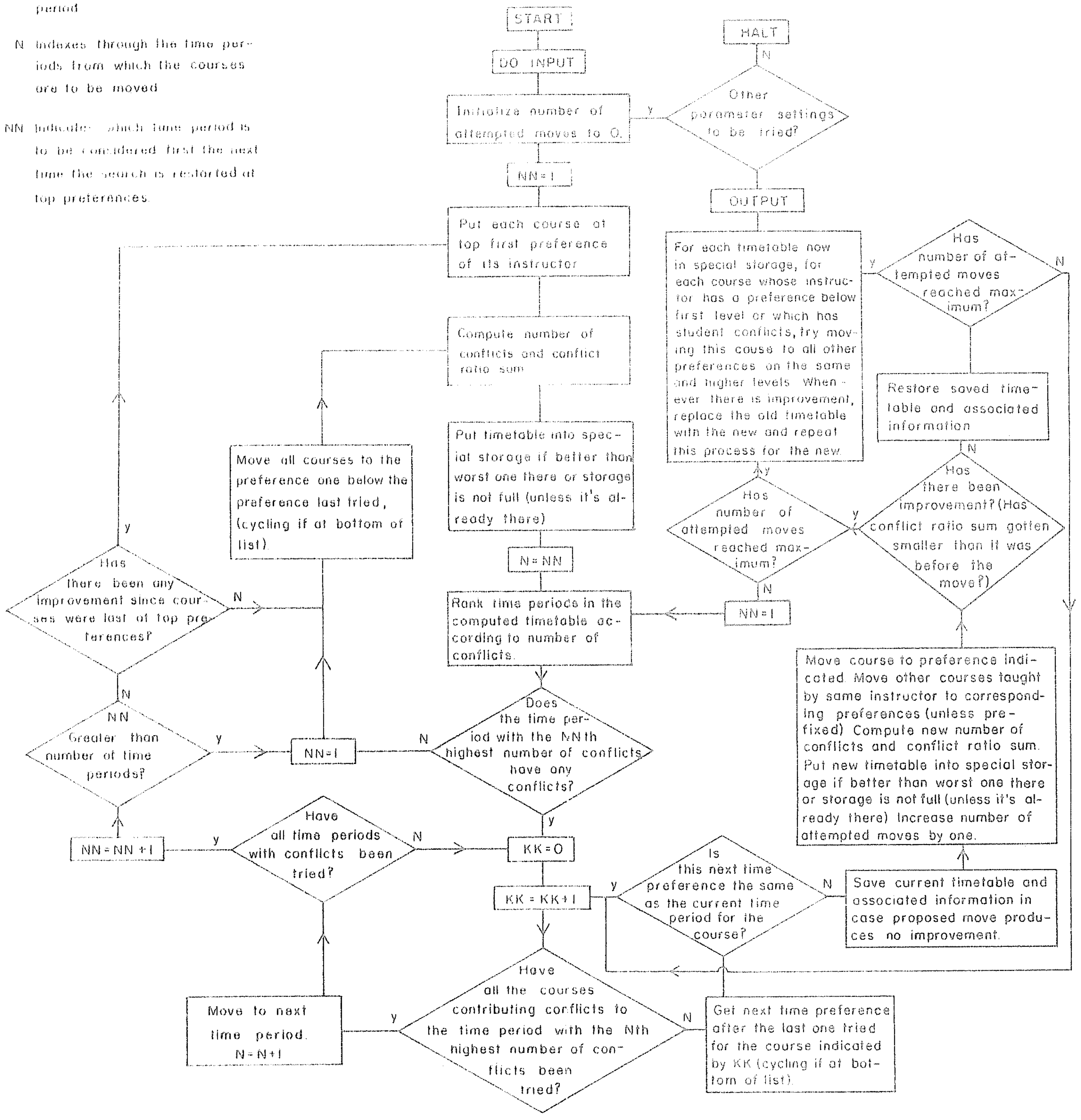
START
DO INPUT
Initialize number of attempted moves to 0.
NN=1
Put each course at top first preference of its instructor
Compute number of conflicts and conflict ratio sum
Put timetable into special storage if better than worst one there or storage is not full (unless it's already there)
N=NN
Rank time periods in the computed timetable according to number of conflicts.
Does the time period with the NNth highest number of conflicts have any conflicts?
Move all courses to the preference one below the preference last tried, (cycling if at bottom of list).
Has there been any improvement since courses were last at top preferences?
NN Greater than number of time periods?
NN=1
NN=NN+1
Have all time periods with conflicts been tried?
KK=0
KK=KK+1
Have all the courses contributing conflicts to the time period with the Nth highest number of conflicts been tried?
Move to next time period. N=N+1
Get next time preference after the last one tried for the course indicated by KK (cycling if at bottom of list).
Is this next time preference the same as the current time period for the course?
Save current timetable and associated information in case proposed move produces no improvement.
Move course to preference indicated. Move other courses taught by same instructor to corresponding preferences (unless prefixed). Compute new number of conflicts and conflict ratio sum. Put new timetable into special storage if better than worst one there or storage is not full (unless it's already there). Increase number of attempted moves by one.
Has there been improvement? (Has conflict ratio sum gotten smaller than it was before the move?)
Restore saved timetable and associated information
Has number of attempted moves reached maximum?
Has number of attempted moves reached maximum?
NN=1
For each timetable now in special storage, for each course whose instructor has a preference below first level or which has student conflicts, try moving this couse to all other preferences on the same and higher levels. Whenever there is improvement, replace the old timetable with the new and repeat this process for the new.
OUTPUT
Other parameter settings to be tried?
HALT
y
N

Figure 1. Flow Chart of SCHEDULER

```
THE 1ST RANKED SCHEDULE FOUND IS AS FOLLOWS.

CONFLICT RATIO SUM =      1.23130
NUMBER OF STUDENT CONFLICTS =      .6400

TRIALTYP =  5             FACTOR =      .20000

  COURSE       PROF          PREF LEV          TIME              TIMECODE   STUDENTS        CONFLICTS
 1 CS 132      DAVIDSON          1         TR  1100-1230              9          0                0
 2 CS 203      PURDOM            1         MWF 0850-0940              2          0                0
 3 CS 302 1    STAFF1            1         MWF 0850-0940              2          0                0
 4 CS 302 2    STAFF2            1         MWF 0955-1045              3          0                0
 5 CS 302 3    STAFF3            1         MWF 1100-1150              4          0                0
 6 CS 302 4    STAFF4            1         MWF 1320-1410              6          0                0
 7 CS 302 5    STAFF5            1         MWF 1425-1515              7          0                0
 8 CS 302 6    STAFF6            1         MWF 1530-1620             11          0                0
 9 CS 412      GOYEN             1         MWF 0850-0940              2          0                0
10 CS 413 1    DANIEL            1         TR  0850-1030              8       3.50                0
11 CS 413 2    HELTON            1         TR  0850-1030              8        .80                0
12 CS 415      STAFF7            1         MWF 0955-1045              3       1.30                0
13 CS 460      HAP               1         TR  0850-1030              8       4.43            .1500
14 CS 465      SLACK             1         MWF 1320-1410              6       2.60            .2400
15 CS 467      COLLINS           1         MWF 1100-1150              4       2.20                0
16 CS 483      RUSSELL           2         MWF 1320-1410              6       6.60                0
17 CS 510      CIOTA             1         MWF 1320-1410              6       3.43                0
```

```
18 CS 525      HU            1      TR  1100-1230        9     6.95         .2500
19 CS 536      WILLIAMS      1      MWF 1425-1515        7     7.80             0
20 CS 717      PARTER        1      MWF 0850-0940        2     4.50             0
21 CS 731      LONDON        3      TR  1320-1500       10     8.20             0
22 CS 761      TRAVIS        1      T   1900-2200       15    13.50             0
23 CS 765      UHR           1      TR  1100-1230        9     6.40         .2500
24 CS 773      KLEIN         1      MWF 1425-1515        7     3.60             0
25 CS 820      MOORE F       1      MWF 1530-1620       11    12.50             0
26 CS 837F     MOORE F       1      MWF 0955-1045        3    10.10             0
 7 CS 837I     NOBLE         1      TR  0850-1030        8     6.10             0
 8 CS 837J     STAFER        1      MWF 1100-1150        4     5.35             0
29 CS 838G     WYLLYS        1      TR  0850-1030        8     2.70         .1500
 0 CS 838H     FIEGE         1      MWF 1100-1150        4     5.90             0
 1 CS 881      CRYER         1      MWF 1320-1410        6     5.45         .2400
 2 CS 885      HALTON        1      TR  1100-1230        9     4.10             0
ONFLICTS IN THE PREVIOUS SCHEDULE ARE AS FOLLOWS.


AUL LAFATA          HAS A CONFLICT BETWEEN CS 465   AND CS 881    OF WEIGHT    .2400.
ARRY D WITTIE       HAS A CONFLICT BETWEEN CS 460   AND CS 838G   OF WEIGHT    .1500.
ARA JORDAN          HAS A CONFLICT BETWEEN CS 765   AND CS 525    OF WEIGHT    .2500.
```

Figure 2. An example of SCHEDULER output.

BIBLIOGRAPHY

1. Anonymous. Generalized Academic Simulation Programs. Manual on work done at MIT.

2. Almond, Mary. An algorithm for constructing university timetables. Comput. J. 8 (Jan. 1966), 331-340.

3. Appleby, J. S., Blake, D. V., and Newman, E. A. Techniques for producing school timetables on a computer and their application to other scheduling problems. Comput. J. 3 (Jan. 1961), 237-245.

4. Broder, Sol. Final examination scheduling. Comm. ACM 7 (Aug. 1964), 494-498.

5. Csima, J., and Gotlieb, C. C. Tests on a computer method for constructing school timetables. Comm. ACM 7 (Mar. 1964), 160-163.

6. Duncan, A. K. Letter to the editor: Further results on the computer construction of school timetables. Comm. ACM 8 (Jan. 1965), 72.

7. Faulkner, Martin. Computer sectioning and class scheduling. Datamation 11 (June 1965), 35-37.

8. Gotlieb, C. C. The construction of class-teacher timetables. Proc. IFIP Congress 62 (Munich, North Holland Publishing Company, 1963, 73-77.

9. Hall, Andrew D., Jr., and Acton, Forman S. Scheduling university course examinations by computer. Comm. ACM 10 (April 1967), 235-238.

10. Holt, Charles C., and Huber, George P. A computer aided approach to employment service placement and counselling. Firm and Market Workshop Paper 6609, Social Systems Research Institute, University of Wisconsin, Dec., 1966.

11. Lions, John. Matrix reduction using the Hungarian method for the generation of school timetables. Comm. ACM 9 (May 1966), 349-354.

12. Macon, N., and Walker, E. E. A Monte Carlo algorithm for assigning students to classes. Comm. ACM 9 (May 1966), 339-340.

13. Schultz, Claire K., Brooks, Alan, and Schwartz, Phyllis. Scheduling meetings with a computer. Comm. ACM 7 (Sept. 1964), 534-541.

14. Timmreck, E. M. A user-oriented description of SCHEDULER. Technical Report No. 2, Computer Sciences Department, University of Wisconsin, Dec., 1966.