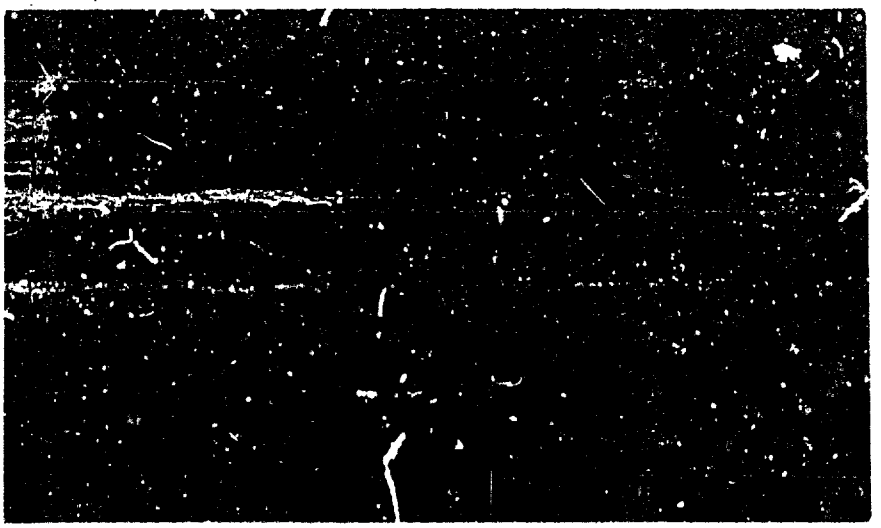
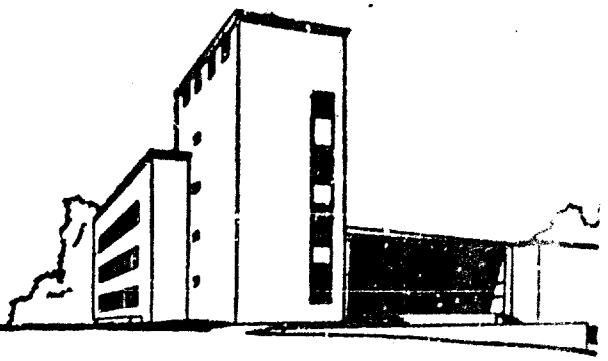
AD620174



Carnegie Institute of Technology

Pittsburgh 13, Pennsylvania



GRADUATE SCHOOL of INDUSTRIAL ADMINISTRATION

William Larimer Mellon, Founder

Management Sciences Research Report No. 46

A METHOD FOR SCHEDULING STUDENTS TO CLASSES

by

Gerald L. Thompson

July, 1965

MANAGEMENT SCIENCES RESEARCH GROUP
GRADUATE SCHOOL OF INDUSTRIAL ADMINISTRATION
CARNEGIE INSTITUTE OF TECHNOLOGY
PITTSBURGH, PENNSYLVANIA 15213

This report was prepared as part of the activities of the Management Sciences Research Group, Carnegie Institute of Technology, (under Contract Nonr 760(24) NR 047-048 with the U. S. Office of Naval Research).

In most universities the mass scheduling of students to classes requires from a half a man-year to several man-years of time, counting only the time of the faculty members and administrators involved. Although the job is spread over a number of individuals, it is an onerous task, that seemingly should be done by a computing machine. Moreover, because most scheduling is presently done in a decentralized way by many people, the only objective that present scheduling techniques can have is that of obtaining a feasible schedule, that is, one that the faculty and students can "live with". One can easily think of other objectives of a good scheduling technique such as: to minimize the number of sections of multiple section courses, while keeping the size of the courses within given bounds; to utilize classroom facilities as efficiently as possible; to preserve as much flexibility as possible in the final schedule to allow for final manual adjustments late registration, course changes, etc. It is reasonable to expect that an efficient computer program for obtaining a feasible schedule could also provide some of these other desirable qualities in a completed feasible schedule.

Solving the scheduling problem for a single student is a relatively simple task for humans. Freshmen learn what the combinatorial problem is and how to deal with it during the first week or so that they are on campus. Moreover, through years of experience, registrars at colleges have learned how to "solve" the college-wide scheduling problem.

A great deal of effort has been expended over the last 8 to 10 years in programming for computers heuristic solutions to college

scheduling problems. The literature cited in the bibliography is typical of some of this effort. An examination of these programs reveal a number of common characteristics. Very little if any search of possible alternatives at a decision point is carred out by these heuristic programs. Instead, decisions are made on an ad hoc basis following the heuristic ideas developed by the author(s) of the programs. Thus such programs are, roughly speaking, just as good as the schedulers who devised them. It is easy to see why these methods do not consider a complete combinatorial search of alternatives for the entire scheduling problem, since that problem is far too big. But it is equally obvious that an improvement over human or straight heuristic performance should be possible if a partial search of the alternatives is carried out.

In the present paper the author proposes a method for solving the problem that combines heuristic and algorithmic ideas. Namely, it uses heuristic ideas for deciding on the order in which to consider the students, and for setting up the actual mathematical problem to be solved, but in making the specific assignment of a student to class sections, an integer programming problem is solved whose objective function is to minimize the sum of the slacks in the sections considered. As far as the author knows this is the first time that heuristic and algorithmic ideas have been combined in this manner.

The proposed method has the following features:

(1) Given (a) a list of the available courses sections, times and maximum enrollments, and (b) for each student a list

of the courses he would like to take together with 2 or 3 hours during the week at which he would like to be free, the program produces schedules for each student, and an over-all feasible schedule for the whole university that keeps within the prescribed bounds (providing such a feasible schedule exists.)

(2) At the heart of the method is a streamlined version of the "stopped simplex method" for solving integer programming problems, developed by the author in [18].

(3) The resulting schedule is not necessarily optimal relative to any objective function. However, several heuristic rules are built into the code which give it high probability of finding a schedule that is optimal or nearly optimal relative to the objective function proposed in the model.

(4) The model as we propose it has a built in objective function that makes it end up with a relatively flexible final schedule which permits final adjustments relatively easily.

(5) The computer time (IBM 7090 or equivalent machine) required to schedule all the students in a university having 15,000 to 20,000 students is estimated to be 2 or 3 24-hour days, i.e., a long weekend. Hence the method is practical on a real-time basis. Alternatively, it could be designed for an "on-line, real-time" application.

(6) The objective function of the model can be changed in various ways by individual users to meet other kinds of scheduling objectives, such as the ones suggested in the first paragraph above.

In the remainder of the paper we first discuss the feasibility model, the solution of which would resemble present scheduling techniques. Next an improved model, the minimax slack model, is described and a computational method for using it is outlined. Finally, computational results with specific examples is presented.

THE FEASIBILITY MODEL

In the present section we shall define a simple linear programming model of the classroom scheduling problem that is similar to present methods in that it requires only feasibility of the resulting student's schedule. There is no assurance that if the feasibility method were used over and over again to find schedules for all the students an overall feasible schedule for the whole university would be found having any given desirable property. We discuss the feasibility model to imitate present scheduling techniques and to set the stage for an improved model that we will discuss in a subsequent section.

Assume that a given student has indicated the courses that he wants to take plus his free times. To define the linear programming problem of the feasibility model of the scheduling problem, let c be the course number, s the section number, and t the time it meets. Then define

the variables[1/]

$$(1) \qquad x(c,s,t) = \begin{cases} 1 & \text{if the student takes course } c, \text{ in section } s, \text{ meeting at time } t \\ 0 & \text{otherwise.} \end{cases}$$

We then define the following constraints on these variables:

$$(2) \qquad \sum_{s,t} x(c,s,t) = 1 \quad \text{for each course } c \text{ desired}$$

This constraint insures that the student will be signed up for exactly one section of course c. To insure that he will not be signed up for more than one course at the same time, we also require:

$$(3) \qquad \sum_{c,s} x(c,s,t) \leq 1 \quad \text{for each } t.$$

We also add constraints of the form

$$(4) \qquad \sum_{c,s} x(c,s,t_i) = 0 \quad \text{for each } t_i,\ i=1,\ldots,k,$$

where k is the number of free times and t_i is the time of the i^{th} free time. Also in case the student wishes to take a course that has only one section, a constraint of the form (4) will be added for each such course. Since most of the courses that juniors and seniors take are one section courses, only a trivial scheduling problem exists for them.

1/ This method of describing the problem is not the form which is best to write a computer program; see later examples.

We also define the variables B(c,s,t) whose initial values are the numbers of seats in course c, section s, meeting at time t. If a student is assigned to that section, the number is reduced by 1. In order to close off a section when there are no more empty seats in the room we add the restriction

$$x(c,s,t) \leq B(c,s,t) \tag{5}$$

When $B(c,s,t) = 0$, this constraint gives $x(c,s,t) = 0$ so that no more students will be assigned to that section.

Inequalities (1) - (4) together with the fact that $0 \leq x(c,s,t) \leq 1$ define a convex set of feasible vectors. We want to find a point in the set that makes all of the values of $x(c,s,t)$ equal to 0 or 1. The computational method used is the stopped simplex method for integer programming described by the author in [1]. To turn the problem into an integer programming problem it is necessary to define an objective function. Exactly what objective function is unimportant for the present model, so we choose the function:

$$\text{Minimize} \quad \sum_{c,s,t} x(c,s,t) \tag{6}$$

We know, in fact, that for any feasible solution the value of the objective function is p where p is the number of courses the student wants to take. Hence the objective function does not rule out the choice of any feasible integer solution vector.

The model defined by (1)-(6) will be called the feasibility model. Because of the form of the objective function, the S^2 method (for stopped simplex method) will simply pick out some feasible schedule and assign it to the student. This resembles in many ways the present methods of hand scheduling, except that hand schedulers may make some attempt to even the load on various sections so that the "most desirable" sections will not be closed out early in the scheduling process, leaving only the "less desirable" ones for late registrations.

Exactly which feasible schedule the S^2 method will choose depends very heavily on the order in which the constraints for the various sections are listed in the problem. If, for instance, the sections were always listed in the same fixed order, there would be a tendancy for sections listed first to be filled up first, then those listed next, etc.

Because of the above objections to the feasibility model, it was not tested out, and instead the minimax slack model, to be described next, was developed.

THE MINIMAX SLACK MODEL

The problem of any scheduling method is to balance the conflicting desires of (a) the students, and (b) the registrar (representing the faculty and administration of the university.) Most students would like to have their classes begin after 10 a.m. and end by 3 or 4 p.m. They also would like not to have Saturday classes. Naturally such schedules cannot be obtained for all students simultaneously. However, it is an empirical observation of many students that, with the present scheduling

techniques, more of their desires can be fulfilled if they register early than if they register late. It seems that a good scheduling technique should treat students on an equal basis as far as it is possible, regardless of when they arrive in the scheduling queue, so long as they meet some registration deadline.

The main objective of the registrar is the construction of a feasible schedule for the whole university having the property that every student takes the courses he wants without conflicts in his scheduled times, and so that more students are not assigned to a classroom than there are empty seats available. After the main objective has been obtained, the registrar would like to have other objectives such as flexibility, classrooms used efficiently, etc.

The feasibility model of the previous section might be sufficient to obtain an overall feasible schedule if care were taken on the order of introduction of constraints, or alternatively, the entire problem were passed through the computer several times. We propose instead a method which will obtain simultaneously the main objective of a feasible schedule plus the secondary objective of retaining as much flexibility as possible for late registration and course changes.

The minimax slack model utilizes expressions (1)-(4) of the previous section (but not (5) and (6)). Also new variables $x(c)$ are added, one for each course. Then the constraints

$$x(c) \geq B(c,s,t) - x(c,s,t) \quad \text{for each } s,t \tag{7}$$

require that x(c) be greater than either (a) B(c,s,t) or (b) B(c,s,t) - 1 depending upon whether x(c,s,t) is 0 or 1. In other words, x(c) is greater than or equal to the maximum slack of any section of course c, where we shall mean by slack the number of unassigned places in a given section. We now state the objective function as

$$\text{Min} \sum_{c} x(c), \tag{8}$$

which means that the objective is to minimize the sum of the maximum slacks in all of the courses. Thus, in selecting sections for a given student, the simplex method will try to select sections with maximum slack, if it is feasible to do so.

In addition we built into the simplex method the rule that whenever several vectors can be brought into the basis, the one with most negative slack is chosen to be brought in. This is a priority rule that tries first of all to choose the class section with maximum slack as the first try at assigning the section for that class. We consider the most negative slack rule as part of the minimax slack model. It would also be possible to further refine the rule to make the priority method for choosing come-in vectors for the simplex method be in the order of decreasing slack. The latter has not yet been implemented in the program.

Equations (1)-(4), (7) and (8) together with the most negative indicator rule, constitute the minimax slack model. Since the variables must have integer values, the model is an integer linear programming model.

In the next section we shall indicate a method for finding solutions to the problem, but first we shall give a simple example.

Suppose that the student wants to take three courses, English, Mathematics, and History, all of which meet on Monday, Wednesday, and Friday. Suppose that the available times are as in Figure 1.

Course	English			Mathematics		History	
Times	9	10	11	10	11	9	10
Slacks	10	10	9	6	7	5	5

Figure 1

This means that the English section meeting at 9 has 10 unassigned seats, the mathematics section at 11 has 7 unassigned seats, etc. We shall set up the corresponding linear integer programming problem using a notation that is less cumbersome than that in equations (1)-(8). Let E, M, and H be the names of variables corresponding to the three courses, and let subscripts on these letters be the times that the sections meet. Thus E_9 is English at 9, M_{11} is Mathematics at 11, etc. Then the minimax slack model consists of the expressions (9)-(23)

$$E_9 + E_{10} + E_{11} = 1 \tag{9}$$

$$M_{10} + M_{11} = 1 \tag{10}$$

$$H_9 + H_{11} = 1 \tag{11}$$

$$E_9 + H_9 \leq 1 \tag{12}$$

$$E_{10} + M_{10} \leq 1 \tag{13}$$

$$E_{11} + M_{11} + H_{11} \leq 1 \tag{14}$$

$$X_E \geq 10 - E_9 \tag{15}$$

$$X_E \geq 10 - E_{10} \tag{16}$$

$$X_E \geq 9 - E_{11} \tag{17}$$

$$X_M \geq 6 - M_{10} \tag{18}$$

$$X_M \geq 7 - M_{11} \tag{19}$$

$$X_H \geq 5 - H_9 \tag{20}$$

$$X_H \geq 5 - H_{11} \tag{21}$$

$$X_E + X_M + X_H \leq t \tag{22}$$

$$\text{Minimise} \quad t \tag{23}$$

Here expressions (9), (10), and (11) correspond to (20; (12), (13), and (14) to (3); and (15)-(21) correspond to (7). Expressions (22) and (23) need some explanation. Instead of putting the objective function in the form (8) we add the constraint (22) which involves a new variable t, and then seek to minimize t, as in (23). The reason for doing this is to

put the problem in prepared form so that the stopped simplex method, described in the next section will be able to work on it immediately. It is clear that (22) and (23) are equivalent to (8).

Because (9), (10) and (11) are equalities, they will be rewritten as a pair of inequalities when put into the final linear programming model. (Actually, it is well known that there are other ways of handling equalities in linear programming, but this particularly simple method is used in order to keep down the number of variables in the problem). Hence the resulting linear programming problem will have 11 variables and 17 constraints. Note that all the coefficients of the variables are either 1, -1, or 0.

It might be conjectured that the above linear programming problem would have a solution given by the simplex method in integers, but, unfortunately this is not the case and recourse must be had to some kind of an integer programming algorithm.

In general it can be shown that if a student wants C courses, such that course i has S_i sections ($i=1,\ldots, C$), which meet at T different times, then the resulting linear integer programming problem will have $S + C + 1$ variables and $T + 2C + S$ constraints, where $S = \sum_{i=1}^{C} S_i$. Thus the typical size of a problem for a student who wants 5 courses in a medium sized college would be around 30 variables and 40 or soconstraints and perhaps considerably greater (see the example discussed later.) Such a problem is an integer programming problem of rather substantial size.

THE COMPUTATIONAL METHOD

As the previous example illustrates, the solution technique for the problem requires an integer programming technique that can solve fairly big problems rapidly. To get an estimate of how fast the method must be, note that if students can be handled at the rate of one a minute then in one 24 hour period only 1,440 students will be scheduled. This would be prohibitively slow, except for very small colleges. To be able to schedule a large university with, say, 15,000 to 25,000 students, it is necessary to be able to schedule students about ten times this fast, on the average. That is, the scheduling technique should handle a student in about 6 seconds on the average. We shall next describe such a method.

In [18] the author describes the stopped simplex method for solving integer programming problems. We shall describe a streamlined version of that method here for solving the above described scheduling problems.

In order to describe the special version of the stopped simplex method we start with a linear programming problem in minimizing form:

$$\begin{aligned} &\text{Min } wb \\ &\text{Subject to} \\ &\quad wA \geq c \\ &\quad w \geq 0 \end{aligned} \tag{24}$$

We assume that w is $1\times m$, A is $m\times n$, b is $m\times 1$, and c is $1\times n$. Also the components of A, b, and c are assumed to be integers (without loss

of essential generality.) We put (24) into what we shall call prepared form, by adding one more variable w_{m+1}, and extending A, b, and c as follows

$$A^* = \begin{pmatrix} A & 0 \\ 0 & 1 \end{pmatrix}, \quad b^* = \begin{pmatrix} -b \\ 1 \end{pmatrix}, \quad c^* = (c, 0) \tag{25}$$

and the new dimensions are A^* is $(m+1) \times (n+1)$, b^* is $(m+1)\times 1$, and c^* is $1\times(n+1)$. Then the prepared form of (24) is

$$\begin{aligned} &\min \ w_{m+1} \\ &\text{subject to} \\ &wb^* \geq 0 \\ &wA^* \geq c^* \\ &w^* \geq 0 \ . \end{aligned} \tag{26}$$

Note that $wb^* = w_{m+1} - wb$ so that minimizing w_{m+1} in (26) is entirely equivalent to minimizing wb in (24).

In the stopped simplex method certain of the variables are held fixed and the resulting linear programming problem is solved using the remaining variables. To set up a notation for this we let

$$w = w^{(k)} + s^{(k)} \tag{27}$$

where

$$\begin{aligned} w^{(k)} &= (w_1, \ldots, w_k, 0, \ldots, 0) \\ s^{(k)} &= (0, \ldots, 0, s_{k+1}, \ldots, s_{m+1}) \end{aligned} \tag{28}$$

Here the entries in $s^{(k)}$ are held fixed during the course of solving a given linear programming problem. Specifically, suppose that $s^{(k)}$ is a fixed vector with nonnegative integer components. Then the linear programming problem $P^{(k)}$ to be solved is

$$
\begin{aligned}
&\text{Min } w_k \\
&\text{Subject to} \\
(29) \qquad P^{(k)}: \qquad & w^{(k)} b^* \geq - s^{(k)} b^* \\
& w^{(k)} A^* \geq c^* - s^{(k)} A^* \\
& w^{(k)} \geq 0.
\end{aligned}
$$

Note that (26) is problem $P^{(m+1)}$.

It should be remarked that problems of the form (29) are <u>degenerate</u> in the usual programming sense and one of the standard techniques of linear programming must be used to prevent cycling or slow convergence in the simplex method.

We shall use the notation $\langle x \rangle$ to stand for "the smallest integer $\geq x$". For instance, $\langle -1/2 \rangle = 0$, $\langle 1/2 \rangle = 1$, $\langle -2 \rangle = -2$ and $\langle 2 \rangle = 2$.

We shall shortly give a formal description of the stopped simplex algorithm. But first we give a brief informal description. The method begins by solving $P^{(m+1)}$. Then it sets $s_{m+1} = \langle w_{m+1} \rangle$ and solves $P^{(m)}$. Then it sets $s_m = \langle w_m \rangle$ and solves $P^{(m-1)}$, etc. At some point, say in the solution of $P^{(k)}$, the problem may have no solution. When this happens, the method marks the $(k+1)$st variable a failure, replaces s_{k+1} by $s_{k+1}+1$

and resolves $P^{(k)}$. When a variable has been marked a failure twice it is "unstopped" and k is increased by 1, and the process continued. The whole process is stopped when $k=0$. The mathematical basis for the above search process is discussed in [18]. Note that the lower bound calculations of that reference are not employed here.

A formal description of the special version of the stopped simplex method for the classroom scheduling problem is as follows:

0. $k = m+1$.
1. Solve $P^{(k)}$. If there is a solution go to 2; if not go to 4.
2. Let h be the index such that $w_h, \ldots, w_k$ are zeros and w_{h-1} is not a zero, or else $h=1$. If $h=1$ print out answer and halt. If $h > 1$ go to 3.
3. Set $s_i = w_i$ for $i=h, \ldots, k$. Set $k = h-1$. Set $s_k = \langle w_k \rangle$. Go to 1.
4. Set $k = h+1$; if $k < m+1$ go to 6. If not go to 5.
5. If $k = m+1$ replace s_{m+1} by $s_{m+1}+1$. Set $k = m$. Go to 1.
6. Mark the $(h+1)$st variable a failure.
7. If the kth variable has been marked a failure twice to to 9; otherwise go to 8.
8. Replace s_k by s_k+1. Set $h=k-1$. Go to 1.
9. Set $s_k = 0$, replace k by $h+1$. Go to 4.

As it stands, the algorithm will find just one solution to the scheduling problem; it can be modified to find all solutions if desired.

Computational experience with the above algorithm for solving the minimax slack model of classroom scheduling will be discussed in the next section.

It should be observed that the above algorithm and the minimax slack model solve the problem of scheduling <u>one</u> student optimally (in the sense of the minimax slack model). But what does this mean about the problem of scheduling <u>all</u> the students optimally? It is, of course, impossible to set up the scheduling problem for all students at a university at once, since that would lead to an integer programming problem having billions and billions of variables.

What we do instead is to use heuristic ideas to extend the minimax slack model to a scheduling technique for an entire university. The ideas employed in the general program are as follows:

1. Put the desired course and free time list for each student on tape.
2. Read in the desired courses and free times for a clock of students from tape (as many as the core memory can handle.)
3. Set up the integer programming problem for a given student and solve it using the S^2 method. Print out his schedule, and update the slacks of the courses to which he is assigned. (In case no feasible schedule exists for a given student because of filled sections, remove the student's free time options and try again. If there is still no solution, print out that fact so that the student can be scheduled by hand after consultation with him.)

4. If all students are scheduled, halt. Otherwise go to 3.

Other heuristics are possible in order to simplify the resulting scheduling problems and speed up the process. We list some of these next.

H1. If the two occurrences of the word "zero" in step two of the stopped simplex algorithm described above are replaced by the word "integer," then the resulting code is not an algorithm. But is a very good heuristic and will end up with feasible, if not optimal schedules. It will also speed up the time to get an answer which will with high probability be optimal. The computational results reported on in the next section are with a code that has this replacement made.

H2. Consider the students in the following order: special students, seniors, juniors, sophomore, freshmen. (The rationale behind this heuristic is that the special students will be the most difficult to schedule since they tend to want non-standard courses. The juniors and seniors will be very easy to schedule since they tend to take one-section courses. The sophomores and freshmen are the most time consuming to schedule since they tend to take large section introductory courses and hence have the most feasible schedules. In terms of computation time they are the most difficult for the model.)

H3. If a course has more than one section meeting at the same time, choose the one with the largest slack.

H4. If a course has (after 2) more than 5 sections, choose a subset of 5 at random, perhaps weighted in the order of increasing slack. Alternatively, let each student pick out a subset of 5

sections he would be willing to accept.

H5. If there are still too many feasible schedules, let the student pick 3 or even 4 hours he wishes to have free. In order to prevent too many students picking the same free hours, divide them into priority classes and permit the student to pick at most one from each class.

H6. To further speed up the computation, permit each student to propose a feasible schedule he would like to have and let the code use the schedule as an initial start for the problem. If it is feasible and optimal the student will get that schedule, but if not, the program will alter it to an optimal schedule.

H7. The scheduling problem can be decomposed into two parts by arbitrarily choosing certain courses to be on M/F and others on TTS. Then two smaller problems result which can be solved very quickly. Some sort of priority rules are needed to require the student to have some courses on TTS.

Obviously many other possible heuristics can be used. In each school there will be special situations which will give rise to other heuristic ideas for working with the model. The actual development of the model will vary from school to school.

COMPUTATIONAL RESULTS

The stopped simplex method for solving minimax slack problems has been programmed for an electronic computer (7090) and a number of examples

have been solved with it. For instance, the problem of Figure 1 was solved (exclusive of problem setup times) in 1.2 seconds which includes time for intermediate printouts. In the course of its solution, 3 stopped linear programming, problems were set up and solved and a total of 31 pivots were made. This is, of course, a very simple example. but it does correspond to the scheduling problem of an upperclassman, since it is likely that at least 2 of his 5 courses would be one section courses and only the remaining 3 would be multiple section courses.

A more complicated example is shown in Figure 2.

English

	MWF 9	MWF 11	MWF 2	TTS 8	TTS 9	TTS 10
(a)	18	17	18	17	17	18
(b)	18	18	18	18	18	18

Mathematics

	MWF 8	MWF 9	MWF 11	MWF 2	TTS 9	TTS 10
(a)	19	19	19	20	19	19
(b)	20	20	20	20	20	20

History

	MWF 8	MWF 10	MWF 1	TTS 10
(a)	17	17	17	17
(b)	17	17	17	17

Economics

	MWF 11	MWF 2	TTS 9
(a)	12	12	12
(b)	12	12	12

Art Laboratory

	TT 1-3	TT 3-5	MW 1-3
(a)	10	10	11
(b)	11	11	11

Figure 2

In that figure are shown the sections open to five different courses. It is assumed that the student desired MWF 4 and TTS 11 as free times, so that sections of the courses open at these hours are not shown. Underneath each section two different assumptions concerning slacks are labelled (a) and (b). Note that in (a) the slacks are not all the same in all the course sections, but in (b) all slacks are the same for each section of the same course. It is expected that (a) would be the most typical situation, and it turns out to be much easier for the stopped simplex method to solve (a) than (b). In either case the problem leads to an A* matrix (see (25)) of dimensions 28x40. The computer times, exclusive of problem setup times, are given in Figure 3

Problem (a)	8.4"	LP 4	Pivots 73
Problem (b)	22.8"	LP 9	Pivots 210.

Figure 3

If we regard the problem of Figure 1 as typical of the scheduling problem posed by an upper classman, and that of Figure 2 as typical of a lower classman (freshman or sophomore), and also regard (a) and much more typical than (b), then a rough average estimate for all students would be approximately 6 to 7 second per student, including problem setup and printout of answer times. Because of the limited experience the author has had with this algorithm this should be regarded as merely an estimate, not a firm time. More accurate estimates will await the actual development of the method in a specific scheduling application.

CONCLUSION

With present computer speeds, according to the above estimates it would be possible to schedule about 15,000 students in one 48 hour period. Hence the method is feasible on a real time basis for moderate sized universities. Moreover, the resulting schedule should be considerably better than present schedules in terms of flexibility. Because of time pressures, it probably would be out of the question to reschedule using, say, a different order of introducing the students, or using different heuristics, to see if a more desirable solution might be found. However, if computing machine times increase by a factor of 10, the same job could be done in 4.8 hours so that it might be possible to make several reruns. And if computer speeds are improved by a factor of 100, it would be possible to do the above scheduling job in 1/2 hour, so that there would be considerable leeway in rerunning problems with various combinations of heuristics.

BIBLIOGRAPHY

[1] Akin, G., "7070/74 and 1401 Class Scheduling", Guide General Program Library 12.9.004, 50 pages and flow diagrams. Author's address: IBM Data Center, 80 East Lake Street, Chicago, Illinois.

[2] Appleby, S. J., D. V. Blake, and E. A. Newman, "Techniques for Producing School Time tables on a Computer and their Application to other Scheduling Problems," The Computer Journal, 3, (1961) pp. 237-245.

[3] Blakesley, J. R., "Computer Scheduling at Purdue University," Master's Thesis, Purdue University, 1963.

[4] Bossert, W. H., J. B. Harmon, M. L. Bullock, "Student Sectioning on the IBM 7090", SHARE Distribution No. 1594 XYZ SSCP (PA), June 1. 1963, 73 pages.

[5] Csima, J. and C. C. Gotlieb, "Test on a Computer Method for Constructing School Time-tables," Comm. ACM, 1, (1964) pp. 160-163.

[6] Gotlieb, C. C., "The Construction of Class-Teacher Time-tables," Proceedings of IFIP Congress 62, Munich, North Holland Pub. Co., Amsterdam (1963) pp. 73-77.

[7] Haga, E. J., "Automatic Class Scheduling at the University of Rhode Island," Journal on Business Education, 39 (1963) 120.

[8] Harding, R. E., "Linear Programming as an Aid in Solving Scheduling Problems," A Progress Report, Dept. of Indus. Engineering and Grad. Sch. of Bus., 1963.

[9] Helmers, F. T., "Generation of an Educational Master Schedule- a Linear Programming System," M. S. Thesis, Dept. of Indus. Engr., A. D. 438-66071, Univ. of Pittsburgh, 1964.

[10] Holz, R. E., "The GASP Manual," June 1963. Available through the author, R. E. Holz, Associate Registrar, Mass. Inst. of Tech., Cambridge, Mass.

[11] Holz, R. E., "School Scheduling - a Prospectus," Available as in [10].

[12] Holzman, A. G., W. R. Turkes, "Optimal Scheduling in Educational Institute," Cooperative Research Project, No. 1323, Univ. of Pittsburgh, 1964.

[13] Sherman, G. R., "A Combinatorial Problem Arising from Scheduling University Classes," Journal of the Tennessee Academy of Science, 38, (1963) pp. 115-117.

[14] Sherman, G. R., "Combinatorial Scheduling: In Finding a Partition of a Finite Set which Maximizes a Set Functions," Ph.D. Thesis, Purdue Univ., 1961.

[15] Sherman, G. R., "The Sequential Method of Scheduling Students," Computer Research Report, Purdue Research Foundation, 1958.

[16] Skilling, H. H., "Computer Class Scheduling: Experience at the University of Massachusetts," Office of Institutional Studies, University of Mass., Amherst, Mass.

[17] Stockman, J. W., "A Guide to Automated Class Scheduling," Data Processing Magazine, 6 (1964) pp. 30-33.

[18] Thompson, G. L., "The Stopped Simplex Method: I. Basic Theory for Mixed Integer Programming; Integer Programming," Revue Francaise de Recherche Operationelle (1964) pp. 159-182.