A Scheduling Algorithm for a Computer Assisted Registration System

W.K. Winters*
University of Tennessee, Knoxville

This paper presents the scheduling algorithm used in the Computer Assisted Registration System at the University of Tennessee. Notation is defined and the logic of the algorithm necessary to implement educational policy is described. Results from the first term's implementation are presented.

Key Words and Phrases: computer assisted registration, scheduling algorithm, timetable
CR Categories: 3.32, 5.39

1. Introduction

Scheduling students at a large university today is only part of a much larger system which involves students, faculty, and administration as well as the resources of the institution. It is important to realize that any scheduling algorithm, if it is to be used realistically at an institution of higher education, must satisfy the goal of implementing the kind of educational policy that is chosen or already established at the institution [1].

What policy or policies should constitute the criteria to be satisfied when an algorithm is developed? The primary goal of the institution is to provide all students with the courses needed to work toward completion of their educational program and to make an effective use of the resources available to the institution [2]. With this major goal in mind, the scheduling algorithm which makes up the scheduling phase of the Computer Assisted Registration System at the University of Tennessee is described.

2. Notation to Describe the Algorithm

One of the phases of the Computer Assisted Registration System that is a necessary prerequisite to the scheduling phase is the timetable construction [3]. It is from this timetable that students will make requests for classes. It is therefore convenient to denote this timetable by C. C can be practically described in a number of ways. The most common timetable is composed of a time configuration in which a class will meet. It will also have other necessary class descriptors such as the course and section numbers, the name of the course, the number of credit hours, the place the class will meet, and the name of the instructor teaching the class. The time configuration will be specifically referred to as a time string in this paper. The time configuration is of the form, for example, MWF 8–8:50. That is, it is composed of a day sequence and a time interval description.

A time string is composed of the elements of the set $\{0, 1\}$. Each position of the string corresponds to a day or time interval (usually in minutes). 0 or 1 being the only allowable elements of the time string, the string is a binary bit string which uniquely defines the time at which a class will be offered. For example, supposing that a class has the time configuration MWF 8–8:50, the time string $(x_1, x_2, \cdots, x_{16})$ would have the form

$$(x_1, \cdots, x_6, x_7, \cdots, x_{16}) = (101010100000000),$$

where

$$x_i = \begin{cases} 1 & \text{if position } i \text{ is occupied in the time configuration,} \\ 0 & \text{otherwise.} \end{cases}$$

Positions $i = 1, 2, \cdots, 16$ will refer, respectively, to Monday, Tuesday, $\cdots$, Saturday, 8–8:50, 9–9:50, $\cdots$, 5–5:50.

The algorithm to be described will attempt to satisfy the set of students S and their request W for classes for a coming term. To make this presentation more concrete the following notation will be introduced.

* Present address: Oklahoma State Regents for Higher Education, Oklahoma City, OK 73105

[1] The rationale for the justification of this difficulty code is the fact that the more sections that are available for a course the easier it is to schedule a student in one of the sections of the source.

C is the set of all classes in the timetable of classes:
$C = \{c_{ij} \mid c_{ij}$ is section j of course i;
$i = 1, 2, \cdots, n,\ j = 1, 2, \cdots, n_i\}$.
n is the number of courses in the timetable.
n_i is the number of sections of the ith course in the timetable.
$N = \sum_{i=1}^{n} n_i$ is the total number of classes in C.
S is the set of all students to be scheduled:
$S = \{s_k \mid s_k$ is the kth student; $k = 1, 2, \cdots, m\}$.
m is the number of students to be scheduled.
t_{ij} is the total number of seats available for section j of course i.
$T_i = \sum_{j=1}^{n_i} t_{ij}$ is the total number of available seats for the ith course.
r_{ij} is the total number of students currently scheduled for the jth section of course i.
$R_i = \sum_{j=1}^{n_i} r_{ij}$ is the total number of students currently scheduled in the ith course.
d_{ij} is the demand for section j of course i.
$D_i = \sum_{j=1}^{n_i} d_{ij}$ is the total student demand for course i.
$u_{ijk} = \begin{cases} 1 & \text{if the } k\text{th student is scheduled in the } j\text{th section of course } i, \\ 0 & \text{otherwise.} \end{cases}$
q_{ij} is the number of seats remaining in section j, course i.
$v_{ijk} = \begin{cases} 1 & \text{if the } k\text{th student requests the } j\text{th section of course } i, \\ 0 & \text{otherwise.} \end{cases}$
To each $c_{ij} \in C$ there is associated a "difficulty code." This code is computed for a course C_i simply by counting the number of unique time strings for this course in the timetable.
p_i is the difficulty code associated with the ith course C_i. The smaller the absolute value of p_i the more difficult[1] it is to schedule the ith course.
$E = \{e^1, e^2, \cdots, e^y\}$ is the set of unique time strings in the timetable C. e^k_{ij} is the kth time string associated with the jth section of the ith course. It is of the form $e^k_{ij} = (x_1, x_2, \cdots, x_{16})$.
y is the total number of time strings in C.
$W = \{w_1, w_2, \cdots, w_q\}$ is the request set for a student. (The w_i are chosen from the available classes defined by C.)
$W_s = \{w_{i_1}, w_{i_2}, \cdots, w_{i_{q_1}}\}$ is the subset of W consisting of the scheduled sections in the request set.
$W_u = \{w'_{j_1}, w'_{j_2}, \cdots, w'_{j_{q_2}}\}$ is the subset of W consisting of the unscheduled courses in the request set.
q_1 is the number of scheduled courses in the request set.
q_2 is the number of unscheduled courses in the request set.
$q = q_1 + q_2$.

There are several assumptions which must be made. The set of students S will be scheduled in a sequence. The sequence is determined from the students level. Let S_G, S_R, S_J, S_P, S_F denote, respectively, the sets of graduate students, seniors, juniors, sophomores, and freshmen. That is, if
$S_G = s_1, s_2, \cdots, s_g$,
$S_R = s_{g+1}, s_{g+2}, \cdots, s_r$,
$S_J = s_{r+1}, s_{r+2}, \cdots, s_j$,
$S_P = s_{j+1}, s_{j+2}, \cdots, s_p$,
$S_F = s_{p+1}, s_{p+2}, \cdots, s_f$,
then the input sequence for scheduling is
$s_1, \cdots, s_g, s_{g+1}, \cdots, s_r, s_{r+1}, \cdots, s_j, s_{j+1}, \cdots, s_p, s_{p+1}, \cdots, s_f$.

Students make section requests for courses from the timetable of classes. There is a c_{ij} corresponding to each w_i unless the student has not made an invalid request from C. Each section j of course i has a unique value with no course having a unique value unless it is a single section course; that is, $n_i = 1$ for C_i.

3. The Scheduling Algorithm

The algorithm to be presented was developed with some very specific objectives to be satisfied. It was first decided that students were to be given the opportunity to request the time and instructor that they wanted. This immediately results in the requirement that section requests will be made as opposed to course requests. Assuming that resources (such as classrooms and staff) are limited, there may be many courses $C_i \in C$ for which $D_i > T_i$. It is therefore a requirement for the algorithm to have a section balancing feature [4, 5].

Later, in Section 4, we will describe some of the means by which the resource assignments can be altered so that a greater number of students can receive complete schedules when the condition $D_i > T_i$ is satisfied for many C_i in C.

Phase I of the Algorithm

1. $\alpha = \alpha'$, $a = a'$ (α' and a' are fixed values for α and a. For the results in Table I, $\alpha' = 0.8$, $a' = 300$.)
2. Obtain a request set W.
3. To each request w_i in W obtain all sections c_{ij} of course C_i in the timetable C.
4. Tag as an invalid request any w_i for which there is not an exact match c_{ij} in C.
5. Any section c_{ij} for which $r_{ij} > \alpha t_{ij}$ is dropped.
6. Considering all sections $c_{i1}, c_{i2}, \cdots, c_{in_i}$ of course C_i there may be multiple sections with the same time string e^k. Denote these multiple sections as having unique time strings $e^{k1}, e^{k2}, \cdots, e^{kt}$ where $t \leq n_i$. Obtain c^*_{ij} associated with $\min_{i_1}, \cdots, i_t\{r_{ii_1}, r_{ii_2} \cdots, r_{ii_t}\}$.
7. There is one exception to Step 6. For any section c_{ij} corresponding to request w_i, for which the associated $r_{ij} > \min\{r_{ii_1}, r_{ii_2}, \cdots, r_{ii_t}\}$ the exact original request c_{ij} is retained.

Note. As a result of Steps 4–7 there is at most one section for any given course for each unique time string number e^k.

8. Attempt to schedule the student in exactly the sections he requested, the $\{c_{1j_1}, c_{2j_2}, \cdots, c_{qj_q}\}$ corresponding to the $\{w_1, w_2, \cdots, w_q\}$, or else schedule him in a section with the same time string e_k. That is, associated with each c_{ij_i} is the corresponding r_{ij_i}, and a section c'_{ij} is chosen with the same time string $e^k_{ij_i}$ for which $r'_{ij} < r_{ij_i}$.

If the student is enrolled at this time he has received exactly the times that were requested.

9. Any courses not scheduled as requested are scheduled next.
   Course in request: $C_1, C_2, C_3, \cdots, C_{q1}$.
   Scheduling difficulty code: $p_1, p_2, p_3, \cdots, p_{q1}$.

The courses are ordered by scheduling difficulty from most difficult to easiest, $p_{i_1} \leq p_{i_2} \leq \cdots \leq p_{i_{q_1}}$. An attempt is first made to schedule the yet unscheduled course, C^*, with the smallest percent of its capacity filled, that is, the C^* associated with $\min_{\{1,2,\cdots,q1\}} \{R_1', R_2', \cdots, R_{q1}'\}$.

10. If C^* will not fit, remove C^* from $\{C_1, C_2, \cdots, C_{q1}\}$. Denote $C^* = c^{(0)}$ and then try $c^{(1)}$ associated with $\min_j \{R_1', R_2' \cdots, R_{q-1}'\}$. Continue until all sections have been tried or the section is scheduled.
11. If no section will fit in the students schedule this course is tagged as being unscheduled. The second most difficult course is attempted by the same process. If need be, all unscheduled courses will be tried.
12. If a complete schedule results then we proceed to the next student. If an incomplete schedule results, then we enter Phase II.

Phase II of the Algorithm

1. We record the schedule W_s that has been given to the student thus far from Phase I: $W_s = \{w_1, w_2, \cdots, w_{q_1}\}$.
2. The courses yet unscheduled are $W_u = \{w_{q_1+1}, \cdots, w_{q_1+q_2}\}$. The set W_s is ordered by difficulty code p_i from easiest to hardest. That is, the $w_{i_1}, \cdots, w_{i_q}$ for which the associated $p_{i_1} \geq \cdots \geq p_{i_{q_1}}$. The set W_u is ordered from hardest to easiest, that is, $w_{j_1}, w_{j_2}, \cdots, w_{j_{q_2}}$ for which the associated $p_{j_1} \leq p_{j_2} \leq \cdots \leq p_{j_{q_2}}$. We know that for the first unscheduled course, w_{j_1}, no section of this course will fit because of Phase I.
3. We remove w_{j_1} from the schedule and try all sections of w_{j_1} in order to place it in the schedule. If no section of w_{j_1} fits, we place w_{i_1} back into the schedule.
4. We continue until all scheduled sections $\{w_{i_1}, w_{i_2}, \cdots, w_{i_{q_1}}\}$ have been tried. If, during the procedure, a section w_j from W_u will fit in the schedule, all sections of the removed scheduled course w_i are tried. When the attempts result in failure, the scheduled courses w_i tried are tagged as unusable combination.
5. After enumerating all w_i, a check is made to see if any courses remain unscheduled.
6. With unscheduled courses still remaining, steps 3 and 4 are repeated with the modification that two scheduled courses are removed one at a time.
7. If need be, steps 3 and 4 are be repeated with three, four, etc., scheduled courses removed until all possible combinations have been tried. If a complete schedule has not been found at this point, a complete schedule does not exist for the given timetable C. The student is given the best incomplete schedule to date by the process.

Notice that in the Phase I of the algorithm Steps 1–3 constitute initialization of the algorithm. Steps 4–7 may be considered editing or bookkeeping steps. Step 8 is the first attempt at a schedule for the student. Steps 9–12 involve further attempts at finding a complete schedule subject to the criterion of section balancing. In actual practice, most students would be scheduled in Phase I of the algorithm. It is not until a great majority of the students have been scheduled and the section availabilities q_{ij} become scarce that Phase II of the algorithm is entered.

Notice in Phase II that the following criterion is adhered to. When a student is to receive an incomplete schedule, the partial schedule that he does receive should be as close to his original request set W as possible.

4. Strategies in Altering the Timetable C

For a fixed timetable C, it is apparent that the results from a scheduling pass may not be satisfactory. In any event, it is necessary to be able to evaluate the results of a scheduling pass. If the results are deemed unacceptable then another pass should be made. A new scheduling pass can then be made with a new timetable C'. The procedure then for improving scheduling results can be described as a feedback process where, from observing the results of a pass, decisions can be made for altering the resources of the timetable and a new pass can be made.

What then is the means for evaluating the timetable C in order to determine where resources should be altered or increased? In C associated with each section j of course i, c_{ij} will be the following accumulated data attributes. We will have computed the demand d_{ij} for each section along with the enrollments r_{ij}. In the predetermined timetable, the available seats t_{ij} for each section are already known. We therefore have, for each c_{ij}, $c_{ij} : t_{ij}, d_{ij}, r_{ij}$. We can also sum over all sections for each course to get $C_i : T_i, D_i, R_i$. Another way to look at this is to consider the scheduling input sequence $s_1, s_2, \cdots, s_m$ and, if we made a pass through all m students, for a section to be adequate to satisfy the student demand for it,

$$\sum_{k=1}^{m} u_{ijk} \leq t_{ij}. \tag{1}$$

The inequality (1) implies that the total number of students scheduled in section j of course i is less than or equal to the seats made available in timetable C for c_{ij}.

For the course C_i to be adequate, if we find

$$L_i = \sum_{j=1}^{n_i} t_{ij}, \tag{2}$$

then we must also have

$$\sum_{k=1}^{m} \sum_{j=1}^{n_i} u_{ijk} \leq L_i. \tag{3}$$

That is, there are enough seats available for course i in C.

Another valuable quantity is the number of seats remaining, q_{ij}, in a section of a course after a pass. We see that

$$q_{ij} = t_{ij} - \sum_{k=1}^{m} u_{ijk}. \tag{4}$$

Assuming that no more students are added to set S after any pass using the scheduling algorithm, we see that the demand d_{ij} on any pass is

$$d_{ij} = \sum_{k=1}^{m} v_{ijk}. \tag{5}$$

There are four cases to consider in possible strategies for altering the timetable C.

Case 1. There are not enough sections of a course C_i offered in C to accommodate the demand for the course. In order to detect this condition we can inspect course C_i to see if

$$D_i > T_i . \tag{6}$$

If (6) is the case and, further, if

$$D_i - T_i > (1/n_i)T_i , \tag{7}$$

then as a general rule it would be advisable to add one more section for each $Q_i = (1/n_i)T_i$ seats that are unavailable.

Case 2. There are too many sections of a course C_i offered with too few students in some or all of the sections. This condition arises when

$$D_i \ll T_i . \tag{8}$$

A rule of thumb for the number of sections to drop in the case of (8) is the following. If

$$T_i - D_i < (2/n_i)T_i , \tag{9}$$

then drop one section for each $(2/n_i)T_i$ additional seats that are available.

Case 3. There are not enough seats available for a section of course i. (In most instances this will be a single section course.) We have in this case

$$d_{ij} > t_{ij} . \tag{10}$$

To satisfy the deficiencies of this condition it is advisable to add $d_{ij} - t_{ij}$ seats to this section provided $d_{ij} - t_{ij} < t_{ij}$. If $d_{ij} - t_{ij} > t_{ij}$, we can resort to case 1.

Case 4. There is insufficient demand for a section j of course C_i to warrant that the section be taught. In this case,

$$d_{ij} \ll t_{ij} .$$

In fact, when $t_{ij} - d_{ij} < \frac{1}{2}t_{ij}$, consideration should be given to whether it is feasible to offer the section of this course.

5. An Example

To illustrate some of the mechanics involved in the algorithm, the following example has been constructed. The first part of the example illustrates a schedule generated as requested. The second part illustrates a complete schedule but one with only two sections remaining as requested.

Assume that student, s_{35}, is to be scheduled. Since s_{35} can also be assumed to be less than s_0, the student is a graduate student at the beginning of the scheduling sequence $s_1, \cdots, s_f$. For the first pass through the algorithm, we will further assume that each section of each course, c_{ij}, will be allowed to fill only to 80 percent of capacity. Therefore, in Step 1 $\alpha = 0.8$ and $a = 300$ iterations will be allowed in letting the algorithm cycle in its procedures for finding a complete schedule. In Step 2 the request set W is obtained for s_{35}. We further assume that

$$W = \{w_1, w_2, w_3, w_4\} = \{c_{14,2}, c_{182,5}, c_{1081,3}, c_{53,1}\}$$

are the requests made by student s_{35} from timetable C. Step 3 requires that all sections of C_{14}, C_{182}, C_{1081}, C_{53} be obtained. Say that the following n_i are associated with each course C_i : $C_{14}, n_{14} = 3$; $C_{182}, n_{182} = 5$; $C_{1081}, n_{1081} = 3$; $C_{53}, n_{53} = 1$. All the w_i have a corresponding valid c_{ij} ; therefore, Step 4 requires no tagging.

The following illustrates the condition of the timetable for courses C_{14}, C_{1081}, C_{182}, C_{53} at the time s_{35} is being scheduled.

i	j	c_{ij}	r_{ij}	αt_{ij}	t_{ij}	e^k_j	$(x_1, \cdots, x_{16})$
14	1	$c_{14,1}$	3	32	40	$e^{6}_{14,1}$	1010000001000000
	2	$c_{14,2}$	1	32	40	$e^{7}_{14,2}$	0101000001000000
	3	$c_{14,3}$	1	32	40	$e^{9}_{14,3}$	0000110010000000
182	1	$c_{182,1}$	2	12	15	$e^{12}_{182,1}$	1010101000000000
	2	$c_{182,2}$	1	12	15	$e^{12}_{182,2}$	1010101000000000
	3	$c_{182,3}$	4	12	15	$e^{13}_{182,3}$	0101011000000000
	4	$c_{182,4}$	2	12	15	$e^{14}_{182,4}$	0101010100000000
	5	$c_{182,5}$	10	12	15	$e^{16}_{182,5}$	1010100100000000
1081	1	$c_{1081,1}$	0	16	20	$e^{32}_{1081,1}$	0000100000011100
	2	$c_{1081,2}$	1	16	20	$e^{34}_{1081,2}$	0010000000011100
	3	$c_{1081,3}$	1	16	20	$e^{36}_{1081,3}$	0010100001100000
53	1	$c_{53,1}$	5	8	10	$e^{82}_{53,1}$	1110000000011000

In Step 5, all sections are retained since for each section in the request set all $r_{ij} \leq 0.8t_{ij}$. In Step 6, as a matter of observation, $c_{182,1}$ and $c_{182,2}$ have the same time string e^{12}. Further, for course C_{14}, min $\{r_{14,1}, r_{14,2}, r_{14,3}\}$ = min $\{3, 1, 1\} = 1$, with the associated section being $c_{14,2}$. For C_{182}, min $\{r_{182,1}, r_{182,2}, r_{182,3}, r_{182,4}, r_{182,5}\}$ = min $\{2, 1, 4, 2, 10\} = 1$ and the associated section is $c_{182,2}$. For C_{1081} and C_{53}, min $\{r_{1081,1}, r_{1081,2}, r_{1081,3}\}$ = min $\{0, 0, 1\} = 0$ and min $\{r_{53,1}\}$ = min $\{5\} = 5$ with the associated sections being $c_{1081,1}$ (or $c_{1081,2}$) and $c_{53,1}$.

In Step 7 note that for courses c_{182} and c_{1081} the original requests $c_{182,5}$ and $c_{1081,3}$ are retained even though sections $c_{182,2}$ and $c_{1081,1}$ (or $c_{1081,2}$) are less filled than the original request. This illustrates the fact that section balancing does not occur until a section j of course i is filled in excess of αt_{ij}.

The schedule generated for s_{35} was as requested. In this particular case, the algorithm updated the timetable totals for r_{ij} and Steps 1–8 cleared s_{35} for a schedule exactly as requested. As can be seen from Table I, on the first pass 71.79 percent of the sections requested were granted. This translates into 35 percent of the student body receiving schedules exactly as requested. In terms of cycling through the algorithm, this means that over one third of the schedules for the entire student body were generated by cycling through Steps 1–8 of the algorithm,

Next, let us modify this example to illustrate the case of a schedule that is generated by cycling through Steps 1–12 of Phase I. Assume first that this student is also a graduate student, but that he is down the sequenced input stream to position 1321—that is, s_{1321}.

The request set of this student is $W = \{w_1, w_2, w_3, w_4\} = \{c_{14,2}, c_{182,2}, c_{1081,3}, c_{53,1}\}$. It is similar to s_{35}'s request set except for course C_{182}. By this point in the sequence assume the condition of these courses in the timetable is as follows:

i	j	c_{ij}	r_{ij}	αt_{ij}	t_{ij}
14	1	$c_{14,1}$	30	32	40
	2	$c_{14,2}$	28	32	40
	3	$c_{14,3}$	34	32	40
182	1	$c_{182,1}$	3	12	15
	2	$c_{182,2}$	13	12	15
	3	$c_{182,3}$	15	12	15
	4	$c_{182,4}$	12	12	15
	5	$c_{182,5}$	13	12	15
1081	1	$c_{1081,1}$	12	16	20
	2	$c_{1081,2}$	13	16	20
	3	$c_{1081,3}$	17	16	20
53	1	$c_{53,1}$	7	8	10

Tracing the algorithm through Steps 1–8, we see that $c_{14,2}$ and $c_{53,1}$ are granted as requested. $c_{182,1}$ is granted, instead of $c_{182,2}$ since it has the same time string, and $c_{182,2}$ has $r_{182,2} > 0.8t_{182,2}$.

It requires cycling through Steps 9–12 to see that $c_{1081,1}$ is scheduled. In Step 10, $C^* = C_{1081}$, and since $c_{1081,1}$ is the section least filled ($r_{1081,1} = 12$), it is included in the schedule. The schedule is complete for s_{1321} and is $\{c_{14,2}, c_{182,1}, c_{1081,1}, c_{53,1}\}$.

6. Results from the First Implementation of the System

In Table I the results from the first implementation of the system are presented. Notice that there are aggregate figures which display the number and percent of complete schedules. One can see from the algorithm in Section 3 that a "complete schedule" means the following. If a student receives all courses in his original request set W, he has a complete schedule. An "exact schedule" is one for which all classes granted are at the same time as the classes in the request set W.

There are two ways in which to evaluate the results of a scheduling pass. The first is to look at the total number of students with complete schedules. A large number of incomplete schedules leads to time-consuming clerical follow-up procedures termed "drop and add" procedures [2]. A second way to evaluate the overall results is to look at the total number of sections granted compared to the total number of sections requested.

Greater emphasis should be placed with the number of students with complete schedules. It is at least the attempt of each large publicly supported institution to furnish each and every student with the courses he needs for any given term.

7. Relationship of the Scheduling Algorithm to Other Approaches

There is one key policy decision that must be made at the beginning before using an already available algorithm or implementing one's own. That decision is

Table I. Scheduling Results for Winter Term 1969

Pass number	1	2	3
Number of students scheduled	16147	16293	16346
Number of students with complete schedules	10762	12304	12606
Percent of students with complete schedules	66.6	75.4	77.2
Number of students that received exact schedules	5650	6019	6020
Percent of students that received exact schedules	35.0	36.9	36.8
Number of students with partial schedules due to closed section	4323	3047	2804
Percent of students with partial schedules due to closed section	26.8	18.7	17.11
Number of students with partial schedules due to conflicts	1040	906	915
Percent of students with partial schedules due to conflicts	6.46	5.58	5.62
Number of students with partial schedules due to excessive iterations	22	36	21
Percent of students with partial schedules due to excessive iterations	0.14	0.22	0.13
Number of students with complete schedules but no lunch	15	17	15
Percent of students with complete schedules but no lunch	0.09	0.10	0.09
Number of sections requested	88111	88969	89355
Number of sections granted as requested	63259	64266	64621
Percent of sections granted as requested	71.79	72.23	72.32
Number of sections granted at requested time	65312	66968	67446
Percent of sections granted at requested time	74.12	75.27	75.48
Number of minutes to run	293.12	286.58	291.93
Number of schedules per minute	55	57	56
Number of minutes per schedule	0.0182	0.0175	0.0178
Number of sections considered	88111	88969	89355
Number of sections granted	80110	83800	84702
Percent of sections granted	90.9	94.2	94.8

whether a student body shall be allowed to make course requests or to make section requests. This seemingly unimportant decision decides both the technical details involved in the scheduling process and the administrative manner in which the registration process will be conducted. It is hard to compare algorithms where one algorithm accepts course requests as input and the other accepts section requests as input. One reason for this is the fact that the algorithm that accepts course requests must develop a complete schedule for a student independent of any knowledge about the type of schedule the individual student may want. One example of this type of scheduling algorithm is the one used at Purdue University as described by Abell [6]. This means that this type of algorithm is designed in such a manner as to section every course in the student's request set of courses. On the other hand, the algorithm that accepts section requests allows the student a preference of both instructor and time which means he may express the type of schedule he desires. In terms of the technical mechanics of the algorithm this means that section balancing can only begin after a section has filled to αt_{ij} of its capacity t_{ij}. The way in which sections of courses are allowed to fill is different for this type of algorithm than for course request algorithms.

For section preference algorithms it is the logical procedure that occurs after the courses have filled to a prespecified level that provides the differences in the technical details. In [4, 5], the authors have proposed means for balancing sections of the same course. The main difference between the algorithm presented here and that one is the fact that our algorithm deals with the question of incomplete schedules as an acceptable means of allocating all resourses available in the best possible manner. It was assumed when this algorithm was first implemented with a live student body that incomplete schedules were acceptable since a follow-up administrative manual procedure would handle any special situations.

The results presented in Table I are live results for an actual institution. All other formally published algorithms [4, 5] have been only experimentally produced (nonlive) data. Without comparisons using the same data, it is unrealistic to compare the results of the algorithm in this way.

The results in Table I should be evaluated on their own merit in the following way. First, in the case of this algorithm, specific measures such as the percent of complete schedules received and the percent of sections granted given knowledge of the number of sections requested are two criteria that can be prespecified by the administration as goals to be met for a particular term. In combination with the feedback procedure described in Section 4, it is possible to modify the resource mix in the timetable in order to attain these goals.

As opposed to previous manual systems, the use of this algorithm provides an objective way in which criteria can be set to achieve prespecified goals for a coming term's registration and enrollment. Implementing this algorithm term after term can provide a historical record of the required proper mix of resources in the timetable and a historical record of results in producing schedules. Successive improvements in timetable construction and in the scheduling process can lead to acceptable results by the administration and student body alike. The real task of the administration is to provide in a timetable of classes the resources of the institution in such a way as to provide the classes needed by all students in the student body. The real desire of the student is to obtain a schedule so that he can attend and receive credit for the classes he needs to complete his program of study.

Received October 1970

References

1. Winters, W. K. Implementation of a computer assisted registration system. Proc. of the Eighth Annual Southeastern Regional Meeting of the ACM, Huntsville, Ala., June 12–14, 1969, pp. 1–9.
2. ——. The plan and method for implementing computer assisted registration at the University of Tennessee. Rep. 2, U. of Tennessee Computer Assisted Registration System, Oct. 1, 1968, pp. 1–55.
3. ——. An investigation of several algorithms for solving the timetable problem. Rep. 6, U. of Tennessee Computer Assisted Registration System, Knoxville, Tenn. July 8, 1969, pp. 1–48.
4. Macon, N., and Walker, E. E. A Monte Carlo algorithm for assigning students to classes. *Comm. ACM 9*, 5 (May, 1966), 339–340.
5. Busam, V. A. An algorithm for class scheduling with section preference. *Comm. ACM* 10, 9 (Sept. 1967), 567–569.
6. Abell, V. A. Purdue academic student scheduling. Mimeographed, 1965, Purdue U., Lafayette, Ind.