On Academic Resource Scheduling

Kwang Min Yang*

CONTENTS

Ⅰ. INTRODUCTION

Ⅰ.1 The Problem

It is assumed throughout this study that, prior to considering the scheduling of classes, the institution has decided what courses will be offered, the number of sections if more than one, and has assigned faculty members to teach these (if not assigned, create a fictitious faculty member who teaches a single course for each unassigned course).

Further assumptions are that a course is taught by a single faculty member, a course uses only one facility, exhibits fixed time patterns throughout the week and that classrooms can neither be partitioned into smaller rooms nor combined together. The problem area is delimited briefly.

There will be two main types of schedules: A cyclic plan which is repeated after a given time frame, and a non-cyclic plan which is used only once. An ordinary school schedule is cyclic, whereas an examination plan, for example, may be non-cyclic. The length of the time slot is usually given beforehand and it will be considered fixed.

It may be assumed as an axiom that no solution exists which satisfies all requirements for an optimal schedule, A schedule is therefore a compromise solution between a large number of conflicts. These conflicts are of a hierarchical nature; i.e., all requirements are not equally strict, One may clearly distinguish between a set of absolute requirements ('hard' constraints) and a set of desirable requirements ('soft' constraints). An absolute requirement is a requirement which has to be satisfied, and a desirable requirement is a requirement which ought to be satisfied.

For the main kinds of scheduling requirements, see Michalsen[41].

* Associate Professor, Chung Ang University.

I. 2 Brief Survey of Earlier Methods

One of the earliest pieces in the literature on this problem is Bowden's work [10]. He explained how to schedule classrooms using conflict matrices.

With the advent of computers, many papers [1, 2, 3, 4, 6, 9, 12, 13, 14, 16, 18, 21, 25, 30, 38, 39, 46, 53, 56, 57, 58, 59] which deal with school scheduling problems emerged in the sixties with the hope that the application of digital computers for the problem might offer substantial advantages over manual scheduling. However, early efforts to realize these advantages, wherein the human was replaced by a computer without any fundamental change of procedure, proved not to be very satisfactory. (It appears that as with many computer applications, the difficulty of the problem has been established instead.) For example, the Generalized Academic Simulation program (GASP) was developed at MIT in the early sixties. Meanwhile Stanford University developed a computer program commonly known as the Quad-S (Stanford School Scheduling System [14]). One of the application reports on Quad-S appears in Wiley [56]. These programs are both heuristic and have been tested at several high schools. The results were rather inconclusive, and some authors even presented opinions on the superiority of manual schedules to computer-generated schedules [51].

In the seventies the problem has been approached by analytical methods such as graph theoretic approaches [31, 53, 54, 55], Boolean methods [46, 49] and mathematical programming [42, 50, 52]. Of course more elaborate and sophisticated heuristic approaches were added, as described by Michalsen [41], for instance.

The problem, which is better known as the time table problem, can be reduced to finding the chromatic number of a hypergraph, the strong chromatic number [7], which is the minimum number of colors required to color the vertices of the hypergraph so that no color appears twice on the same edge [53, 58]. However the computation algorithm [13, 17, 48] for finding the exact value of a graph usually becomes impractical as the number of vertices increases. Computational results available in the literature still are not very encouraging. Knauer [31] reported that the computation time of his graph theoretic approach was between 1.5 and 2.0 minutes per class on Electrologica EL X8 (Which is about as fast as the IBM 360/40).

The problem can no longer be represented by a coloring problem if we introduce a facility capacity restriction such that any classroom, whose seating capacity is large eno-

ugh for a course, can be assigned to the course. This is the one we well develop in this paper. Swart [52] reported that his mixed integer programming (MIP) approach took 4 to 26 minutes for 115 variables and 135 constraints problems (approximately equivalent to a 12—14 course scheduling problem) on a Burroughs B—5500. Liggett [37] also applied a quadratic assignment approach to the problem.

II. MODELS

There have been a number of model formulations of the problem of constructing good class schedules. The models can be classified into pseudo-Boolean equations [46, 49], integer linear programming (ILP) formulations [36, 42, 46, 50, 52], quadratic assignment formulations [37], and graph theoretic models [31, 53, 58]. Since Boolean equations and graph theoretic models can be transformed into an equivalent ILP problem [46], we describe only ILP and quadratic assignment models here. Quadratic assignment problems can also be formulated as ILP problems [35]. The model developed here can be subsumed as a part of an integrated academic information and planning system such as the one described by Dyer & Mulvey [20] and Dyer [19].

Ⅱ.1 ILP Model

The classroom scheduling problem is one of the various combinatorial optimization problems that can be formulated as an ILP. Various ILP models [36, 42, 46, 50, 52] have been proposed.

Consider that, for the planning period, a certain number of courses (multi-section courses are treated as different courses) are to be scheduled to time slots. Define binary variables such that $x_{ij}=1$ if course j is scheduled to time slot i and $x_{ij}=0$ otherwise. The constraints defining the set of permissible schedules fall into six types; these are, course assignment constraints, facility constraints, faculty time conflict, student scheduling conflict and two other structural constraint types for continuity requirements for multi-periods courses. The objective can be expressed as the function of maximizing (minimizing) the aggregate preference (disutility) of faculty and students.

Thus the detailed ILP model is

$$\text{Maximize} \quad \sum_i \sum_j p_{ij} x_{ij}$$

$$x_{ij}, y_{i'j} \in \{0, 1\}$$

$$\text{subject to} \quad \sum_i x_{ij} = t_j \qquad \forall\, j \tag{1}$$

$$\sum_{j \in R_l} x_{ij} \leq c_{il} \qquad \forall\, i, l \tag{2}$$

$$\sum_{j \in F_r} x_{ij} \leq 1 \qquad \forall\, i, r \tag{3}$$

$$\sum_{j \in S_u} x_{ij} \leq 1 \qquad \forall\, i, u \tag{4}$$

$$\sum_{i=i'}^{i'+t_j-1} x_{ij} \geq t_j y_{i'j} \qquad \forall\, i' \in I_j = \{i \mid 1, 2, \cdots, (|I| - t_j + 1)\}, j \tag{5}$$

$$\sum_{i' \in I_j} y_{i'j} = 1 \qquad \forall\, j \tag{6}$$

where p_{ij} denotes the aggregate preference when course j is assigned to time slot i. The preference may be an individual faculty member's preference for a particular time slot or a student group's average preference for a particular time or both. The purpose of equations (1) is to assure that each course is assigned to exactly t_j time slots. The classroom seating capacities are enforced by constraints set (2). The index set R_1 identifies those courses which have facility (classroom or lab) size group 1. c_{il} is the number of classrooms of size group 1 available at time slot i. The second type of constraint is cumulative in nature, i.e., $c_{il} < c_{i,l+1}$ and $R_l \subset R_{l+1}$; hence grouping 1 is assumed to be inorder. In other words a large capacity classroom can be used for any course whose enrollment is less than or equal to its capacity. The constraints (3) preclude courses to be taught by a faculty member from being assigned to the same time slot i.e., faculty conflict. The index set F_r identifies those courses which are taught by faculty member r. In a similar fashion, inequality (4) restricts certain courses from meeting at the same time because of student scheduling conflicts. S_u is the set of courses which must be taken by the student group u. Any other constraints which serve to indicate, that for one reason or another, certain courses should not be offered at the same time may be included in this constraint set. The constraints (5) and (6) are to assure that the time slots are assigned consecutively for a particular course. t_j denotes the number of consecutive lecture periods required. $y_{i'j}$ is a binary structural variable and $|I|$ denotes cardinality of the index set I.

Note that if it is assumed that all even time slots are used, the foregoing model is simplified as

$$\text{maximize} \quad \sum_i \sum_j p_{ij} x_{ij}$$

$$x_{ij} \in \{0, 1\}$$

$$\text{(p)}\quad \text{subject to}\quad \sum_i x_{ij}=1 \qquad \forall\, j \qquad (1)$$

$$\sum_{j\in R_l} x_{ij}\leq c_{il} \qquad \forall\, i,l \qquad (2)$$

$$\sum_{j\in F_r} x_{ij}\leq 1 \qquad \forall\, i,r \qquad (3)$$

$$\sum_{j\in S_u} x_{ij}\leq 1 \qquad \forall\, i,u \qquad (4)$$

The assumption of even time slots gives us a tremendous computational advantage over the original formulation both in the number of constraints and in the number of variables that must be dealt with.

Ⅱ.2 Quadratic Assignment Model

The classroom scheduling problem can be stated as follows (again identical time slots are assumed).

$$\underset{x_{ik}\in\{0,1\}}{\text{Minimize}}\quad \sum_i\sum_k a_{ik}x_{ik}+\sum_i\sum_j\sum_k\sum_l Q_{ij}C_{kl}x_{ik}x_{jl}$$

$$\text{subject to}\ \sum_k x_{ik}=1 \qquad \forall\, i$$

The above is the same as the Koopmans-Beckmann objective function [33], with an addition of linear terms.

The subscripts, i and j, denote courses, and k and l denote classroom timeslots. The coefficients are:

$$Q_{ij}=\begin{cases}\infty & \text{if a conflict exists (either professor or student conflict) between course } i \text{ and } j.\\ 0 & \text{otherwise.}\end{cases}$$

$$C_{kl}=\begin{cases}1 & \text{if timeslots are same time.}\\ 0 & \text{otherwise.}\end{cases}$$

$$a_{ik}=\begin{cases}p_{ik} & \text{if course } i \text{ is assignable to room-timeslot } k\\ \infty & \text{if cannot be assigned.}\end{cases}$$

If there are m courses to be scheduled and n classroom-timeslots ($m\leq n$) then Q is an m x m, c is an n x n and (a_{ik}) is an m x n matrix. p_{ik} is the preference when course i is assigned to timeslot k. We have assumed here that all courses require identical time slots. For any feasible solutions, the quadratic terms will be vacuous.

Ⅲ. SOLUTION APPROACHES

Ⅲ.1 Problem Characteristics

A real classroom scheduling problem may contain several hundred course offerings and several hundred time slots which may yield a problem having several tens of thousands of binary variables. The problem formulated in section Ⅱ can theoretically be solved by an ILP algorithm; however, the current state-of-the-art does not render a practical solution within a reasonable amount of solution time.

In an attempt to analyze the problem more carefully so as to benefit from the structure of the problem in designing an efficient solution method, some of the problem features are noted:

a) Size of the problem: For 150 courses, 100 professors, 20 student-major groups, 5 classroom groups. and 10 time slots problem (i. e. the case of the Graduate School of Management (GSM, UCLA), the size of the problem matrix is approximately 1,200×1,300 with all 0-1 variables, which is well beyond the current integer programming capability-computer storage for the problem matrix alone requires 1.6 million words if a full matrix version is adopted.

b) problem Matrix: The problem matrix consists of either zeros or ones.

c) Sparseness: The density of the problem matrix is very low (in the GSM case, $7,350/(1,200\times1,300)=47\%$). This extreme sparseness of the problem must be taken into account in designing an algorithm.

d) Degeneracy: Computational experiences indicate that LP-based approaches to this type of problem have encountered difficulty in solving the associated LP problems due to massively degenerate solutions [40].

e) 'Multiple Choice' Constraints: Every course must be assigned to a particular time slot. Those constraints constitute disjoint and mutually exclusive generalized upper bound (GUB) type constraints.

Due to the aforementioned problem characteristics, it appears to be inevitable to resort to either a network type of algorithm for efficiency both in storage and computing time, or a factorization approach [26] if the problem is ever to be solvable by an analytical method within a reasonable amount of solution time.

Ⅲ. 2 Early Attempts

Ⅲ. 2-1 Mulvey's Approach

Mulvey suggested a heuristic solution method for the classroom scheduling problem. The general strategy is summarized as follows [42, 43]:

Step 1. Generate the data required for the approximate model. This data includes faculty information and course information, both modified to reflect the administrative policies.

Step 2. (Approximation) Generate the network model, and solve.

Step 3. (Evaluation) Determine whether the candidate schedule is a feasible alternative. If not, return to Step 1.

Step 4. (Evaluation) Determine whether the candidate schedule is acceptable. If not, go to Step 6.

Step 5. Print the schedule.

Step 6. Decide whether to persist in attempting to improve the candidate schedule by hand. If not, go to Step 8.

Step 7. (Modification) Make manual changes in the candidate schedule. Go to Step3.

Step 8. (Modification) Make changes in the network formulation. Go to Step 3.

The rationale behind this approach is that the problem is often ill-defined. Many constraints are not quantifiable and there may be many different criteria by which the solution may be evaluated. However, what Mulvey suggested in Step 2 (approximation) appears to be an oversimplification because it hardly ever produces a feasible solution unless the problem is trivial. The underlying idea is to try to take advantage of an efficient network code; however, only a small portion of the problem consists of network transformable constraints. Also Mulvey failed to suggest a sound (proven) criterion as to how to modify the candidate schedule, which is crucial to the efficiency of the method.

Ⅲ. 2-2 Swart's Approach

Swart [52] suggested a solution algorithm for (p) which takes advantage of the unimodular property of the imbedded constraint sets. He indicated that, although the entire problem could not apparently be expressed as a network, it was true that a substantial number of the constraints could be interpreted as a network type problem, while the

remainder of the constraints could be treated as additional restrictions on the flows in the network.

Suggested are three different ways in which to define the imbedded network in the class scheduling model. These are:

i) constraint set (1) and part of constraint set (2)

ii) constraint set (1) and constraint set (2)

iii) constraint set (1) and constraint set (3)

However, Swart failed to recoginize that 1) partitioning scheme i) and ii) can be simplified in terms of the number of arcs by introducing the slack and surplus variables to constraint set (2), and 2) most importantly, constraints (1), (2) and (3) together possess the unimodular property (although they are not a network).

The computational results were also presented. Unfortunately they were not of practical value primarily due to the inefficient network code (out-of-kilter) for relatively large problems, no availability of a tighter bound (network relaxation tends to be too loose), and no efficient selection of the branching variables.

Ⅲ. 2—3 Quadratic Assignment Method

The classroom scheduling problem can be represented by an assignment problem with an introduction of quadratic terms for handling relationships between variables such as the one shown in Section Ⅱ.

Inasmuch as the traveling salesman's problem, and other hard problems, can be formulated as special cases of the quadratic assignment problem, it is not surprising that no efficient procedure has been found to obtain exact solutions. In fact, it appears to be safe to say that little real progress has been made in the ten years or so since the problem began to receive widespread attention. Among the approaches which have been attempted are the following [35]:

a) Branch-and-bound methods, e.g., Graves and Whinston [27], Gilmore [24], Lawler [34], Hillier and Conners [35].

b) Formulation of equivalent integer linear programming problems, e.g., Lawler [34]. Actual computational experience with this approach is not available.

c) Obtaining an exact solution to a related continuous problem and then discretizing. See Kodres [32].

d) Heuristic procedures, e. g., pairwise interchanges of assignments. Heuristics have been tested by many investigators, with varying degrees of success. For an excellent survey, see Hanan and Kurtzberg [28].

Liggett [37] formulated the problem as a quadratic assignment problem and implemented it at the School of Architecture and Urban Planning, UCLA. The solution algorithm used was by Graves and Whinston [27], and the solutions were obtained within a couple of minutes. However the problem size was relatively small and not so tightly constrained as in the case of GSM.

Ⅲ. 3 Proposed Approaches

As discussed in Ⅲ. 1 the problem apparently has a special structure, namely, sparseness, an all 0-1 coefficient matrix, GUB constraints (constraint set (1)), identity matrices, and a right-hand side (RHS) of 1's except for the constraint set (2).

The problem also possesses some latent features such as total unimodularity of constraint sets (1), (2) , and (3) (which will be proved later in this chapter) and other properties which help to generate integer solutions (see Section Ⅲ. 3. 2).

Proposed are: 1) a network-based branch-and-bound-based on the property of having GUB constraints and identity matrices, and 2) an LP-based approach-based on the properties of total unimodularity of most of the constraint matrix, a 'likely integer optimal solution generating' problem matrix (due to GUB constraints and identity matrix) and the fact that most of the RHS coefficients are 1.

Ⅲ. 3. 1 Network-based Branch and Bound Approach

The basic idea behind this approach is two-fold. We observe that the constraints in the model(P) developed in the previous chapter are not coupled by subscript i except in constrain tset (1). Hence if we relax the (P) by dropping constraint type (1), the resulting relaxed problem separates into a series of small problems which can be handled efficiently and can be viewed as network problems (it will be shown that constraints (2) and (3) constitute networks) with side constraints, i. e., constaints (4). Conceptually, this corresponds to solv ing the (1), (2) and (3) constraints while ignoring (4), which may be satisfied, if violated, during the course of the branch and bound by imposing additional restrictions in the form of fixing the variables. Hence, it can also be viewed as a multi-commodity flow network

problem, Computational experience shows that the resulting problems are quite often pure networks, if the original problem is not overly constrained, to which an extremely efficient algorithm can be applied such as GNET [11].

We explore some of the special sturctures of the problem. It is easy to see constraint sets (2) and (3) are totally unimodular respectively. Due to Heller and Tompkins [29], the constraint set (2) in (P) is totally unimodular since $R_l \subset R_{l+1}\ \forall\ l$ (a small l represents large size classrooms). The constraint set (3) contains only one entry, i.e. $+1$ per column, therefore constraint set (3) itself is obviously totally unimodular.

Now we show that the constraints (2) and (3) constitute networks. The constraint set (2) can be transformed into a matrix with one entry per column (for slack and surplus variables, at most two entries per column) by introducing slack and surplus variables.

Since $R_l \subset R_{l+1}\ \ \forall\ l$ and

$$c_{il} < c_{i,l+1}\ \ \forall\ i,\ l \in \{1, 2, \cdots, l'\},$$

then, $$\sum_{j \in R_l} x_{ij} \leq c_{il}\ \ \forall\ i,\ l$$

implies $$\sum_{j \in R'_l} x_{ij} + s_{il} - s_{i,l+1} = c'_{il}\ \ \forall\ i, l \qquad (2)$$

where $R'_1 \overset{d}{=}, R_1$

$$R'_{l+1} \overset{d}{=} R_{l+1} - R_l\ \ \forall\ l$$

$$c'_1 \overset{d}{=} c_1$$

$$c'_{l+1} \overset{d}{=} c_{l+1} - c_1\ \ \forall\ l$$

$$s_{i,l'+1} \overset{d}{=} 0$$

and s_{il} and $s_{i,l+1}$ denote slack and surplus variables respectively for constraint l. Note that the surplus variable for constraint l, $s_{i,l+1}'$ is the slack variable for constraint $l+1$. The constraints (2)' and (3) together are totally unimodular and constitute a network since:

i) all entries are either 0, or ± 1.

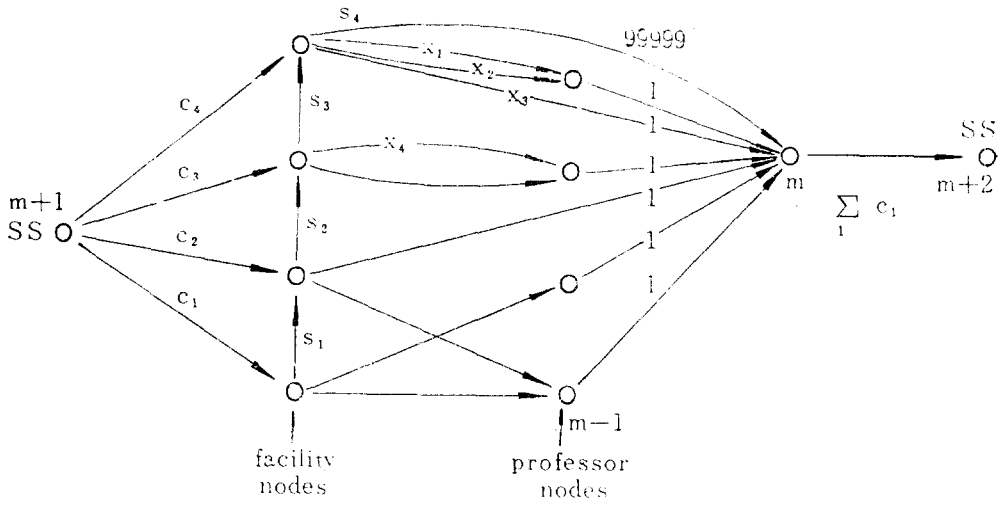
ii) constraint (2)' contains at most one $+1$ for the original variables (i.e., coefficient of x_{ij}) and at most one $+1$ and one -1 as coefficient for the slack and surplus variable (Note that $R'_l \cap R'_{l+1} = \phi\ \forall\ l$).

iii) constraint (3) contains at most one $+1$.

iv) constraints (2)′ and (3) are obviously bipartite.

Multiplying either constraint (2)′ or (3) by -1 we obtain a directed graph. An example is shown in Figure 1. The subscript i was removed for clarity. The node numbers, $m+1$ and $m+2$, denote super source and super sink respectively and the numbers on arcs represent their capacities.

〈Figure 1〉



The algorithm developed in this section utilizes the above imbedded network.

The branch and bound algorithm used here is basically a conventional one with the nice feature that the bound is easily calculated by solving a series of trivial subproblems (i. e., the aforementioned networks) instead of solving large linear programming problems. The approach adopted here is much the same as the branch and bound algorithm for the generalized assignment problem by Ross and Soland [47] except for computing bounds. Bounds are obtained from solving a series of network problems, instead of solving knapsack problems. The efficiency of the algorithm is based on the minimal effort required to solve the associated network problems with the efficient handling of candidate problems using a LIFO scheme.

Ⅲ. 3-2 LP-based Method

This approach is a conventional one, i. e., an ordinary LP-based integer programming

approach, either the branch and bound or the cutting plane method will work nicely. No one ever tried and conventional LP-based algorithm for this problem simply because of the magnitude of the problem size and the integer requirement. Surprisingly enough, however, in reality this conventional approach works beautifully, particularly with a code with GUB capability, such as Graves' ILP code.

As stated earlier, the problem matrix has a special structure, all zero-one entries, GUB constraints, and most of RHS of 1's. However this property alone dose not render any advantage in designing a solution algorithm. Yet computational experience shows that quite often using an ordinary LP for real problems produces natural integer optimum solutions. This fact naturally leads us to investigate the total unimodularity of the problem matrix.

It is, however, not totally unimodular in general.

Now we explore what makes the problem appears to be so amenable to integer optimal solution by an ordinary LP.

Theorem−1 [23] : (a_{ij}) is a totally unimodular matrix with $m \times n$ if and only if any non-empty set $I\{1, 2, \cdots, m\}$ can be divided into two disjoint sets I_1 and I_2 such that

$$\left| \sum_{i \in I_1} a_{ij} - \sum_{i \in I_2} a_{ij} \right| \leq 1 \quad (j = 1, 2 \cdots, n)$$

The following proposition is in order.

proposition-2: The constraint sets (1), (2) and (3) together in (P) constitute a totally unimodular matrix.

Proof: As shown earlier the constraint set (2) can be transformed into a set with one entry per column except for slack and surplus variables (for slack and surplus variables, at most two entries per column). The constraints (1) and (3) each have one entry per column. Therefore the constraint sets (1), (2) and (3) individually are totally unimodular and since the constraints (1), (2) and (3) can obviously be partitioned, any combination of the two constraint sets is also totally unimodular. Now we have to show the constraints (1), (2) and (3) altogether are unimodular. If we assign a sign to all constraints of the constraint set (2) and the opposite sign to all of the constraint set (3), then the resulting column sums of coefficients of the constraints (2) and (3) are either 0, +1 or −1. Since the constraint set (1) is a series of identity matrices (for a given

row, the coefficients of different courses do not couple), then constraint set (1) is independent of constraint sets (2) and (3) in terms of assigning a sign. By assigning the opposite sign of the column sum of the constraints (2) and (3) to the corresponding constraint (i. e., the constraint which contains 1 in the column) of (1), the above theorem is always satisfied. Q. E. D

The implication of this proposition is that if constraint set (4), which is relatively small in size in comparison with (1), (2) and (3), is not binding, the ordinary LP solution to the problem is naturally integral.

Total unimodularity is a necessary and sufficient condition to have all integer extreme points no matter what the RHS vector may be. (Total unimodularity is a sufficient condition to have all integer extreme points for problems with a given RHS.) What we are interested in is a necessary and sufficient condition to have all integer extreme points for problems with a given RHS. Berge [8] shows that any submatrix of a balanced matrix has this property against RHS of all 1's and yet it is not a totally unimodular matrix. A matrix which is neither totally unimodular nor balanced and nevertheless has the property that all basic feasible solutions to the LP with RHS of 1's are integral exists-a perfect matrix. Padberg [44, 45] gives a characterization of a perfect 0-1 matrix; unfortunately the identification of a perfect matrix is as computationally difficult as solving an equivalent ILP problem itself in an operational sense. Note that a perfect matrix is also a sufficient condition to have an integer optimal solution for problems with given objective function.

In summary, we have;

Totally Unimodular Matrix => Balanced Matrix => Perfect Matrix => Integer Optimal Solution.

As we can see from the above, total unimodularity is, in general, too strict a condition for having an integer optimal solution

Now we try to answer the question of what special structure of the problem other than total unimodularity enables an LP to generate integer solutions. Two of the special cases are: 1) as discussed earlier, the constraint set (4) is not binding (not in the basis), and 2) in the case that the constraint set (2) is not binding, then that the constraint matrix is balanced [8] (or more strictly speaking, perfect [44, 45]) is sufficient to have an all integral basis, which is a much weaker requirement than total unimodularity.

However it is very difficult to expect that any of the above cases are met in real problems. Nonetheless the test problems were all natural integer solutions.

Definition-3[15]; A square matrix A of order k with coefficient $a_{ij} \in \{0, \pm 1\}$ is called eulerian if $\sum_i a_{ij} \equiv 0 (\text{mod } 2)$ V j and $\sum_j a_{ij} \equiv 0 (\text{mod } 2)$ V i.

Theroem-4[15]: Let A be an $m \times n$ matrix with coefficients $a_{ij} \in \{0, \pm 1\}$. A is totally unimodular if and onlf if A does not contain any non-singular eulerian submatrix.

Now we are in the position to pressent the following property concerning the problem(P)

Property-5: Let A_1 be an $m \times n$ matrix with coefficients of constraint (2), (3) and (4) and A be a $(m+k) \times n$ matrix made up by adding k constraints of (1) to A_1. A is totally unimodular if and only if A_1 is totally unimodular.

Proof: (Sufficiency) By the definition of total unimodularity, A_1 is totally unimodular if A is totally unimodular since A_1 is a submatrix of A.

(Necessity) The constraint set (1) itself is totally unimodular since it has only one entry, +1, per column. The only case in which A is not totally unimodular when A_1 is totally unimodular is when A_1 is totally unimodular is when A_1 and constraint (1) consist of a non-singular eulerian sub-matrix according to the Theorem-4. However, the constraint (1) and A_1 can never make up an eulerian matrix since constraint (1) is coupled only by the i subscript (time slot) and constraints (2), (3) and (4) are coupled only by the j subscript (course). Q. E. D.

In other words the constraint set (1) never affects the solution in terms of not generating interger solutions, and yet it is always in the basis since the constraints are equality constraints. In fact the constraint set (1) aids to generate integer solutions even when the constraints (2), (3) and (4) contain non-singular eulerian submatrices. It may be considered that constraint set (1) functions as a cut. This will be explained later. Property 5 also proves the proposition 2, that constraints (1), (2) and (3) have a totally unimodular property, since constraints (2) and (3) can always by transformed into a network which, of course, is totally unimodular.

Now we can limit ourselves to considering only the cases of non-singular eulerian submatrices composed of (2), (3) and (4), instead of the full constraint matrix of (P).

Theorem-6: A constraint set $F(b) = \{x \mid Ax \leq b, \quad b: \text{integer}\}$, yields all integer extreme points (we assume $F(b) \neq \phi$) if and only if every basis matrix B such that $B^{-1} b \geq 0$, does

not contain any non-singular eulerian submatrix of A.

Proof: (Sufficiency) From Theorem 4, immediate. (Necessity) Every feasible basis is totally unimodular since it does not contain any non-singular eulerian matrix. Hence B^{-1} will contain only $\{0, \pm 1\}$. Therefore B^{-1} b is an integer vector since b is assumed to be integer. Q. E. D.

A stronger condition to insure all integer extreme points for the problem (P) follows.

Property-7: (P) has all-integral solutions if and only if (P) does not have at least two non-singular eulerian matrices in the basis, each of which comprises the same set of j subscripts (course).

Proof: Instead, we prove that it is not possible to have only one non-singular eulerian matrix in the basis. i) The constraint set (1) must be in the basis for (P) to be feasible, and since we assume only one non-singular eulerian matrix, all variables which correspond to the non-singular eulerian matrix must be 1. ii) To have a non-singular eulerian matrix in the basis, the basis must contain at least one constraint from the constraint set (4), since otherwise it is totally unimodular. The constraint which came from (4) contains at least two entries for it is part of the eulerian matrix. The RHS of constraint (4) is 1; therefore, all the variables in (4) can not have 1. i) and ii) contradict, hence the property holds. Q. E. D.

The non-existence of at least two non-singular eulerian matrices in the basis is a sufficient condition to have an optimal integer solution for a given RHS.

Ⅳ. IMPLEMENTATION AND COMPUTATIONAL EXPERIENCE

Two algorithms, based on the (1) Network-based branch and bound approach and the (2) LP-based method (hereafter we refer to Approach-1 and Approach-2 respectively), have been programmed and tested for the case of GSM, UCLA.

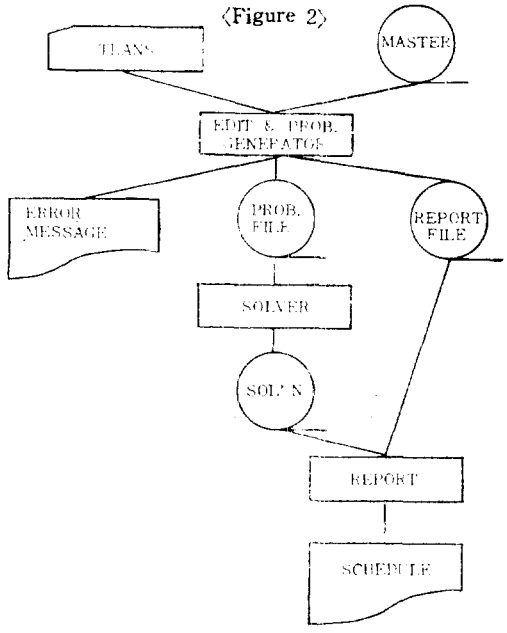
A run chart is shown in Figure 2. The EDIT program includes the following problem reduction procedures.

Some professors teach only one course, thus constraints from (3), professor conflict, can be dropped for those professors since there will be no time conflict. Likewise, for those student groups who require just one course and the rest are electives, constraints (4), student conflict, can be dropped. For various reasons, certain time-slots of a particular classroom (or faculty member) are reserved for specific purposes; we do not define

variables for these time-slots instead of setting them to zeros.

The following reduction rule is implemented. Let M be the row index set of A where A is the coefficient matrix. Let a_i be the i-th row of A. If for some i, $k \in M, a_i \geq a_k'$ then we can remove row k by redundancy. This rule can only be applied to constrains (3) and (4). For constraint set (2), if $|R_l| \leq c_{il}$ $\forall$ i, l, we can drop the i l constraint

〈Figure 2〉



since they are never binding. Other reduction rules which are somewhat more expensive to implement can be found in [5, 22]. EDIT also does the following cheap feasibility checks:

i) Facility capacity: $|R_l| > \sum_i c_{il}$ $\forall$ l

ii) Professor and student conflict: $|F_r| > |I|$ $\forall$ r and $|S_u| > |I|$ $\forall$ u.

The problem is infeasible if any of the above conditions are met.

After assigning courses to time slots, the SOLVER assigns courses to facilities by solving t (t is the number of time slots, $|I|$) separate transportation problems.

Minimize $\sum_j \sum_k c_{jk} x_{jk}$

$x_{jk} \in \{0, 1\}$

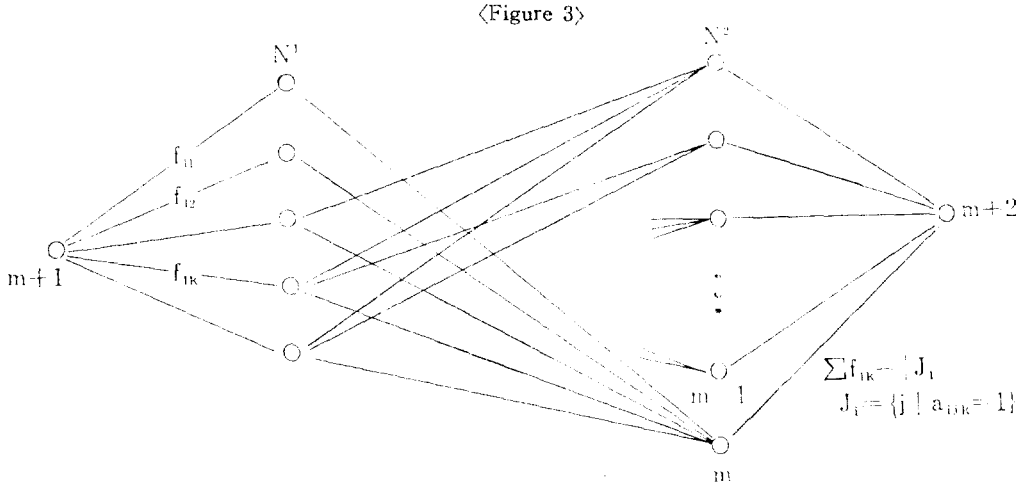
Subject to $\sum_k x_{jk} = 1$ $\forall$ j

$$\sum_j a_{ijk}x_{jk} \leq f_{ik} \quad \forall\ i=1-t,\ \ k \in k_j$$

$$(\sum_i a_{ijk}=1 \quad \forall\ j, k \in k_j)$$

where c_{jk} denotes disutility, subscripts i, j and k represent time slot, course, and facility respectively. c_{jk} was derived from a quadratic utility function ($c_{jk} \overset{d}{=} a(s_k - s_j)^2/s_j$) where a is a constant, s_j is the seating capacity of the correct size facility for course j). f_{ik} denotes availability of facility k at time i (1 if available, 0 otherwise). The problem separates into t independent transportation problems (T_i). The case of $i=1$ is depicted in Figure 3.

〈Figure 3〉



N^1 is the facility node set and N^2 is the course node set. Nodes $m+1$, and $m+2$ denote super source and super sink respectively and arc $(m)-(m+2)$ represents a slack arc.

The programs were written in ANSI FORTRAN for problems with up to 150 courses, 10 time slots, 20 classrooms, 100 professors, and 30 student groups. Approach-1 and -2 take 200K and 360K 32-bit words of core storage respectively.

Three real problems (Table 1) were solved.

Table 1. Test Problems

Problem ID	# of Courses	# of Facilities	# of Timeslots	# of Variables	# of Constraints
Prob-1	115	13	9	1035	1007
Prob-2	125	13	10	1250	1256
Prob-3	125	13	10	1250	1257

For Prob-1, randomly generated numbers 1 through 9 were used as preference ratings and actually surveyed preference ratings, measured on the same 1-9 scale, were used for Prob-2.

The computational results are shown in Table2 (in seconds of CPU time on the IBM 360/91).

Table 2. Computational Results

	Approach-1 (Network-based)				Approach-2 (LP-based)		
	CPU Time	Sol'n* Value	#of Iter.	#of Feas. Sol'n	CPU Time	Sol'n* Value	#of Pivots
Prob-1	1.45 23.75	269 262	43 661	1 4	2.49	251	277
Prob-2	45.21	375	1121	1	15.18	369	933
Prob-3**	60.00***	—	1692	0	11.44	—	806

* Objective function coefficients are measured on 1-9 scale

** Problem is infeasible

*** Terminated by time limit

Prob-3 was infeasible. Unfortunately the solution to Prob-1 and Prob-2 were natural integers so these results do not provide much insight into non-integer problems. However, we can conclude Approach-2 would be far superior to Approach-1 even for non-natural integer cases since we do not except many non-singular eulerian matrices in the basis as discussed in Section III. Approach-1 can produce feasible solutions quite rapidly if the problems are not overly constrained, as in the case of Prob-1, but it is worse in detecting infeasibility. The capability of producing feasible solutions early if a problem is not strictly constrained comes partly from the efficiency of the network code (GNET [11]) used.

The quality of the schedule is very encouraging. The average preference ratings (per course) of the optimum schedules were approximately 2.2 and 3 for Prob-1 and Prob-2, respectively. (Note that many preference coefficients for those courses whose professors did not have preference (34% of faculty) were set to 5 in Prob-2.) Recall that prefence was measured on a scale of 1 through 9, the lower the rating the more desirable the time slot. The facility utilization ratio (total assigned classroom-timeslots/total available classroom-timeslots) for the two problems were about 98% and 96%.

An attempt was made to generate a real problem with non-natural integer solution but

it did not succeed. This strongly supports the fact that the problem is amenable to soluti on by any efficient LP as discussed in Section III.

V. CONCLUSION

The class scheduling problem has long been noted and studied, and yet there have not been any successful analytical approaches to realistic problems, primarily due to the immense problem size and the integer requirement. However, this study reveals that the problem can rather easily be approached, at least for even time slot problems, through an ordinary ILP with a GUB handling capability. It is rendered solvable due to the fact that, for the most part (constraints (1), (2) and (3)), the problem matrix has the totally unimodular property and the existence of GUB constraints, Note that total unimodularity is a sufficient condition to have all integer extreme points for the problems with given RHS. Hence, even though the whole problem matrix does not possess total unimodularity, the problem may have all integer extreme points (or conservatively an integer optimal solution) as it was for the test problems.

Future study must focus on the characterization of a 'perfect' matrix—a necessary and sufficient condition to have an all integral basis for given RHS. Methods of scheduling multi-period (uneven time slot) lectures and university-wide scheduling of classes must also be resolved in order to be of full practical value.

BIBLIOGRAPHY

1. Almond, Mary, "An Algorithm for Constructing University Timetables," *The Computer Journal*, Vol. 8, No. 4 (January 1966), pp. 331—340.
2. ___ "A University Faculty Timetable," *The Computer Journal*, Vol. 2, No. 3 (August 1969), pp. 215—217.
3. Appleby, J. S., D. V. Blake and E. A. Newman, "Techniques for Producing School Timetables on a Computer and Their Application to Other Scheduling Problems," *The Computer Journal*, Vol. 3 (January 1961), pp. 237—245.
4. Aust, R. J., "An Improvement Algorithm for School Timetabling," *The Computer Journal*, Vol. 19, No. 4 (November 1976), pp. 339—343.
5. Balas, Egon and Manfred W. Padberg, "Set Partitioning: A Survey, *SIAM Review*, Vol. 18, No. 4 (October 1976), pp. 710—760.

6. Barraclough, Elizabeth D., "The Application of a Digital Computer to Construction of Timetables," *The Computer Journal*, Vol. 8, No. 2 (July 1965), pp. 136—146.
7. Berge, Claude, *Graphs and Hypergraphs*, North-Holland Publishing Company, 1973.
8. _____ "Balanced Matrices," *Mathematical Programming*, Vol. 2, (1972), pp. 19—31.
9. Bossert, W. H. and H. B. Harmon, *Student Sectioning on the IBM 7090*, IBM Corp., Cambridge, Mass., 1963.
10. Bowden, Joseph, *Making a Recitation Schedule*, Joseph Bowden, New York, 1922.
11. Bradley, Gordon H., Gerald G. Brown and Glenn W. Graves, "Design and Implementation of Large Scale Primal Transshipment Algorithms," *Management Sciece*, Vol. 24, No. 1 (September 1977), pp. 1—34.
12. Broder, Sol, "Final Examination Scheduling," *Communication of the ACM*, Vol. 7, No. 8 (August 1964), pp. 494—498.
13. Brown, J. Randall, "Chromatic Scheduling and the Chromatic Number Problem," *Management Science*, Vol. 19, No. 4 (December 1972), pp. 456—463.
14. Bush, Robert, Dwight Allen and Robert Oakford, *The Stanford School Scheduling System*, School of Education and Department of Industrial Research, Stanford University.
15. Camion, Paul, "Characterization of Totally Unimodular Matrices," *Proc. of the American Mathematical Society*, Vol. 16, No. 5 (October 1965), pp. 1068~1073.
16. Cole, A. J., "The Preparation of Examination Time-Table Using a Small-store Computer," *The Computer Journal*, Vol. 7, No. 2 (July 1964), pp. 117~121.
17. Corneil, D. G. and B Graham, "An Algorithm for Determining the Chromatic Number of a Graph," *SIAM J. Comput.*, Vol. 2, No. 4 (1973), pp. 311~318.
18. Csima, J. and C. C. Gotlieb, "A Computer Method for Constructing School Time Tables," *Communication of the ACM*, Vol. 7 (1964), pp. 160~163.
19. Dyer, James S., "Academic Resource Allocation Models at UCLA," in A. C. Heinlein (ed.), *Decision Models in Academic Administration*, The Decision Science Institute of the Center for Business and Economics, Kent State University, Kent, Ohio, 1974.
20. _____and John M. Mulvey. "An Integrated Optimization/Information System for Academic Departmenal Planning," *Management Science*, Vol. 22, No. 12. (August 1976), pp. 1332~1341.
21. Foxley, E. and K. Lockyer, "The Construction of Examination Timetables by Computer," *The Computer Journal*, Vol. 11, No. 3 (November 1968), pp. 264~268.
22. Garfinker, Robert S. and George L. Nemhauser, *Integer Programming*, John Wiley and Sons, 1972.
23. Ghouila-Houri, Alain, "Caracterisation des matrices totalement unimodulaires," *C. R. Academie des Sciences*, Vol. 254 (1962), pp. 1192~1194.
24. Gilmore, P. C., "Optimal and Sub-optimal Algorithms for the Quadratic Assignment Problem," *SIAM J. Appl. Math.*, Vol. 10 (1962), pp. 305~313.
25. Gotlieb, C. C., "The Construction of Class-Teacher Timetables." *Proc. IFIP Congress* 1962, 1963. pp. 73~77.
26. Graves, G. W. and R. D. McBride, "The Factorization Approach to Large-scale Linear Progra-

mming," *Mathematical Programming*, Vol. 10, No. 1 (February 1976), pp. 91~110.

27. _____ and A. B. Whinston, "An Algorithm for the Quadratic Assignment Problem," *Management Science*, Vol. 16, No. 7 (March 1970), pp. 453~471.
28. Hanan, M. and J. M. Kurtzberg, A Review of the Placement and Quadratic Assignment Problems," *SIAM Review*, Vol. 14 (1972), pp. 324~342.
29. Heller, I. and C. B. Tompkins, "An Extension of a Theorem of Dantzig's," in H. W. Kuhn and A. W. Tucker (ed.), *Linear Inequalities and Related Systems*, Princeton University Press, 1956, pp. 247~254.
30. IBM, *Student Scheduling System/360 Application Description Program Numbers 360A-US-06X, 360A-US-07X* (GH20-0202).
31. Knauer, Bernd A., "Solution of a Timetable Problem," *Computer & Operational Research*, Vol. 1 (1974), pp. 363~375.
32. Kodres, U. R., "Geometrical Positioning of Circuit Elements In a Computer," *Conference Paper* 1172, AIIE Fall General Meeting, October 1959.
33. Koopmans, T. C. and M. J. Beckmann, "Assignment Problems and the Location of Economic Activities," *Econometrica*, Vol. 25, No. 1 (1957), pp. 52~75.
34. Lawler, E. L., "The Quadratic Assignment Problem," *Management Science*, Vol. 9, No. 4 (1963), pp. 586~589.
35. _____"The Quadratic Assignment Problem: A Brief Review," in B. Roy (ed.), *Combinatorial Programming: Methods and Application*, D. Reidel Publishing Co., Dordrecht-Holland, 1975 pp. 351~360.
36. Lawrie, N. L., "An Integer Linear Programming Model of a School Timetabling Problem," *The Computer Journal*, Vol. 12, No. 4 (November 1969), pp. 307~316.
37. Liggett, Robin, Private communication (June 27, 1978),
38. Lions, J., "The Ontario School Scheduling Program." *The Computer Journal*, Vol. 10, No. 1 (May 1967), pp. 14~21.
39. Manlove, Donald C. and David W. Beggs, Ⅲ, *Flexible Scheduling*, Indiana University Press, 1965.
40. Marsten, Roy E., "An Algorithm for Large Set Partitioning Problems," *Management Science*, Vol. 20, No. 5 (January 1974), pp. 774~787.
41. Michalsen, Harald, *A Working Strategy for General School Scheduling*, (2nd ed.), Tapir, 1973.
42. Mulvey, John M., "A Calssroom/Time Assignment Model," HBS 76~11, Graduate School of Business Administration, Harvard University.
43. _____"Special Structure in Network Models and Associated Applications," Ph. D. Dissertation, UCLA 1975.
44. Padberg, Manfred W., "Perfect Zero-One Matrices," *Mathematical Programming*, Vol. 6 (1974), pp. 180~196.
45. _____ "Characterisations of Totally Unimodular, Balanced and Perfect Matrices," in B. Roy (ed.), *Combinatorial Programming: Methods and Applications*, D. Reidel Publishing Co.,

Dordrecht-Holland, 1975, pp. 275~284.

46. Rosenberg, Ivo, "Time-Table Scheduling," in Peter L. Hammer and Sergiu Rudeanu, *Boolean Methods in Operations Research and Related Areas*, Springer-Verlag, 1968, pp. 274~276.
47. Ross, G. Terry and Richard M. Soland, "A Branch and Bound Algorithm for the Generalized Assignment Problem," *Mathematical Programming*, Vol. 8 (1975), pp. 91~103.
48. Sakaki, T., K. Nakashima and Y. Hattori, "Algorithms for Finding in the Lump Both Bounds of the Chromatic Number of a Graph," *The Computer Journal*, Vol. 19, No. 4 (November 1976), pp. 329~332.
49. Schmidt, Gunther and Thomas Ströhlein, "A Boolean Matrix Iteration in Timetable Construction," *Linear Algebra and Its Applications*, Vol. 15 (1976), pp. 27~51.
50. Shih, Wei and James A. Sullivan, "A Linear Programming Model for Faculty Teaching Assignments," presented at the Joint National Meeting of the Operations Research Society of America and the Institute of Management Science, Boston, April, 1974.
51. Swabb, Alexander M., *School Administrator's Guide to Flexible Modular Scheduling*, Parker Publishing Co., 1974.
52. Swart, Wiliam W., "A Course Scheduling Problem: Model Development, Analysis, and Solution Algorithm," Presented to the 41st National Meeting of the Operations Research Society of America, New Orleans, April 26~28, 1972.
53. Welsh, D. J. A. and M. B. Powell, "An Upper Bound for the Chromatic Number of a Graph and Its Application to Timetabling Problems," *The Computer Journal*, Vol. 10, No. 1 (May 1967), pp. 85~86.
54. de werra, D. "A Few Remarks on Chromatic Scheduling," in B. Roy (ed.), *Combinatorial Programming: Methods and Applications*. D. Reidel Publishing Co., Dordrecht-Holland, 1975, pp. 337~342.
55. _____"Balanced Schedules," *INFOR J.*, Vol. 9, No. 3 (November 1971), pp. 230~237.
56. Wiley, W. Deane and Lloyd K. Bishop, *The Flexible Scheduled High School*, Parker Publishing Co., 1968
57. Wood, D. C., "A System for Computing University Examination Timetables," *The Computer Journal*, Vol. 11, No. 1 (May 1968), pp. 41~47.
58. _____"A Technique for Colouring a Graph Applicable to Large Scale Timetabling Problems," *The Computer Journal*, Vol. 12, No. 4 (November 1969), pp. 317~319.
59. Yule, A. P., "Extensions to the Heuristic Algorithm for University Timetables," *The Computer Journal*, Vol. 10, No. 4 (February 1968), pp. 360~364.