IIE Transactions
Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/uiie20

A Three Phased Approach To Final Exam Scheduling

TAGHI ARANI [a] & VAHID LOTFI [b]
[a] CompuNet , Alexandria, VA
[b] Jacobs Management Center , School of Management University of Buffalo , Buffalo, NY, 14260
Published online: 31 May 2007.

To cite this article: TAGHI ARANI & VAHID LOTFI (1989) A Three Phased Approach To Final Exam Scheduling, IIE Transactions, 21:1, 86-96, DOI: 10.1080/07408178908966211

To link to this article: http://dx.doi.org/10.1080/07408178908966211

A Three Phased Approach To Final Exam Scheduling

TAGHI ARANI
CompuNet
Alexandria, VA

VAHID LOTFI
Jacobs Management Center
School of Management
University of Buffalo
Buffalo, NY 14260

Abstract: A multi-phase examination scheduling process applicable to large university settings in general and SUNY at Buffalo (SUNYAB) in particular is proposed. Each scheduling phase is considered an integral part of the overall scheduling process and solved independently. Phase one of scheduling process is with the assignment of examinations to exam blocks (each containing one or more exams). The objective of this phase is to minimize the number of students taking more than one exam in the same exam block. The problem is solved using a variation of the quadratic assignment problem. Phase two of the scheduling process uses the results from phase one as input. The exam blocks are assigned to exam days in such a way that some measure of students' comfort is maintained. Phase two of the scheduling process is formulated as a set covering problem with an extra constraint. Phase three of the scheduling process which is involved with the assignment of exam blocks to exam periods in each day and optimal ordering of exam days is solved heuristically using a traveling salesman problem as part of solution procedure. The performance of the algorithms devised for the multi-phase scheduling process are tested both in terms of quality of the solutions obtained and the computer time to generate these solutions.

● The scheduling of examinations or assignment of examinations to time periods is a difficult problem for large universities at the end of each semester. Examination scheduling in its "most simple" version can be typically formulated in one of two ways [28].

Formulation 1. The total number of time periods is not predetermined. The problem is to make assignments so as to minimize the number of time periods under the restriction that no conflicts occur. A conflict occurs between two concurrently scheduled examinations if one or more students must take both examinations.

Formulation 2. The total number of time periods is predetermined. The problem is to assign the set of examinations to time periods so as to minimize the number of student conflicts.

Normally in practical problems other requirements are also to be satisfied. Those requirements which are termed side-constraints can be classified as follows:

i) A limit on the number of students taking examinations in any one period;

ii) room capacity constraints;

iii) consecutive examination constraints (i.e., certain examinations must be in adjacent periods);

iv) no-consecutive exam constraints (i.e., no examinations in succession for any student);

v) pre-assignment of some examinations to certain time periods;

vi) certain examinations may not be scheduled for particular periods;

vii) examinations should be uniformly scheduled over the examination period.

Requirements other than the above Seven may be necessary in a particular problem under study. As White and Chan [33]

Received October 1986; revised January 1988. Handled by the Applied Optimization Department.

state, "the requirements for a good examinations schedule are easy to statt: but difficult to achieve."

A review of the literature on scheduling of final examinations reveals that the primary goal has been to develop an examination schedule so that no student is required to take more than one examination in any time period. This, in fact, is such a common objective that the secondary objectives which have serious effects on large numbers of students have lost importance. Many authors use a graph theory approach to generate the overall exam schedule. Various restrictions introduced to the problem limit the use of a graph theoretic approach and related concepts. The integer programming formulations of the problem are generally so large in size that existing solution techniques have little value.

Although this problem has attracted the attention of many researchers in mathematical programming, the heuristic methods have been more successful in generating the final exam schedule in a reasonable amount of computer time. However, the heuristic methods do not guarantee "good" solution quality. To reconcile the two forementioned approaches, one should take into account the size of the problem, on one hand, and the quality of the solution, on the other.

We propose a three-phase scheduling approach in which each phase is considered an integral part of the overall process. The phases will be processed sequentially and the outcomes from one phase will be used as inputs to the next phase. In order to expand upon the concept of a multi-phase scheduling process, it is necessary that the following terminology be defined:

Exam Block: A group of one or more final exam(s) to be scheduled simultaneously.

Exam Period: A slot of time to which an exam block may be assigned to.

Exam Day: Made up of one or more exam period(s) (See Figure 1).

The first phase of our proposed process consists of the assignment of all final exams *(N)* to available exam blocks *(NP)* using Formulation 2. That is, in assigning the N exams to NP blocks, we approximately minimize the number of students with simultaneous exams.

The second phase of our procedure involves the assignment of the NP exam blacks to NP exam periods and the grouping of the periods to form exam days. Here, the objective is to minimize the number of students with two or more exams per exam day. We present a new set-covering-type integer programming formulation for this phase. The proposed problem is then solved using a Lagrangian Relaxation approach.

The third phase involves ordering the NPD exam periods in each day so that the number of students with two consecutive exams is minimized. At this point the exam days

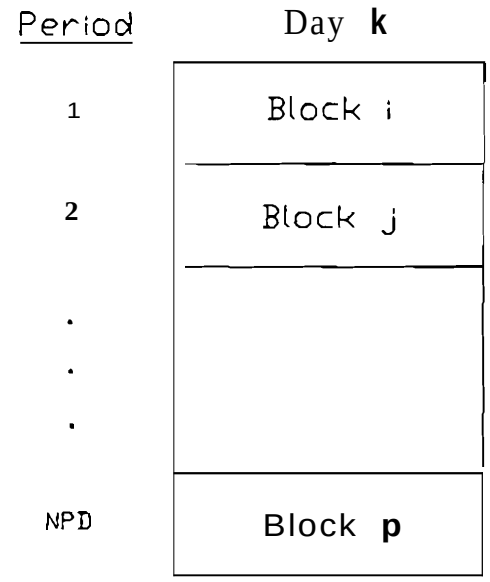


Figure 1. Pictorial Representation of an Exam Day

are ordered so that the number of students with an exam in the last period one day and another exam in the first period of the next day is minimized *(see* Figure 2).

In what follows, we present the three phases of our procedure. We first discuss the existing methodologies *(if* any) and then present other alternatives. Finally, we report some computational experience for the new alternatives using data obtained from the State University of New York at Buffalo (SUNYAB).

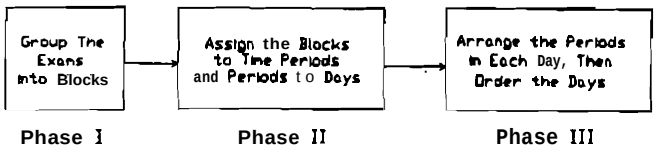


Figure 2. The Three Phases of the Scheduling Process.

Phase I: Assigning The Exams To Exam Blocks

As mentioned earlier, the focus of the first phase is to assign the N final exams to NP exam blocks. The problem is best described by utilizing graph theory. Let $G=(V,E)$ be a graph in which each vertex $v \epsilon V$ represents a final exam. An edge $(i,j) \epsilon E$ represents the existence of at least one student taking both exams i and j. Associated with each edge $(i,j) \epsilon E$, we define a_{ij} to be the number of students participating in exams i and j. The degree d_v of a vertex v is defined as the number of edges incident to it. For example, consider the graph of Figure 3 in which $d_7 = 6$. The weight $a_{25} = 10$ indicates that there are ten students who must participate in both exams 2 and 5.

Most of the existing algorithms for solving the phase one problem can be classified into one of two categories. The first category consists of the methods which attempt to solve

Formulation 1, using a graph coloring procedure. A coloring of G is an assignment of colors to the vertices of G such that no two adjacent vertices share the same color. For example, the graph of Figure 3 may be colored by assigning color 1 to nodes 1, 3, 4, 5, 6, 11 and 13, color 2 to nodes 2, 8, 9, 10 and 15, color 3 to nodes 7, 12 and 14, and color 4 to node 16. Note that other colorings of this graph with fewer colors may exist. In this context each color constitutes an exam period. The list of methods in category one is rather long. For this reason we will *not* present a survey of the literature for these methods. However, we have compiled a list of these algorithms in Table 1. Further, the interested reader may refer to [20], [7], [3], and [14].

Table 1. Graph Coloring Methods

Algorithm	Nature	Authors
Randomly ordered sequential	Heuristic	Matula et al.
Largest degree first	Heuristic	Wabh and Powell
Largest degree first: Fill from top	Heuristic	Peck, Williams
Largest degree first recursive: Fill from top	Heuristic	Dunston, Carter
Largest modified degree first	Heuristic	Williams, Dunston
Recursive largest degree first	Heuristic	Leighton
Largest modified 1degree	Heuristic	Dunston
Smallest degree last-recursive	Heuristic	Matula et al.
Smallest degree last-recursive with interchange	Heuristic	Matula et al.
Largest saturation degree	Heuristic	Brelaz
Maximum independent set	Heuristic	Lotfi and Sarin
Vertex sequential	Exact	Brown
Breath-first tree search	Exact	Christofides
Modified breath-first tree search	Exact	Wang
Modified breath-first tree search	Exact	Roschke, Furtado
Multiple intersection method	Exact	Korman
Dynamic reordering method	Exact	Korman
Contraction-connection method	Exact	Zykov
Modified contraction-connection method	Exact	Corneil, Graham

The second category includes methods which attempt to solve various mathematical programming models associated with Formulation 2. That is, exams are assigned to a predetermined number of exam blocks such that total number of students with simultaneous exams is minimized. For example, suppose that the sixteen exams of Figure 3 are to be scheduled into three blocks. One such a configuration is to use the same coloring we presented earlier but assign color 3 to node 16 also. In this case exams (nodes) 1, 3, 4, 5, 6, 11 and 13 are assigned to block 1, exams 2, 8, 9, 10 and 15 are assigned to block 2, and exams 7, 12, 14 and 16 are assigned to block 3. The total conflict for this schedule will be 10 (due to edge (12, 16)). In the remainder of this section, we will present a review of the literature pertaining to the existing methods in the second category.

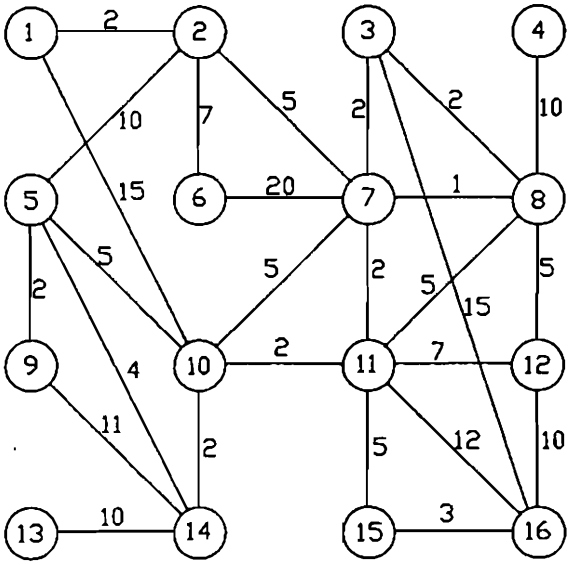


Figure 3. Graph of the Example Problem.

Many researchers have proposed various methods for solving the problem of final exam scheduling using Formulation 2 (see for example [5], [11], [26], [35], [17], [16], [30], [15], [33], [20], [29], [23], and [7]. Without exception, the primary objective in these procedures has been to minimize the number of students with two or more simultaneous exams. However, these procedures differ in that they include various types of constraints. One of the more common requirements has been the pre-assignment of a certain set of exams (see for example [26], [33], [20], and [23]).

Another common restriction has been the limitation on the capacity of class rooms (see for example [11], [35], [15] and [33]). Various other constraints such as: certain exams be consecutive, certain exams be held only in the morning, maximum number of three exams in a row, and certain exams not preceding others, have also been imposed. We now present a new approach that we implemented to solve the phase one problem. Our algorithm will assign the exams to exam blocks in such a way that the number of students with simultaneous exams is minimized. Although, our implementation does not involve satisfying any side constraints, restrictions such as the pre-assignment of some exams and a limit on the number of exams per block may readily be included. This

algorithm was tested on several real-world and structured problems.

Phase I: Problem Formulation and Proposed Algorithm

The problem of scheduling N exams into NP blocks such that the total conflict (number of students with simultaneous exams) is minimized (Formulation 2) may be formulated as a Quadratic Assignment Problem (QAP) without column constraints. That is, let $x_{ij} = 1$ if exam i is assigned to block j and 0 otherwise.. Then, the QAP without column constraints is formulated as,

$$\text{(P1)} \quad \text{Min } z = 2z' = \sum_{j=1}^{NP} \sum_{i=1}^{N} \sum_{k=1}^{N} a_{ik}\, x_{ij}\, x_{kj} \qquad (1)$$

$$\text{S.T.} \quad \sum_{j=1}^{NP} x_{ij} = 1 \quad i=1,2,\ldots,N \qquad (2)$$

$$x_{ij} = 0,1 \quad \text{for all } i \text{ and } j \qquad (3)$$

where a_{ik} is the number of students taking exam i and exam k.

In (1) z' gives the total number of students taking more than 1 exam in the same exam block and is to be minimized. (2) ensures that each exam is assigned to only one exam block. (3) restricts the decision variables to be integer, 0 or 1. The general quadratic assignment problem has proved to be much more difficult to solve than the linear assignment problem (see Murtagh *et* al. [24]). The QAP is not only nonlinear but it is not unimodal. The paper by Burhard and Stratman [6] provides a good review of the numerical investigation on QAP. However, due to the absence of column constraints, problem (P1) has a special structure. Carlson and Nemhauser [9] investigated the special structure of problem (P1). They developed and proved theorems which resulted in an efficient heuristic solution method to solve problem (P1).

In an attempt to solve (P1), we first used a heuristic (presented below) to obtain an initial solution. We then applied the Carlson's method to improve the initial solution. In obtaining the initial solution, exams are first sorted in descending order of weighted degrees. The weighted degree of an exam is:

$$w_i = d_i \sum_k a_{ik}$$

where d_i is the degree of exam (vertex) i and $\sum_k a_{ik}$ is the total number of students taking exam i and all other exams in conflict with exam i. For example, in the graph of Figure 3, $w_6 = 54$. The rationale behind the weighted degree is that it takes both the number of students and the degree of exams into account. Using a coloring algorithm this results in assigning the "hard" exams in earlier periods. The steps of the algorithm are as follows.

Let $v_{[1]}, v_{[2]},\ldots,v_{[N]}$ represent the N final exams (vertices) ordered as stated above.

Step 0: Choose the first exam from the top of the ordered list, say $v_{[1]}$ and assign it to block 1 (i.e., $v_{v_{[1]},1} = 1$). Set $n = 1$ and $j = 1$. Go to 1.

Step 1: If the list is exhausted, all exams are scheduled, go to 4. Otherwise, choose the next exam $v_{[n+1]}$ from the list and check to see if it is in conflict with exams already assigned to block t for $1 \le t \le j$. If it is not in conflict with any exams in a block, say k, the exam in question is assigned to block k (i.e., $x_{v_{[n+1]},k} = 1$) where k is the least block index used so far. Let $n \leftarrow n+1$ and repeat step 1 until no free block is available. Go to step 2.

Step 2: If $j < NP$ go to 3. Otherwise, let I_t be the index set of exams assigned to block t $(1 \le t \le j)$. Compute the following quantity,

$$t' = \min_t \sum_{s \in I_t} a_{[n+1],s}$$

Assign the exam $(n+1)$ to block t'. Let $n \leftarrow n+1$, go to 1.

Step 3: Let $j \leftarrow j+1$. Assign exam $v_{[n+1]}$ to block j. Let $n \leftarrow n+1$, then go to 1.

Step 4: All exams are scheduled. Use this solution x_{ij}, $i=1,2,\ldots,N, j = 1,2,\ldots,NP$ as a starting solution and apply the Carlson's method for possible improvements.

It is worth mentioning that before applying the above solution procedure, one may attempt to reduce the size of the problem by scanning the vertex degrees. That is, the exams with degree strictly less than NP (number of exam blocks) can be initially dropped from the problem and can be scheduled without creating conflict once the algorithm terminates. When there is a restriction on the number of exams scheduled in a particular block, a dropped exam (with a degree less than NP) is assigned to a least conflicting unfilled block.

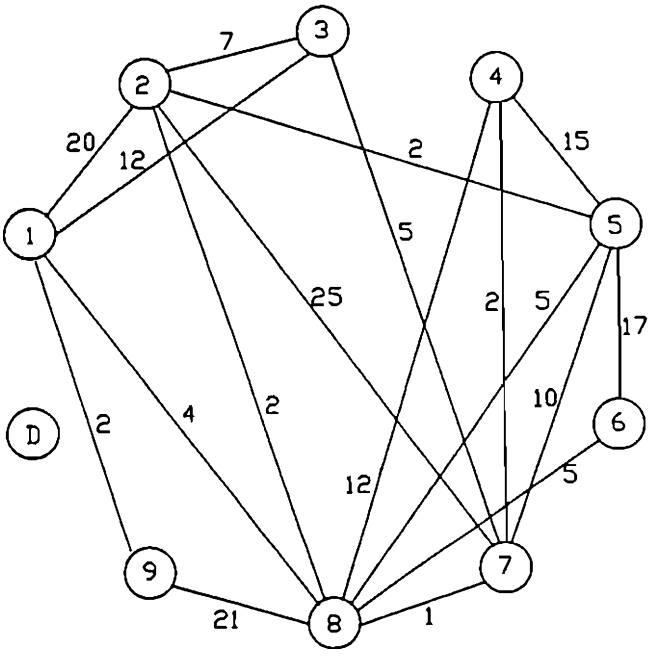
Phase II: Assigning the Exam Blocks to Exam Days

Phase one of the exam scheduling problem assigns all of the exams to NP exam blocks. For example, suppose that for the example problem of Figure 3 the number of blocks $NP = 9$. One solution to phase one for this problem is;

Block	1	2	3	4	5	6	7	8	9
Exams	1	6	2	3	11	12	7	8	9
	5	10		4	16		15	14	13

Clearly, the above solution has no first level conflict. The aim of phase two is to assign these nine blocks to NPD periods within ND days. One alternative is to make the assignment such that the number of students with consecutive exams is minimized.

When the last time period of one day and the first time period of the next day are considered consecutive (adjacent) the problem may be converted to a traveling salesman problem. In this case, each block may be considered a city and the optimum salesman tour constitutes the arrangement of the NP exam blocks such that the number of students with consecutive exams is minimized. Figure 4 presents the graph of the traveling salesman formulation of our example if we use the above solution to phase I. For example, the weight of 7 for the edge (2,3) (Figure 4) is the number of students with exams in both blocks 2 and 3 (i.e. exams 2, 6 and 10 with $a_{2,6} = 7$ and $a_{2,10} = 0$). Edges with ϕ weight are present in the graph but not shown in Figure 4. Note that an additional dummy block (labeled as node D) is utilized to indicate that the first period of the first day and the last period of the last day are not adjacent. Clearly, all edges incident to node D have zero weights.



Legend:
Each node represents an exam block
Missing arcs have zero length
Node (D) is fictitious.

Figure 4. Graph of the Traveling Salesman Formulation of the Example Problem.

Colijn [12] uses a heuristic procedure to solve the traveling salesman formulation of the problem. His approach reduced the number of students with consecutive exams by about 30 percent. White and Chan [33] present a computer program which uses a variation of an algorithm by Peck and Williams [26] to assign the exams to blocks. They also proceed by using a traveling salesman approach to assign the blocks to time periods.

Another possibility is that each exam day contains exactly two (NPD = 2) time periods and the second period of one day is not considered adjacent to the first period of the next day. In this case, the assignment of the NP exam blocks to the time periods reduces to a minimum weighted matching problem. The end-nodes of the edges in the matching are the pairs of exam blocks scheduled in the same day.

In many practical situations there are three or more time periods per day and the last period of one day need not be considered adjacent to the first period of the next day. An important issue here is the students who have to take two or more exams in a day. This becomes practically more important when the university administration intends to reduce the total number of time periods for the overall final examination schedule. This was the case taken under consideration by the SWNYAB administration in the Spring of 1985. The administration expressed the desire to reduce the length of the final examinations period from the traditional Seven days and three periods per day to a five days with three periods per day. As a consequence, the number of blocks were to be reduced from twenty 'one to fifteen.

In this case, the problem of assigning the exam blocks to time periods is neither the traveling salesman problem nor the weighted matching problem. In what follows, we present a new set-covering-type formulation which will minimize the number of students with two or three exams per day (second degree conflict). We then propose a Lagrangian Relaxation based solution method to obtain exact solutions for the proposed problem.

Let all of the final examinations be scheduled in NP blocks. There are NV different arrangements of NP blocks into groups of NPD each. Where,

$$NV = \frac{NP!}{NPD!\ (NP-NPD)!}.$$

Note that of the NV different arrangements, we wish to find ND arrangements which form a feasible and least conflicting set.

Clearly, when $NP > NPD \times ND$ there is no feasible solution. In case $NP < NPD \times ND$, we introduce sufficient dummy blocks to make them equal. For example, there are 20 different arrangements of 6 blocks into groups of 3, e.g.,

$$(1,2,3),\ (1,2,4),\ \ldots\ldots,\ (4,5,6).$$

Let c_j be the total number of students having two or more exams in the e m blocks that form the jth arrangement. Let $y_j = 1$ if the jth arrangement is a part of an optimum sohtion and $y_j = 0$, otherwise. Then, the problem of minimizing the total number of students taking two or more exams in one day is given as:

$$\text{(P2) Min } Z = \sum_{j=1}^{NV} c_j y_j \tag{4}$$

$$\text{S.T.} \quad \sum_{j=1}^{NV} e_{ij} y_j = 1 \qquad i = 1,2,\ldots,NP \tag{5}$$

$$\sum_{j=1}^{NV} y_j = ND \tag{6}$$

$$y_j = 0,1 \qquad j = 1,2,\ldots,NV \tag{7}$$

where $e_{ij} = 1$ if block i is in arrangement j and $e_{ij} = 0$ otherwise. (4) represents the total number of students participating in two or more exams in the same day. Constraints (5) force each block to be assigned to exactly one day. Constraints (6) are used to limit the number of days to ND days. (P2) is a binary integer programming problem with NV variables and $NP + 1$ constraints.

It is easy to show that the quality in (5) may be replaced with greater than or equal to without changing the set of optimal solutions of (P2). In that case, (P2) becomes a set covering problem with an additional constraint (6). Several specialized versions of the implicit enumeration method of Balas [2] have been proposed for the set covering problem. The most relevant of these is by Pierce and Lasky [27]. However, their approach is more appropriate for problems without constraint (6).

We utilized a branch and bound procedure similar to that of Pierce and Lasky to solve problem (P2). However, instead of solving the linear programming relaxation at each node, we used a Lagrangian relaxation approach to obtain better bounds at each node of the branch and bound tree. For problem (P2) a natural Lagrangian relaxation is defined below by relaxing constraint set (5).

$$\text{(Pu) Min } Z = \sum_{j=1}^{NV} c_j y_j + \sum_{i=1}^{NP} u_i \sum_{j=1}^{NV} (1 - e_{ij} y_j) \tag{8}$$

$$\text{S.T.} \quad \sum_{j=1}^{NV} y_j = ND \tag{9}$$

$$y_j = 0,1 \qquad j = 1,2,\ldots,NV \tag{10}$$

$$u_i \geq 0 \qquad i = 1,2,\ldots,NP.$$

Equivalently, (Pu) may be written as:

$$\text{(Pu) Min } Z = \sum_{j=1}^{NV} (c_j - \sum_{i=1}^{NP} e_{ij} u_i) y_j + \sum_{i=1}^{NP} u_i \tag{11}$$

$$\text{S.T.} \quad \sum_{j=1}^{NV} y_j = ND \tag{12}$$

$$y_j = 0,1 \qquad j = 1,2,\ldots,NV, \tag{13}$$

$$u_i \geq 0 \qquad i = 1,2,\ldots,NP.$$

Our solution procedure consisted of using the solution to the above Lagrangian problem as a bounding scheme in each node of the branch and bound tree. Moreover, we utilized various side constraints to enhance the quality of the bound (see Arani [1]). To solve the Lagrangian problem, we need the values of multipliers u_i. Initially, we solved the linear relaxation of (P2) to obtain these values at node Zero. We then used a subgradient search procedure to selectively update these multipliers at subsequent nodes.

The performance of the branch and bound technique can be improved significantly by providing a good initial upper bound. We developed a heuristic procedure to solve (P2) in order to obtain the initial incumbent.

To begin, NP exam blocks are to be grouped into ND days with NPD exam blocks per day. Suppose that ND $(NPD\text{-}1)$ blocks have been, arbitrarily, assigned to the first $(NPD\text{-}1)$ periods of every day and we are about to assign the last ND blocks to the last period of each day. The optimal arrangement of the last ND blocks among the remaining ND last periods, such that the total additional conflict (beyond those of $NPD\text{-}1$ periods of each day) is minimized, is the usual assignment problem. For example, suppose that we have arbitrarily assigned blocks (1,2), (3,4), and (5,6) to the first two periods of days 1, 2, and 3 respectively as follows;

Period	Day 1	Day2	Day3
1	1	3	5
2	2	4	6
3	?	?	?

where ? represents the block to be assigned to the last period of each day. Clearly, we have three candidates, block 7, 8, or 9. The assignment of the three blocks to the last three periods may result in additional conflict (beyond that of blocks in the first two periods). This is the usual assignment problem with the cost matrix.

3rd Period of Day	Candidate Blocks 7	8	9
1	25	6	2
2	7	12	0
3	10	10	0.

For example, assigning block 7 to the third period of day I will result in 25 additional students with two or more exams per day (see Figure 4). An optimal solution to the above assignment problem is to assign block 7 to day 2, block 8 to day 1, and block 9 to day 3. The additional conflict is 13 and the total number of students taking two or more exams per day equals 50.

Once an optimal solution to this assignment problem is found, we may fix the blocks assigned to the last periods.

Note that the arbitrary assignment of blocks to the first $NPD-1$ periods in conjunction with the optimal assignment of the last periods results in a suboptimal solution for the overall problem and consequently an upper bound.

To continue reducing the total conflict within each day, we backtrack to the $NPD-1$st periods and try to rearrange the ND blocks (assigned arbitrary). As with the last period, we keep the remaining $NP(NPD-1)$ blocks fixed and solve the associated assignment problem. For our example this will result in:

Period	Day 1	Day 2	Day 3
1	1	3	5
2	?	?	?
3	8	7	9

where ? represents the blocks to be assigned to the second periods (candidates are 2, 4 and 6). The resulting assignment cost matrix is

2nd Period of Day	Candidate Blocks 2	4	6
1	22	12	5
2	32	2	0
3	2	15	17

The optimal solution for this problem is to assign block 2 to day 3, block 4 to day 2, and block 6 to day 1. The additional conflict will be 9 with the total number of students with two or more exams equaling 18.

This backtracking process continues until the assignment problem for the first period is solved. Then a forward move is attempted by solving an assignment problem for the second periods. The process of rearranging ND blocks at a time is repeated, going back and forth between the first to last periods, until no further reduction is possible. At this point the heuristic terminates and the last solution is used as the initial incumbent for the branch and bound method.

Phase III: Arranging the Exam Days and Exams Within Each Day

Consider for example an optimal solution to (P2) in which $Y_1 = \{1,3,4\}$, $Y_2 = \{2,7,8\}$, and $Y_3 = \{5,6,9\}$ are equal to 1. That is, the block combinations i, j and k result in minimum number of students with two or more exams in the same day. How should the three blocks combinations be assigned to the three exam days? There are 3! or six possible arrangements. Moreover, once an arrangement is selected, we must decide on the configuration of the exam blocks within each block combination (in our case each block combination contained three blocks, hence three configurations).

The latter question can be answered by selecting the two most conflicting exam blocks in each block combination and assigning them to the first and third periods of each day. For our example, in the ith combination the conflicts are, $a_{13} = 10$, $a_{14} = 0$, and $a_{34} = 0$ (see Figure 4), we assign exam block 1 to the first period, and exam block 3 to the third period of a day. In general, when there are more than three blocks per day, this problem can be formulated as a traveling salesman problem. That is, for each of the ND days, we must solve a $(NPD+1)$-node traveling salesman problem in which the nodes represent the NPD blocks within that day. The $NPD+1$st node is fictitious and is used similar to that of Figure 4.

Once we have configured the blocks within each block combination we may continue to find the optimal sequencing of the exam days. In this case, an optimal sequence of exam days is such that the number of students having exams during the last period of one day and the first period of the next day is minimized. This problem can also be formulated as a traveling salesman problem. Here, each exam day corresponds to a city and the weight c_{ij} for arc (i,j) is the number of students taking exams in the last period of day i and the first period of day j. Again, because the first exam day is not predetermined, we must introduce an additional fictitious city $ND+1$ with arc costs $c_{ND+1,j} = c_{j,ND+1} = 0$ for all j. The optimal traveling salesman tour then, represents the optimal sequence of exam days.

Computational Results

The solution methodologies proposed for the three phases of the exam scheduling problem were programmed and tested using several real-world and randomly generated problems.

Phase I algorithm (PH-1) was evaluated on three real-world and three structured problems. The results were compared with those of two other methods. First, we tested PH-1 on two real-world problems obtained from the scheduling office at SUNYAB (for Spring 1984 and 1985). This part of the testing was done on a CDC Cyber 815, under the NOS 2.1.

There were an average of 27600 students enrolled for 981 final exams in the Spring of 1984 and 766 final exams in the Spring of 1985 at SUNYAB. The student conflict matrix and course conflict matrix were generated from the course-list provided by the scheduling office. Generation of required data took an average of 200 seconds of CPU time. After generating the data, exams with degree of less than NP (number of exam blocks available) were initially dropped from the data files. Table 2 presents the number of exams which were left after screening the data for different exam blocks.

table 2. Number of Dropped and Retained Exams

	Spring 84		Spring 85	
NP*	Dropped	Retained	Dropped	Retained
15	354	627	240	526
18	405	576	280	486
21	438	543	316	450

*NP = Number of exam blocks

Table 3 presents the results of PH-1 for the above two data sets. These results compare favorable with those of the current algorithm (clue to Broder [5]) employed at SUNYAB. The current algorithm has been programmed in FORTRAN on a Univac 1100/90, under UPS/OS 39R4. Broder's algorithm, being probabilistic, is run 100 times to produce a reasonable schedule (one with a minimum first-level conflict). In this case, it resulted in 12 conflicts for 21 exam blocks (for Spring 84) and 123 conflicts for 15 exam blocks (for Spring 85). In its present implementation, it requires about 25 minutes of execution time to generate 100 schedules.

Table 3. Results of Phase I (PH-1) Algorithm Assignment of Exams to Exam Blocks

NP	Spring 84	Spring 85	Average Execution Time (sec)
15	118*	50	95.6
18	38	6	88.2
21	1	0	71.2

*Number of students with two or more exams in the same block.

As seen from Table 3, the computational time for 21 exams blocks is somewhat smaller than the time for 15 exam blocks. This is because more exams are initially dropped in case of 21 blocks than in 18 or 15 blocks (see Table 2). Once the retained exams are scheduled, we begin to assign the dropped exams one at a time. Each dropped exam is assigned to a block with which it does not result in any conflict.

To further evaluate the performance of PH-1, we solved three structured problems (obtained from [8]) and one more real-world problem (SUNYAB data for Spring 1987). Table 4 presents the characteristics of the structured problems. The Spring of 87 problem included 762 final exams.

Table 4. Characteristics of the Structured Problems

Problem	Number of nodes	Number of edges	Density (percent)	Largest Clique*
A	139	1,396	14.6	14
B	181	4,706	28.9	18
C	190	4,793	26.7	21

*The largest subgraph complete

The above four problems were solved using three different methods on an IBM 3081-GX under CMS. The first two methods consisted of PH-1 and Broder. The third method is similar to that of Mehta's [23]. Mehta, implemented a graph coloring algorithm to first color the graph. Once the graph was colored, the extra time frame, beyond the allotted number of frames, had to be resolved. Each time frame was examined for resolution and the potential increase in the total amount of conflict was computed. The frame with minimum increase was resolved by reassigning its exams to other time frames.

Our implementation was similar to Mehta's in that we first used a graph coloring algorithm to assign the exams to as many time fiames as needed (to obtain a zero first-level conflict assignment). However, since this assignment may result in more than one time frame to resolve, we sequentially selected those frames with the minimum number of exams to resolve. The exams in such frames were reassigned, one at a time (in a decreasing order of degree) to the least conflicting time frames.

The results of the three methods on the stated four problems are presented in Table 5. As seen from Table 5, Bro-

Table 5. Results of the Three Phase I Methods Assignment of Exams to Exam Blocks

Problem	NP	Phase I Number of conflicts	Phase I Exec. Time*	Broder** Number of conflicts	Broder** Exec. Time	Variation of Mehta Number of conflicts	Variation of Mehta Exec. Time
A	6	166	2	197	20	241	6
	9	55		96	25	54	5
	12	4	1	54	27	4	4
B	12	237	2	284	52	245	11
	15	77	3	134	80	108	8
	18	33	2	78	104	40	8
	21	4	1	33	79	6	6
C	12	263	2	329	91	334	10
	15	125	2	173	127	184	15
	18	58	1	94	132	81	11
	21	17	1	65	113	19	4
SUNYAB/ SP. 87	15	132	17	592	481	185	28
	18	29	13	377	556	53	24
	21	0	11	269	602	26	19

*Execution times are in seconds
**Best of 100 trial runs, execution time is total for 100 runs
• Less than one second

der's method consistently performs worse than the other two methods in both the number of conflicts and the execution time. Although the execution times for Phase I are somewhat less than those of the Mehta's, the difference is insignificant for a practical purpose. In terms of the number of conflicts, except for one case (Problem A, $NP = 9$), Phase I outperforms Mehta's variation.

In order to evaluate the performance of the Phase II algorithm we solved several randomly generated problems first. Table 6 presents the characteristics of the randomly generated problems. This initial computational experience was implemented to test various parameters and side-constraints (see Arani [1]) and develop a most efficient code. The results of the "fine-tuned" algorithm are presented in Table 7. The fine-tuned branch and bound algorithm was further improved by performing the subgradient search (to update the Lagrangian multipliers) only at nodes whose bounds were within 5% of being fathomed. This resulted in a reduction of about 34% in the execution times and 83% in the number of nodes processed.

Table 8. Characteristics of the Randomly Generated Problems for Phase II

Problem Set	NP	NPD	ND	No. of Variables	No. of Constraints
1'	15	3	5	455	16
2	18	3	6	816	19
3	21	3	7	1,330	22

'Each set consists of five problems.

Table 7. Results of the Randomly Generated Problems

Problem Set	Total Time	Bound Time	Total Nodes
1'	13.8''	3.1	713
2	63.7	19.7	2,268
3	190.2	46.4	4,273

'Entries represent averages for the five problems in the set.
''Execution times are in seconds.

The latter version of the algorithm was utilized to solve the first two SUNYAE problems. To compare the relative performance of the proposed Phase II method (PH-2), we also solved the same two problems using three other approaches. First, we used a naive approach (SEQNTL) by assigning the NP exam blocks sequentially to exam periods. That is, exam block 1 was assigned to the first period of the first day, etc. Second, we formulated the problem as a traveling salesman problem (TSP) and solved it using the Held and Karp [17] algorithm. Third, we recorded the solutions generated through the heuristic procedure (HURST) for obtaining the upper bound in Phase II. Each problem was solved for 15, 18, and 21 blocks. The results are presented in Tables 8 and 9.

As seen from Tables 8 and 9, in terms of computational time, SEQNTL and HURST are about the same and outperform the other two methods. However, in terms of solution quality, only HURST outperforms TSP and SEQNTL consistently performs worse than the other three methods. In all cases, PH-2 outperforms the other three approaches in terms of minimizing the number of students with two or more exams per day. For some cases (e.g. Spring 85, $NP = 18$ and 21), HURST performs slightly better (than PH-2) on reducing the number of students with three exams per day.

Table 8. Results of Phase II Algorithms for Spring 84 Assigning Exam Blocks to Exam Days

NP	Solution Method	No. of students with 2 Exams	3 Exams	Total	Exec. Time
15	SEQNTL	15,154	587	15,741	0.07'
	TSP	11,394	209	11,603	0.76
	HURST	10,588	181	10,769	0.11
	PH-2	10,314	187	10,501	2.26
18	SEQNTL	13,383	495	13,878	0.19
	TSP	9,425	206	9,631	57.40
	HURST	8,157	121	0,278	0.24
	PH-2	7,854	86	7,940	14.88
21	SEQNTL	13,559	511	14,070	0.18
	TSP	7,424	103	7,527	34.50
	HURST	6,290	42	6,332	0.33
	PH-2	8,108	45	6,153	9.09

'Execution times are in seconds on a CDC Cyber 815.

Table 9. Results of Phase II Algorithms for Spring 85 Assigning Exam Blocks to Exam Days

NP	Solution Method	No. of students with 2 Exams	3 Exams	Total	Exec. Time
15	SEQNTL	12,825	758	13,583	0.07'
	TSP	9,354	239	9,593	0.79
	HURST	8,610	149	0,759	0.12
	PH-2	8,399	146	8,545	18.23
18	SEQNTL	12,025	790	12,815	0.12
	TSP	8,247	113	8,360	38.40
	HURST	6,835	68	6,903	0.24
	PH-2	6,271	81	6,352	4.11
21	SEQNTL	11,728	733	12,501	0.12
	TSP	7.512	98	7,610	14.18
	HURST	5,135	28	5,163	0.37
	PH-2	4.708	36	4,744	6.83

'Execution times are in seconds on a CDC Cyber 815.

Once, the exam blocks were assigned to exam periods, we used the heuristic procedure presented in Phase III to form the TSP. We then used Held and Karp's [17] algorithm to solve the resulting TSP. Again, in order to have a basis for comparison, we determined the number of students with consecutive within days and between days for the other three methods (i.e., SEQNTL, TSP, and HURST). The resulting TSPs for the proposed Phase III (PH-3) had 6, 7, and 8 nodes for the 15, 18, and 21 blocks problems, respectively. The execution times for the entire Phase III procedure (including the heuristic as well as the TSP) ranged from 1 to 3 seconds

(on a CDC Cyber 815). The results are presented in Tables 10 and 11. As seen from these tables, SEQNTL Consistently performs poorly when compared with the other methods. On the total consecutiveness, TSP outperforms the other methods (recall however that this method is not the best for minimizing the number of students with two or more exams per day). PH-3 and HURST are about the same.

Table 10. Results of Phase III Algorithm for Spring 84
Arranging the Exam Days and Exams Within Each Day

NP	Solution Method	Consecutive within Days	Consecutive between Days	Total
15	SEQNTL	10,357'	2,934"	13,291
	TSP	5,261	3,495	8,756
	HURST	5,819	3,723	9,542
	PH-3	5,674	3,722	9,396
18	SEQNTL	9,037	2,292	11,329
	TSP	3,711	2,631	6,342
	HURST	4,382	2,810	7,192
	PH-3	3,948	3,236	7,184
21	SEQNTL	9,153	1,669	10,822
	TSP	2,688	1,655	4,343
	HURST	2,357	2,688	5,045
	PH-3	2,214	2.704	4,918

'Number of students with two consecutive exams in a day.
"Number of students with an exam in the last period of one day and the second exam in the first period of the next day.

Table 11. Results of Phase III Algorithm for Spring 85
Arranging tho Exam Days and Exams Within Each Day

NP	Solution Method	Consecutive within Days	Consecutive between Days	Total
15	SEQNTL	9,079'	2,005"	11,084
	TSP	4,263	2,659	6,922
	HUAST	4,704	2,803	7,507
	PH-3	4,518	3,042	7,560
18	SEQNTL	8,845	1,460	10,305
	TSP	2,747	1,967	4,714
	HURST	2,113	2,629	4,742
	PH-3	2,939	2,626	5,565
21	SEQNTL	8,661	1.042	9,703
	TSP	1,401	1,681	3,082
	HURST	1,514	2,945	4,459
	PH-3	1,491	2,087	3,578

'Number of students with two consecutive exams in a day.
"Number of students with an exam in the last period of one day and the second exam in the first period of the next day.

Conclusions

A three-phase approach to final examination scheduling, applicable to large university settings, was developed. The first phase consisted of assigning the exams to blocks so as to approximately minimize the number of students with simultaneous exams. The second phase involved grouping the exam blocks into days such that the number of students with *two* or more exams per day were minimized. The third phase addressed the problem of arranging the exam days and exams within each day.

Our proposed methods are being implemented at SUNY-AB. Since we completed this study, our CDC Cyber mainframe computers were replaced with an IBM 3081-GX. Consequently, some of our codes (being machine dependent) had to be rewritten. We have completed the programs for the first phase. We have also implemented a new feature in the program which allows it to be used as a decision support tool. After generating the initial (Phase I) schedule, the program reports (for every exam and every block) the potential increase in total conflict if an exam were to be reassigned to a block other than the one it is currently assigned to. This enables our scheduling coordinator to rearrange those exams which will not increase the total conflict as desired. Further, the program *can* also impose a limit on the number exams assigned to each block.

Acknowledgement

The authors wish to thank the reviewers for their valuable suggestions.

REFERENCES

[1] Arani, T., "A Multi-Phase Final Examination Scheduling Problem." Doctoral Dissertation, SUNYAB, 1986.

[2] Balas, E., "An Additive Algorithm for Solving Linear Programs with Zero-one Variables." *Operations Research*, 1965, 13, 517-546.

[3] Brelaz, D., "New Method to Color the Vertices of a Graph." *Communication of the ACM*, 1979, 22, 251-256.

[4] Brown, J. R., "Chromatic Scheduling and the Chromatic Number Problem." *Management Science, 1972, 19, 456-463.*

[5] Broder, S., "Final Examination Scheduling." *Communication of the ACM*, 1964, 7, 494-498.

[6] Burhard, R. E. and Stratman, K. H., "Numerical Investigations on Quadratic Assignment Problems." *Naval Research Logistic Quarterly*, 1978, 25, 12948.

[7] Carter, M. W., "A Decomposition Algorithm for Practical Timetabling Problems." *University of Toronto, Dept. of industrial Engineering*, Working Paper No. 83-86, 1983.

[8] Carter, M. W., private communication.

[9] Carlson, R. and Nemhauser, G., "Scheduling to Minimize Interaction Cost." *Operations Research*, 1967, 14, 52-58.

[10] Christofides, N., "An Algorithm for Chromatic Number of a Graph." *Computer Journal*, 1971, 14, 38-39.

[11] Cole, A. J., "The Preparation of Examination Time-table Using a Small-Store Computer." *Computer Journal*, 1964, 117-121.

[12] Colijn, A. W., "Reduction of Second Order Conflict in Examination Timetables." As cited in White and Chan (1979).

[13] Corneil, D. G. and Graham, B., "An Algorithm for Determining the Chromatic Number of a Graph." *SIAM Journal of Computing*, 1973, 2, 311-318.

[14] Dunston, F. D. J., "Sequential Colorings of Graphs." *Proc. 5th British Comb. Conf., Aberdeen, 1975, Cong. Num. XV, Utilitas Math.*, 1976, 151-158.

[15] Desroches, S., Laporte, G. and Rousseau, J. M., "HOREX A Computer Program for the Construction of Examination Schedules." *INFOR*, 1978, 16, 294-298.

[16] Grimes, J., "Scheduling to Reduce Conflict in Meetings." *Communications of the ACM.* 1970, 13, 351-352.
[17] Hall, A. and Acton, F., "Scheduling University Course Examinations by Computers." Communications of the ACM, 1967, 10, 235-238.
[18] Held, K. and Karp, R. M., "The Traveling Salesman Problem and Minimum *Spanning* Trees." *Operations Research*, 1970, 18, 1138-1162.
[19] Korman, S. M., The Graph-Coloring Problem. In *Combinatorial Optimization*, John Wiley and Sons, New York, 1979.
[20] Leighton, F., "A Graph Coloring Algorithm for Large Scheduling Problems." *Journal of Research of the National Bureau of Standard*, 1979, 84, 489-506.
[21] Lotfi, V. and Sarin, A., "A Graph Coloring Algorithm for Large Scale Scheduling Problems." CORS, Forthcoming.
[22] Matula, D. W., Marble, G. and Isaacon, *1.* B., "Graph Coloring Algorithms." In *Graph Theory and Computing*, Ed., New York, Academic Press, 1972.
[23] Mehta, N. K., "The Application of a Graph Coloring Method to an Examination Scheduling Problem." *Interface*, 1981, 11, 57-64.
[24] Murtagh, B. A. and Jefferson, T. R., "A Heuristic Procedure for Solving the Quadratic Assignment Problem." *European Journal of Operational Research*, 1982, 9, 71-76.
[25] Paul, M. C. and Unger, S. H., "Minimizing the Number of States in Incompletely Specified Switching Function." As *Cited in Hall and Acton*, 1967.
[26] Peck, J. E. L. and Williams, M. R., "Algorithm 286 Examination Scheduling." *Communications of the ACM*, 1969, 9, 433434.
[27] Pierce, J. F. and Lasky, J. S., "Improved Combinatorial Programming Algorithms for a Class of all Zero-One Integer Programming Problems." *Management Science*, 1973, 19, 528-543.
[28] Salazar, A. and Oakford, R. V., "A Graph Formulation of a School Scheduling *Algorithm." Communications of the ACM*, 1974, 12, 696-698.
[29] Tripathy, A., "A Lagrangian Relaxation Approach to Course Timetabling." *Journal of the Operational Research Society*, 1980, 31, 599-603.
[30] Vishnevskii, L., "A Scheduling Problem." *Cybernetics*, May-June, *1976*, 115-118.
[31] Wang, C. C., "An Algorithm for the Chromatic Number of a Graph." *Journal of ACM*, 1974, 21, 385-391.
[32] Welsh, D. J. A. and Powell, M. B., *"An* Upper Bound for the Chromatic Number of a Graph and Its Application to Timetabling Problems." *Computer Journal*, 1967, *10*, 85-86.
[33] White, G. and Chan, P., "Towards the Construction of Optimal Examination Schedules." *Canadian Journal of Operational Research and Information Processing*, 1979, 17, 219-229.
[34] Williams, M. R., "Heuristic Procedures—(If they work, leave them alone)." *Software Practice and Experience*, 1974, 4, 237-240.
[35] Wood, D. C., "A System for Computing University Examination Timetables." *Computer Journal*, 1968, 11, 41-47.
[36] Zykov, A. A., "On Some Properties of Linear Complexes." *Mat. s.b.*, 1949, 24, 163-188, *Amr. Math. Soc.* Translation No. 79, 1952.

Vahid Lotfi is an Assistant Professor of Management Science and Systems in the School of Management, State University of New York at Buffalo. He received his Ph.D. in Operations Research from SUNY at Buffalo. His articles have appeared in *EJOR*, *OR Letters* and *Computers* and *Operations Research*. He is co-author of Decision Support Systems for Production and Operations Management. His current research interests are multiple criteria decision making and applied operations research.

Taghi Arani received his Ph.D. in Operations Research from SUNY at Buffalo. He has published in *EJOR*. He is a consultant at CompuNet, Washington, DC. His current research interests are mathematical programming and applied operations research.