Journal of the Operational Research Society (1998) 49, 635–646 ©1998 Operational Research Society Ltd. All rights reserved. 0160-5682/98 $12.00
http://www.stockton-press.co.uk/jor

An empirical comparison of heuristic methods for creating maximally diverse groups

RR Weitz[1] and S Lakshminarayanan[2]
[1] *Seton Hall University, NJ and* [2] *MinMax Consulting, NY, USA*

This research identifies, describes, and empirically contrasts five heuristics for forming maximally diverse groups of any specified size from a given population. Diversity is based upon multiple criteria specified by the decision maker. The problem has immediate application in academic or training settings where it may be desired to create class sections, or project groups within classes, such that students are immersed in a diverse environment. Furthermore this research has an even broader utility, as the problem is mathematically identical to an eclectic set of applications ranging from final exam scheduling to VLSI design. Here we consider five different heuristics, drawn from student-workgroup assignment and final exam scheduling applications. The methods are tested on a 'real-world' data set and evaluated on the criteria of solution quality and computational resources.

Keywords: allocation; education; heuristics, maximal diversity; timetabling

Introduction

This research identifies, describes, and empirically contrasts five heuristics for forming maximally diverse groups of any specified size from a given population. The problem has immediate application in academic or training settings where it may be desired to create class sections, or project groups within classes, such that students are immersed in a diverse environment. Furthermore this research has an even broader utility, as the problem is mathematically identical to an eclectic set of applications ranging from final exam scheduling to VLSI design.

The problem is onerous due to the multiple criteria and the large number of students (and hence large solution space); it has been shown to be NP-complete.[1,2] Here we consider five different heuristics for solving the problem, drawn from student-workgroup assignment and final exam scheduling applications. One method is constructive, two are based on pairwise switching, and two are based on 'G-wise' switching (where G is the number of groups). The methods are tested on a 'real-world' data set and evaluated on the criteria of solution quality and computational resources.

Mathematical programming representation

The maximum diversity student workgroup problem may be formulated as a quadratic integer program as follows.[2]

$$\max \sum_{p=1}^{G} \sum_{i=1}^{N-1} \sum_{j>i}^{N} d_{ij} x_{ip} x_{jp}$$

Correspondence: Dr RR Weitz, Computing & Decision Sciences, Stillman School of Business, Seton Hall University, South Orange, NJ 07079, USA.
E-mail: weitzrob@lanmail.shu.edu

subject to

$$\sum_{p=1}^{G} x_{ip} = 1 \quad \text{for } i = 1, 2, \ldots, N$$

$$\sum_{i=1}^{N} x_{ip} = S \quad \text{for } p = 1, 2, \ldots, G$$

There are N students, G groups, and $N/G = S$ students in each group

$$d_{it} = \text{difference between students } i \text{ and } j.$$
$$x_{ip} = 1 \text{ if student } i \text{ is in group } p,$$
$$0 \text{ otherwise.}$$

As an example, at the European Institute of Business Administration (INSEAD) the entire entering class is partitioned into sections (students in the same section travel together from class to class) and each section is partitioned into groups for project work. At INSEAD, the objective is for each section and each group to be as diverse as possible, based on the criteria of nationality, mother tongue, gender, age, university attended, undergraduate discipline, and area of previous work experience. The Stern School of Business at New York University (NYU) follows a similar procedure using the criteria of nationality, gender, discipline, GMAT verbal and GMAT quantitative scores. (In this work, we use data from NYU).

The difference between two students is obtained by summing the weighted contributions of all criteria by which the two students differ. At NYU, the first three characteristics are measured on a nominal scale. The GMAT scores are directly represented as percentiles; that is, a difference in GMAT verbal scores between two students of 20% is twice the difference of 10%.

Weights reflect the relative importance of each criterion. At NYU (and INSEAD) initial values were provided by the administrators based on their experience and goals in partitioning the students. (More detailed discussions of the difference calculations and the weighting scheme may be found in Reference 3).

Literature review

The multiple-criteria maximum diversity student workgroup problem

The maximum diversity student workgroup problem was first described by Weitz and Jelassi[3] in their work detailing the development and implementation of a multiple-criteria, heuristic-based decision support system at INSEAD. (The heuristic is method 'WJ' described below.) A modified version of that system was subsequently implemented at NYUs Stern School.[2] As part of the NYU project, an integer upper bound for the solution was developed, several integer programming (IP) versions of the problem were formulated, and empirical trials were conducted comparing the performance of the heuristic with that of the IP approaches.

Mingers and O'Brien[4] focus on creating similar student groups; that is, the objective is to disperse each student characteristics as evenly as possible between groups. (This goal of minimising between group differences implies maximising the distribution of characteristics within groups, and is hence akin to the maximum diversity problem.) Using only binary-valued criteria, they contrast variations of their sequential multiple-criteria heuristic approach with a goal program. One variation of the heuristic works towards maximising a metric drawn from Information Theory, two others directly attempt to minimise deviations across groups. In subsequent work O'Brien and Mingers[5] extend their heuristic approach to include multi-valued nominal and ordinal attributes, interval valued attributes, and problems with unequal group sizes. In addition, they suggest a protocol for coding the data which they claim improves the solutions obtained via the heuristic.

Mathematically equivalent problems

Lotfi and Cerveny[6] implemented a system for scheduling final exams at a large university; they describe an approach which decomposes the entire problem into four sub-problems, as follows:

> The multi-phase final-exam-scheduling process consists of first grouping the final exams into sets called 'blocks' in such a way as to minimise the number of students with simultaneous exams. Secondly, the blocks are assigned to exam days while minimising the number of students with two or more exams per day. Thirdly, the exam days and exam blocks within days are arranged so as to minimise the number of students with consecutive exams. Fourthly, exams are assigned to classrooms so as to maximise the space utilisation.

The sub-problem of assigning exam blocks to days is mathematically equivalent to the minimisation version of the student–group IP presented above. For this application,

d_{ij} = number of students with exams in both block i and block j.

x_{ip} = 1 if exam block i is assigned to day p, 0 otherwise.

Lotfi and Cerveny[6] and Arani and Lotfi[7] describe implementations for solving the final exam scheduling problem in real-world settings. Their heuristics for solving phase two of the problem are given below as methods LC and AL.

Another mathematically identical application is in VLSI design; here the task is to group highly connected modules onto the same circuit. (Consider each module to be a student or exam block, and the connectivity between modules to be d_{ij}s). Feo and Khellaf[1] showed that the problem is NP-complete, and devise a bounded, 'matching-based' set of heuristics using an approach drawn from graph theory. In subsequent work Feo *et al*[8] developed improved bounds for two of their heuristics. These graph theoretic methods are not considered here; a more detailed description and empirical evaluation of these methods is provided.[9]

Related problems

Group formation: mathematical programming approaches

Kuo *et al*[10] focus on the problem of forming a *single* group selected from a given population. The objective is to create the most diverse group based upon multiple criteria. They developed a [0,1] integer program formulation with a quadratic objective function (which they show to be NP-hard), and two equivalent linear, mixed integer models. Hill *et al*[11] use a linear [0,1] integer programming approach with the goal of maximising student preferences in assigning students to project teams, and students to job interviews. Reeves and Hickman[12] use a multiple-criteria, mixed-integer, linear programming model for assigning MBA students to summer field study project teams. Miyaji *et al*[13] use a goal programming approach to assign students to laboratories based upon students' preferences for the labs and the laboratories' preferences for the students.

Group formation: single-criterion heuristic approach

Beheshtian-Ardekani and Mahmood[14] used a simple, sequential heuristic for assigning students to project groups. Their goal was to create groups balanced by the single criterion of experience. Muller[15] conducted an experiment examining the effectiveness of the this balanced-group method. Donahue and Fox[16] repeated Muller's experiment and considered several additional hypotheses.

Group manufacturing

The idea here is to improve overall manufacturing performance by grouping parts into part families (based on similar processing requirements) and machines into 'cells' which can satisfy the processing requirements of one or more families. Typically, similarity coefficients are used and a common approach is to view the task as a clustering problem, although math programming, graph theoretic, and other techniques have been applied as well. One perspective on the problem addressed in this work is to view it as a clustering problem with fixed size clusters. See Singh[17] for a review of group manufacturing approaches.

The heuristics

The foregoing section has outlined work on forming maximally diverse (student) groups, research on mathematically identical problems in the domains of final exam scheduling and VLSI design, and has briefly made the connection to some related problems. In this section we describe the heuristics under study here: these include a constructive method (Weitz–Jelassi), two related pair-switching methods (Lotfi–Cerveny and Lotfi–Cerveny–Weitz) and two related G-wise switching methods (Arani–Lotfi and Arani–Lotfi–Weitz). The constructive method is drawn from student workgroup research; the switching methods are drawn from work on the final exam scheduling problem.

Method 'Weitz–Jelassi' (WJ)

The basic heuristic implemented at INSEAD and NYU works by avoiding placing the most similar students in the same group. The first student, usually selected randomly, is placed in the first group. The heuristic then selects the student least different from the first student and places him/her in the next group. The model continues in this fashion, at each iteration taking the student least different to the previous student and placing him/her in the next group.

(1) *Randomly select a starting student, and assign this student to group 1.* Initialise: (a counter) $c = 1$; $x_{ip} = 0$ for all i, p; $p = 1$. Select starting student i (arbitrarily). Let $x_{ip} = 1$. Mark student i assigned. Increment c.

(2) *Of those students still unassigned, select the student most similar to (least different from) the last student assigned. (In case of a tie, arbitrarily break it.) Assign this student to the next group.* Increment p. If $p > G$, reset $p = 1$. Find k such that $d_{ik} = \min_j d_{ij}$ for all students j unassigned. Set $x_{kp} = 1$. Mark student k assigned.

(3) *Stop if all students have been assigned; otherwise continue.* Increment c. If $c > N$, stop. Otherwise, set $i = k$ and go to step 2.

The basic WJ method requires $N^*(N-1)/2$ comparisons of the d_{ij}s to reach a solution. As the method is computationally quite simple, for any conceivable circumstance it would be possible (and advisable) to run the method 'N times', each time starting with a different student, and then to select the best of the generated solutions. In the empirical section of this work, we take this approach.

Final exam scheduling

Method 'Lotfi–Cerveny' (LC)

The following heuristic is from Lotfi and Cerveny[6] Appendix B. We have replaced exam blocks with students, and exam days with groups, and switched the objective from minimisation to maximisation. Previously we discovered several flaws in Lotfi and Cerveny's heuristic—these are corrected here; for a complete discussion of our revisions see Weitz and Lakshminarayanan.[18] LC is based on pairwise switching of students.

(1) *Create an arbitrary initial solution*

$$x_{ip} = \begin{cases} 1 & \text{for } i = (p-1)^*S+1, (p-1)^*S+2, \ldots, \\ & (p-1)^*S+S, \text{ and } p = 1, 2, \ldots, G \\ 0 & \text{otherwise} \end{cases}$$

(2) *For the present assignment, determine the matrix R which specifies for all students i and all groups p, the contribution to the total difference (a) that is a result of student i currently being assigned to group p, or (b) that would result if student i (not currently in group p) were added to group p.* Let $D = [d_{ij}]$ and $X = [x_{ip}]$, determine $R = DX$. Set $i = 0$, and Flag = false, and go to step 3.

(3) *Select (the next) student i, and determine which group q is most likely to contain a student j with whom a switch with student i might improve the objective.* Increment i. If $i > N$, go to step 6. Otherwise find $r'_{iq} = \max r_{ip}$. If $x_{iq} = 0$ go to step 4. Otherwise, repeat step 3.

(4) *Determine which (if any) student j in group q should be switched with student i.* In row i of X find column t with $x_{it} = 1$. For each row j in which $x_{jq} = 1$, let $w_j = (r_{iq} - r_{it}) + (r_{jt} - r_{jq}) - 2^*d_{ij}$. Let $w_k = \max w_j$. If $w_k > 0$, (that is, the objective function increases if students i and j are switched) go to step 5, else go to step 3.

(5) *Revise X to incorporate the switch. Recalculate R. Set Flag to true to indicate that a switch has taken place.* Let $x_{kq} = 0$, $x_{kt} = 1$, $x_{iq} = 1$, and $x_{it} = 0$. Let $R = DX$, Flag = true and go to step 3.

(6) *If no students have been switched, stop. Otherwise reset the Flag and i, and go to step 3.* If Flag = false, stop. No further improvement is possible. Otherwise, let Flag = false, $i = 0$, and go to step 3.

Method 'Lotfi–Cerveny–Wetiz' (LCW)

LCW is a variation of LC proposed in Weitz and Lakshminarayanan.[18] Here we remove from step 3 the condition limiting the search for potential students j (for switching with the current student i) to those students assigned to group q. This condition, 'find $r'_{iq} = \max r_{ip}$', is an attempt at selecting a single 'best' group to be involved in the switch with the current student, thereby minimising the number of calculations required at each iteration of the heuristic. LCW eliminates this constraint and considers *all* students j assigned to *all* groups other than the group to which student i is currently assigned. LCW revises steps 3 and 4 in LC as follows.

(3) *Select (the next) student i as a potential candidate for a switch.* Increment i. If $i > N$, go to step 6. Otherwise go to step 4.
(4) *Determine which (if any) student j should be switched with student i.* In row i of X find column t with $x_{it} = 1$. For each column $q \neq t$, consider each row j, in which $x_{jq} = 1$, let $w_j = (r_{iq} - r_{it}) + (r_{jt} - r_{jq}) - 2{*}d_{ij}$. Let $w_k = \max w_j$. If $w_k > 0$, go to step 5, else go to step 3.

LC must at each iteration do $(G - 1)$ comparisons of r_{ik}s, calculate (N/G) w_js, and do $(N/G) - 1$ comparisons of the w_js. (This presumes r_{ik} is not a minimum for the group to which student i is currently assigned.) LCW does no comparison of r_{ik}s, calculates $(G-1){*}(N/G)$ w_js and does $(G-1){*}(N/G) - 1$ comparisons. Using LCW, there is an increase in computational effort per exchange considered; however, intuitively LCW may provide improved solutions, and reduce the number of solutions considered.

Method 'Arani-Lotfi' (AL)

Arani and Lotfi[7] use a different heuristic towards solving the same problem. Their heuristic proceeds by first arbitrarily assigning all the students (in the same manner as LC). Then it removes G students, one from each group, and reassigns them optimally, given the current assignment of the other $N-G$ students. (Optimal here means that these G students are assigned such that the total additional diversity, beyond that of the $N-G$ students already assigned, is maximised.) The optimal assignment process is then repeated using a different set of G students, one from each group. The heuristic continues, G students at a time, for S iterations, until all students have been optimally assigned in turn. Now the process is repeated 'in the other direction': consider those G students assigned in iteration $S - 1$ and reassign them optimally; then do the same with the G students assigned in iteration $S - 2$. Continue until the G students in iteration one are assigned. The process works its way backwards and forwards until no assignments are changed in a full cycle of iterations. Again, it may be noted that AL is equivalent to a 'G-wise switching' of students at each iteration.

(1) *Create an (arbitrary) initial solution.* Here we choose, for ease of implementation, to assign students in batches of size G, one to a group.
   (a) Initialise: (a counter) $c = 1$.
   (b) $x_{ip} = 1$ for $i = (c - 1){*}G + p$, and $p = 1, 2, \ldots, G$
   0 otherwise
   Mark this set of G students assigned at the cth iteration, as G_c. Increment c; $c > S$, stop. Otherwise repeat step 1b.
(2) *Optimally reassign the students, G at a time, starting with the last set ($c = S$), and ending with the first set.*
   (a) Initialise: $c = S$; Flag = false.
   (b) Optimally reassign students G_c (the reassignment procedure is given below). If this process results in a change in assignment for any of the G_c students, set Flag = true. Decrement c; if $c < 1$, to go step 3. Otherwise, repeat step 2b.
(3) *If no assignments have been changed through an entire cycle stop. Otherwise repeat the reassignment process 'in the other direction'.* If Flag = false, stop. Otherwise, go to step 4.
(4) *Optimally reassign the students, G at a time, starting with the first set and ending with the last set ($c = S$).*
   (a) Initialise: c = 1; Flag = false.
   (b) Optimally assign students G_c (the reassignment procedure is given below). If this process results in a change in assignment for any of the G_c students, set Flag = true. Increment c; if $c > S$, to step 5. Otherwise, repeat step 4b.
(5) *If no assignments have been changed through an entire cycle, stop. Otherwise repeat the reassignment process 'in the other direction'.* If Flag = false; stop. Otherwise, go to step 2.

Reassigning the G students at each iteration is, as Arani and Lotfi[7] put it, 'the usual assignment problem'. To solve it, we adapted a publicly available subroutine.[19] For this application the assignment problem objective becomes one of maximisation; it may be formalised as follows

$$\max \sum_{p=1}^{G} \sum_{i \in \{G_c\}} d'_{ip} x_{ip}$$

subject to

$$\sum_{p=1}^{G} x_{ip} = 1 \quad \text{for } i \in \{G_c\}$$

$$\sum_{i \in \{G_c\}} x_{ip} = 1 \quad \text{for } p = 1, 2, \ldots, G$$

where $x_{ip} \geqslant 0$, and

d'_{ip} = the additional diversity in group p resulting from placing student i in group p, given the $S-1$ students currently in group p; that is,

$$d'_{ip} = \sum_{\substack{j=1 \\ j \notin \{G_c\}}}^{N} d_{ij} x_{jp}$$

The structure of the problem results in values for x_{ip} which are binary 0,1.

Method 'Arani–Lotfi–Weitz' (ALW)

Method ALW is identical to AL except that the G students assigned at each iteration are selected *randomly*, one from each group, from those students who have not yet been considered for reassignment during the current cycle. We surmised that improved solutions may be obtained without the 'same group' constraint; our concern was that with this adjustment the heuristic would not converge as quickly, or at all.

(1) *Create an (arbitrary) initial solution.*

(2) *Randomly select G students at a time, one from each group, from those students who have not yet been reassigned during the current sequence. Optimally reassign these G students.*

   (a) Initialise: (a counter) $c = 1$; Flag = false; mark all students 'unassigned'.

   (b) Randomly select G students, one from each group, from among those marked 'unassigned'. Mark these selected students 'assigned'. Optimally reassign these G students. If this process results in a change in assignment for any of the G students, set Flag = true. Increment c; if $c > S$, go to step 3. Otherwise, repeat step 2b.

(3) *If no assignments have been changed through an entire cycle stop, otherwise repeat the reassignment process.* If Flag = false, stop. Otherwise, go to step 2a.

Methods AL and ALW do an assignment problem at each iteration; the assignment problem is well structured and may be solved in polynomial time. Of course, for all the switching methods (LC, LCW, AL and ALW) it is not possible to predict the number of iterations that will be required to reach a solution.

The experiments

Each of the above methods was tested on a real-world data set. In these experiments, two figures were determined: solution quality and computational resources required. The former is given by the total diversity of the resulting partition (given mathematically by the objective function of the mathematical program), and the latter by the cpu processing time required by the method.

The data

The above techniques were tested using Fall 1993 data from the Stern School at NYU. As some of the methods require the same number of students per group, we deleted four arbitrarily selected students from the 1993 data set, leaving 360 students in the class. Each row of the data set contained the student number, name, and the five criteria of interest: nationality, gender, discipline, GMAT verbal and GMAT quantitative scores.

We followed requirements at NYU (which appear to be fairly typical for this application), and partitioned the entire class into six blocks of 60 students each, and each block into ten working groups of six students each. Additionally, experiments were conducted on the following problems: partitioning ten students into two groups, 12 students into three groups, and 12 students into four groups. These problems were chosen for several reasons: (1) in terms of size, they are on the order of the final exam scheduling and VLSI problems reported in the literature, (2) it was of interest to compare the methods on both large and small problems, and (3) Weitz–Jelassi results for these sets were already available from previous research which contrasted the method with mathematical programming approaches.[2]

The Weitz–Jelassi heuristic

For Weitz–Jelassi, the experiment proceeded in the following manner. (The description here will be for the 60 student, ten groups problem; the procedure is analogous for the other cases.) Firstly, the list of students was randomised, then the first 60 students were selected. The heuristic was applied to this set of 60 students, starting with the first student in the set, with ten groups of size six formed. For each group, the sum of the (different) pairwise differences was calculated; the sum of these ten within-group differences is the measure of the quality of the solution reached by the heuristic, starting with that particular student. (This is the objective function of the IP representation of the problem.) The process was then repeated on this (60 student) data set an additional 59 times, each time starting the heuristic with a different student. The maximum (best) and average of the 60 difference (quality) measures was then determined; these provide a gauge for the performance of the heuristic on this set of (60) students for this number of groups. The entire process was then repeated for the next 60 students, and so on until the list of students was exhausted. For the 60 student problem, this procedure provides maximum and average heuristic values for six (that is, 360/60) sets of data.

Detailed results for the 60 student, ten group problem are provided in Table 1. (Detailed results for the other problems are in Weitz and Lakshminarayanan[20]). Summary results (over all sets) for all problems are in Table 2. In addition to the best and average WJ results, these tables include: (1) the average solution quality for 100 arbitrary allocations, (2) the results of applying WJ with the objective of *minimising* diversity, and (3) an integer upper bound. The 100 random allocations are provided as a baseline measure. The WJ heuristic can easily be adapted for minimising diversity by directing it to select the most *different* student at each iteration; these 'worst case' results are provided to indicate the range of the distribution of possible solutions. (For example, one can imagine achieving or approaching this worst case periodically in real-world circumstances where allocations are performed randomly.) The integer solution upper bound is provided as another means of assessing the quality of the heuristics, and is described below.

An integer bound

A integer upper bound (which may be shown to be superior to the relaxation of the integer programming formulation of the problem) was developed in Lakshminarayanan and Weitz.[2] This bound may be calculated by considering the calculation for the average difference metric for a group (and thereby for an entire allocation). For each student i in a group, there are $(S-1)$ pairwise differences which may be calculated (all possible d_{ij}s, $i \neq j$). The sum of these $S(S-1)$ differences divided by two comprises the group difference; the sum of all the group differences in a particular partition is the difference metric for the partition. A maximum bound may therefore be determined by considering the matrix of pairwise differences between all students, and calculating the difference metric using the largest $(S-1)$ differences for each student. Clearly this method does not guarantee a feasible solution; however it does provide an upper bound on a feasible solution.

More formally, the integer bound is given by

$$\frac{1}{2}\sum_{i=1}^{N}\sum_{j\in M(i)} d_{ij}$$

where $M(i)$ is the largest $(S-1)$ differences for student i.

Interpretation of results

To interpret the results of these (and later tables) consider the NYU application and the example of the initial partitioning process of 360 students and six groups. Table 1 indicates that over the long run, on average, the solution afforded by using WJ best is expected to outperform a random allocation by 9.76%. If NYU uses a random procedure (or some other procedure) and the allocation happens to be particularly poor in terms of diversity ('worst case'), the WJ best solution can be expected to outperform it by roughly 20%. If the institution uses a single WJ run to perform their allocations, they can expect over the long term to achieve allocations about 1.18% worse than if they use WJ best. Finally Table 1 indicates that for this problem WJ best provides solutions that are about 65% of the upper bound; in other words, the worst case is that these solutions are approximately 65% of the (unknown) optimal solution.

The other heuristics

In order to avoid effects due to the order of the students, the following procedure was followed for each of the other four heuristics; again, for illustration purposes, the 60 student, ten group problem will be described. Starting with the randomised list of 360 students, the first 60 were selected. The heuristic was then applied to this set of 60 students, and the following data recorded: the initial (arbitrary) solution value, the final heuristic value, and the number of cycles and number of switches undertaken by the heuristic. (A cycle is completed whenever the heuristic has iterated through the entire set of students; a switch occurs whenever two or more students are exchanged.) This set of 60 students was then re-randomised. The heuristic was then again applied to the set and the results recorded. The randomisation and application of the heuristic process was then repeated until 100 solutions for the set were obtained. The process was then repeated for the remaining (five) sets. So, for each (60 student) set, for each heuristic, there are 100 initial (arbitrary) partition values, 100 final heuristic values, 100 cycle counts and 100 number of

Table 1 Weitz–Jelassi heuristic results (sets of 60 students divided into 10 groups, each of size 6)

No. students No. groups $N=60$, $G=10$	*Worst (minimise diversity)*	*Average of 100 random partitions*	*Average (maximise diversity)*	*Best (maximise diversity)*	*Integer bound*	*% Improvement worst to best*	*% Improvement 100 avg. to best*	*% Improvement avg. WJ to best*	*Best/bound*
Set no. 1	63.13	71.70	77.79	78.46	111.56	24.29	9.43	0.86	0.70
no. 2	58.16	61.98	67.00	67.99	109.85	16.91	9.70	1.47	0.62
no. 3	55.20	58.17	63.78	64.67	105.75	17.15	11.16	1.38	0.61
no. 4	58.13	64.49	68.85	69.90	108.30	20.26	8.39	1.52	0.65
no. 5	55.94	64.72	70.47	71.02	106.07	26.97	9.73	0.78	0.67
no. 6	59.83	66.47	72.46	73.21	109.25	22.37	10.14	1.03	0.67
mean	58.40	64.59	70.06	70.88	108.46	21.32	9.76	1.18	0.65
Std. dev.	2.86	4.52	4.81	4.71	2.25	4.00	0.90	0.32	0.03

Note: % Improvement A to $B=((B-A)/A)*100$.

Table 2 Weitz–Jelassi heuristic performance summary

	Worst (minimise diversity)	*Average of 100 random partitions*	*Average (maximise diversity)*	*Best (maximise diversity)*	*Integer bound*	*% Improvement worst to best*	*% Improvement 100 avg. to best*	*% Improvement avg. WJ to best*	*Best/bound*
$N=10$, $G=2$	6.98	8.52	9.05	9.12	11.51	32.19	7.00	0.70	0.79
$N=12$, $G=3$	7.67	7.71	8.43	8.56	11.33	11.39	10.96	1.53	0.75
$N=12$, $G=4$	4.32	5.12	5.82	5.98	7.90	35.63	15.14	2.51	0.71
$N=60$, $G=10$	58.40	64.59	70.06	70.88	108.46	21.32	9.76	1.18	0.65
$N=360$, $G=6$	2960.92	3398.58	3436.55	3438.59	7283.20	16.13	1.18	0.06	0.47
				avg. % improvement		23.34	8.81	1.20	0.67

switches counts. The averages of these values for each set are reported in Table 3 for the 60 student, 10 group problem. (Note that the 100 arbitrary, initial partitions here are the same ones as those used for comparison with WJ earlier.) Summary results for all the problems are given in Table 4; WJ results from Table 2 are provided here as well for easy reference.

Solution times

Table 5 provides solution times for each method. The programs were coded in C and run on a non-dedicated dual processor Sun SparcStation-20. As expected, WJ best solution times increased significantly with problem size. WJ solution times are on the order of the other four methods. The four switching methods required comparable resources for the smaller problems; LC and LCW times tended to increase more than those for AL and ALW as the problem size increased. Perhaps most importantly, Table 5 indicates that solution times for all the methods are quite reasonable for practical sized problems. A more detailed analysis is provided in the results section below.

Combining methods

We decided to determine how the methods might work in sequence—that is, how applying one method to the best solution obtained by another method might improve the final result. We chose to start with the best solution provided by WJ (as it is the only approach unaffected by the starting order of the students) and apply each of the

Table 3 Heuristic rersults for 100 random starting solutions (sets of 60 students divided into 10 groups, each of size 6)

		Lotfi–Cerveny				*Lotfi–Cerveny–Weitz*			
Set no.	*Start value*	*Final value*	*% improvement*	*# cycles*	*# switches*	*Final value*	*% improvement*	*# cycles*	*# switches*
1	71.70	79.64	11.20	4.52	37.94	79.80	11.41	3.42	43.40
2	61.98	69.19	11.73	4.47	38.31	69.28	11.86	3.15	39.66
3	58.17	65.19	12.14	4.46	38.21	65.33	12.39	3.16	41.38
4	64.49	71.39	10.79	4.46	37.67	71.54	11.01	3.47	42.33
5	64.72	72.01	11.35	4.28	37.11	72.15	11.56	3.28	41.96
6	66.47	74.34	11.98	4.43	38.28	74.49	12.20	3.26	41.95
Mean	64.59	71.96	11.53	4.44	37.92	72.10	11.74	3.29	41.78
Std. dev.	4.52	4.87	0.51	0.08	0.47	4.88	0.51	0.13	1.24
		Arani–Lotfi				*Arani–Lotfi–Weitz*			
Set no.	*Start value*	*Final value*	*% improvement*	*# cycles*	*# switches*	*Final value*	*% improvement*	*# cycles*	*# switches*
1	71.70	79.51	11.01	4.90	11.79	79.70	11.27	7.01	17.90
2	61.98	68.97	11.37	4.64	11.54	69.18	11.70	6.70	17.44
3	58.17	65.06	11.93	4.71	11.71	65.23	12.21	6.77	17.90
4	64.49	71.27	10.61	4.44	10.85	71.43	10.85	6.64	16.66
5	64.72	71.93	11.22	4.90	12.24	72.06	11.43	6.89	17.47
6	66.47	74.16	11.71	4.56	11.60	74.38	12.04	7.02	18.30
Mean	64.59	71.82	11.31	4.69	11.62	72.00	11.58	6.84	17.61
Std. dev.	4.52	4.87	0.48	0.18	0.45	4.88	0.51	0.16	0.56

Table 4 Heuristic results for 100 random starting solutions-performance summary (all values are averaged over sets)

N = 10, G = 2 No. sets = 36
Initial average starting solution = 8.52

Method	*Final value*	*% improvement*	*# cycles*	*# switches*
LC	9.12	7.46	2.12	2.56
LCW	9.12	7.46	2.03	2.67
AL	9.04	6.42	2.15	1.92
ALW	9.07	6.84	2.42	2.33
WJ	9.12	7.00		

N = 12, G = 3 No. sets = 30
Initial average starting solution = 7.71

Method	*Final value*	*% improvement*	*# cycles*	*# switches*
LC	8.59	11.98	2.46	4.62
LCW	8.60	12.12	2.23	4.89
AL	8.51	10.98	2.46	2.90
ALW	8.56	11.54	3.10	3.86
WJ	8.56	10.96		

N = 12, G = 4 No. sets = 30
Initial average starting solution = 5.12

Method	*Final value*	*% improvement*	*# cycles*	*# switches*
LC	6.02	18.59	2.47	4.86
LCW	6.05	19.14	2.25	5.48
AL	5.97	17.58	2.47	2.71
ALW	6.02	18.55	3.08	3.60
WJ	5.98	15.14		

N = 60, G = 10 No. sets = 6
Initial average starting solution = 64.59

Method	*Final value*	*% improvement*	*# cycles*	*# switches*
LC	71.96	11.53	4.44	37.92
LCW	72.10	11.74	3.29	41.78
AL	71.82	11.31	4.69	11.62
ALW	72.00	11.58	6.84	17.61
WJ	70.88	9.76		

N = 360, G = 6 No. sets = 1
Initial average starting solution = 3398.58

Method	*Final value*	*% improvement*	*# cycles*	*# switches*
LC	3442.55	1.29	6.02	171.98
LCW	3442.58	1.30	3.73	172.14
AL	3441.39	1.26	7.42	81.49
ALW	3442.93	1.31	14.79	121.22
WJ	3438.59	1.18		

Overall results

Method	*% improvement*	*# cycles*	*# switches*
LC	10.17	3.50	44.39
LCW	10.35	2.71	45.39
AL	9.51	3.84	20.13
ALW	9.96	6.04	29.72
WJ	8.81		

other four methods to it. Summary results for all problems are provided in Table 6.

Results

Random starting solutions

For each method over all sets, (Table 4) we can see that the best performer was LCW, followed in order by LC, AL, ALW and WJ. Table 7 presents the relative rankings for each problem.

Starting with WJ

Overall rankings using the sequenced approach are provided in Table 8; again, we see that method LCW appears to be a consistent winner.

In viewing all the results, the following observations may be made.

Solution quality

- Overall, the best performer was method LCW, both for random starting solutions and for best WJ starting solutions.
- The difference between the best and the worst performing heuristic was on the order of 1% for random starting solutions, less for best WJ starting solutions.
- The heuristics outperformed the 100 random allocations on average by about 10%; they outperformed the 'worst' solutions obtained by minimising within group diversity by about 25%.
- The method currently in use at two business schools, WJ, outperformed the random allocations by on average 8.8%. WJ performed in the middle for smaller sized problems, and was poorest performer for the larger sized problems. Overall, 'best' WJ outperformed the average WJ by 1.25%.
- The 'final value' columns in Tables 4 and 6, indicate that there is a consistent though slight advantage in starting with the WJ approach. Sequencing the heuristic seems to provide greater improvement for the smaller problems.
- The LCW modification outperformed LC in all but one case, with no apparent significant computational penalty. All indications are that LCW is superior to LC.
- The ALW modification outperformed AL in all the random starting solution problems; conversely AL outperformed ALW in all the problems starting with the best WJ solution. Apparently the more limited search of AL is at a disadvantage starting with random partitions,

Table 5 Solution times (cpu seconds)

	WJ *best solution*		*WJ single solution (best/N)*	*Average solution times (time for 100 runs/100)*							
				LC		*LCW*		*AL*		*ALW*	
No students No. groups	*Time*	*% improvement*	*Time*	*Time*	*% improvement*	*Time*	*% improvement*	*Time*	*% improvement*	*Time*	*% improvement*
$N=10$, $G=2$	0.011	7.00	0.0011	0.040	7.46	0.038	7.46	0.031	6.42	0.043	6.84
$N=12$, $G=3$	0.017	10.96	0.0014	0.063	11.98	0.059	12.12	0.041	10.98	0.055	11.54
$N=12$, $G=4$	0.014	15.14	0.0012	0.056	18.59	0.058	19.14	0.042	17.58	0.059	18.55
$N=60$, $G=10$	2.562	9.76	0.0427	0.786	11.53	1.073	11.74	0.285	11.31	0.425	11.58
$N=360$, $G=6$	2875.740	1.18	7.9882	17.504	1.29	18.454	1.30	1.081	1.26	1.974	1.31
Averages:	575.669	8.81	1.6069	3.690	10.17	3.936	10.35	0.296	9.51	0.511	9.96

Note: Solution times and corresponding percent improvement from 100 random solutions for each method (from Table 4). Relative computation time and performance may be assessed by viewing across each row.

Table 6 Heuristic results starting with best Weitz–Jelassi solution-performance summary (all values are averaged over sets)

N = 10, G = 2 No. sets = 36
Initial (WJ) starting solution = 9.12

Method	*Final value*	*% improvement*	*# cycles*	*# switches*
LC	9.13	0.13	1.31	0.47
LCW	9.13	0.13	1.31	0.47
AL	9.12	0.03	1.14	0.19
ALW	9.12	0.01	1.06	0.08

N = 12, G = 3 No. sets = 30
Initial (WJ) starting solution = 8.56

Method	*Final value*	*% improvement*	*# cycles*	*# switches*
LC	8.60	0.44	1.80	1.27
LCW	8.61	0.56	1.83	1.30
AL	8.60	0.44	1.63	0.83
ALW	8.58	0.20	1.50	0.53

N = 12, G = 4 No. sets = 30
Initial (WJ) starting solution = 5.98

Method	*Final value*	*% Improvement*	*# cycles*	*# switches*
LC	6.04	1.05	2.00	1.60
LCW	6.05	1.28	1.93	2.10
AL	6.03	0.99	1.77	1.23
ALW	6.03	0.83	2.13	1.47

N = 60, G = 10 No. sets = 6
Initial (WJ) starting solution = 70.88

Method	*Final value*	*% improvement*	*# cycles*	*# switches*
LC	71.97	1.53	4.00	23.00
LCW	72.09	1.70	3.17	29.83
AL	72.03	1.61	3.17	11.00
ALW	71.99	1.57	6.17	14.67

N = 360, G = 6 No. sets = 1
Initial starting solution = 3438.59

Method	*Final value*	*% improvement*	*# cycles*	*# switches*
LC	3442.57	0.12	5	121
LCW	3442.57	0.12	3	108
AL	3442.26	0.12	6	87
ALW	3441.96	0.10	16	84

Overall results

Method	*% improvement*	*# cycles*	*# switches*
LC	0.65	2.82	29.47
LCW	0.76	2.25	28.34
AL	0.63	2.74	20.05
ALW	0.54	5.37	20.15

but is better at focusing the search when starting with a 'good' solution.

- The heuristics performed at approximately 80–50% of the bound on the optimal solution; performance degraded with respect to the bound as the problem size increased.
- The heuristics uniformly performed best (in terms of percentage improvement over random partitions), for 'middle-sized' problems, less good for the smallest problem, and worst for the largest problem. A set of runs on problems between the middle sized problems and the largest problem indicate that the performance drop-off appears monotonic over this range.

Computational resources

- In practical terms, solution times for all the methods were quite reasonable for any real-world application.
- Finding the best WJ solution was significantly more time consuming that the other methods. WJ best solution times were also the most sensitive to increases in problem size. Solution times for a single WJ solution were on the order of those of the other four heuristics.
- Times for LC, LCW, AL, and ALW were roughly comparable for the smaller problems. AL and ALW times remained relatively stable as the problem size increased; LC and LCW solution times showed large increases with problem size relative to AL and ALW.
- Solution times for LC and LCW were comparable; LCW consistently required fewer cycles but more switches than LC.
- Solution times for ALW were consistently greater than those for AL. This difference increases with problem size; for the largest problem the average solution time for ALW is almost twice that of AL (though for this problem

Table 7. Relative ranking of methods by problem (1 = best, 5 = worst). (Based on % improvement over 100 random starting solutions for each set)

Problem	*N = 10 G = 2*	*N = 12 G = 3*	*N = 12 G = 4*	*N = 60 G = 10*	*N = 360 G = 6*
Heuristic					
WJ	3	3	3	5	5
LC	1	2	2	3	3
LCW	2	1	1	1	2
AL	5	5	5	4	4
ALW	4	4	4	2	1

Table 8. Relative ranking of methods by problem (1 = best, 4 = worst). (Based on % improvement starting with the "best" WJ solution for each set)

Problem	*N = 10 G = 2*	*N = 12 G = 3*	*N = 12 G = 4*	*N = 60 G = 10*	*N = 360 G = 6*
Heuristic					
LC	1	2	2	4	1
LCW	1	1	1	1	2
AL	2	3	3	2	3
ALW	3	4	4	3	4

ALWs time is but a tenth of that for LC/LCW). For all random starting solution problems and most best starting solution problems, ALW required more cycles/switches than AL.

Conclusions and future research

This work addresses two related important themes in business and business schools today: expanding diversity in the workplace and the increasing reliance on teams as an organisation structure. This application has become particularly important in business schools, which have been expanding their focus on teamwork skills by incorporating more group work into their MBA programs; presumably this movement is a result of two recent corporate trends: increasing diversity of the workforce (partly due to internationalisation) and the current inclination of companies to organise themselves around work groups (Byrne and Bongiorno[21]). Our focus in this research was on studying different methods for creating maximally diverse groups and comparing them on the criteria of performance and computational complexity.

A principal conclusion of this work is that, of the five methods examined here, LCW is the consistent best performer. In particular, it is superior to LC, and this study suggests that replacing LC in final exam scheduling applications will result in improved solutions, with no increased computational burden. LCW also has the advantage of being a less complex method than LC.

Additionally, LCW provides an improvement of about 1% over WJ, a method currently in use for study group and block formation in at least two well-known MBA programs. We know that a difference of 9% (the average difference between WJ best and random solutions) has a drastic impact on group quality; it is not clear whether a 1% difference has practical implications for this application. (Differences in solutions will likely have different impacts in different domains—a 1% difference in a VLSI design problem may be more critical than in exam scheduling or group formation.) WJ maintains an advantage in terms of intuitiveness and simplicity. It also straightforwardly allows for a certain flexibility. For example the basic heuristic was easily modified at INSEAD to incorporate the option of requiring specified pairs of students to be in the same group, and forbidding others. Additionally, WJ does not presuppose equal groups sizes. (The mathematical program and the other heuristics considered here presume equal group sizes, though unequal group sizes may be accommodated by modifying the heuristics or by including 'dummy' students in the dataset.)

Lastly, the study also indicates that improved accuracy may be achieved through the sequential use of methods; in this case starting with WJ and then applying LCW gave the best overall results.

A caveat is in order however, as this study was performed on a single data set. Additional work should be undertaken using student data from other universities, and final exam data sets, before these results can be generalised with absolute confidence. Finally, our current work in this area is focused on the performance of (1) the graph theoretic methods drawn from VLSI design approaches to the problem (Weitz and Lakshminarayanan[22]) and (2) genetic algorithm based approaches.

References

1 Feo T and Khellaf M (1990). A class of bounded approximation algorithms for graph partitioning. *Networks* **20**: 181–195.
2 Lakshminarayanan S and Weitz R (1994). Creating groups of maximum diversity: contrasting math programming and heuristic approaches, working paper, Stillman School of Business, Seton Hall University, South Orange, NJ 07079 (revised January 1996).
3 Weitz RR and Jelassi MT (1992). Assigning students to groups: a multi-criteria decision support system approach. *Dec Sci* **23**: 746–757.
4 Mingers J and O'Brien FA (1995). Creating student groups with similar characteristics: a heuristic approach. *Omega* **23**: 313–321.
5 O.Brien FA and Mingers J (1995). The equitable partitioning problem: a heuristic algorithm applied to the allocation of university student accommodation. Warwick Business School Research Paper, no. 187.
6 Lotfi V and Cerveny R (1991). A final exam scheduling package. *J Opl Res Soc* **42**: 205–216.
7 Arani T and Lotfi V (1989). A three phased approach to final exam scheduling. *IIE Trans* **21**: 86–96.
8 Feo T, Goldschmidt O and Khellaf M (1992). One-half approximation algorithms for the *k*-partition problem. *Opns Res* **40** (Supp 1): S170–S173.
9 Weitz R and Lakshminarayanan S (1996). An empirical comparison of graph theoretic and heuristic methods for creating maximally diverse groups, working paper. Stillman School of Business, Seton Hall University, South Orange, NJ 07079, March.
10 Kuo CC, Glover F and Dhir KS (1993). Analyzing and modeling the maximum diversity problem by zero-one programming. *Dec Sci* **24**: 1171–1185.
11 Hill AV, Naumann JD and Chervany NL (1983). SCAT and SPAT: large-scale computer based optimization systems for the personnel assignment problem. *Dec Sci* **14**: 207–220.
12 Reeves GR and Hickman EP (1992). Assigning MBA students to field study project teams: a multicriteria approach. *Interfaces* **22**(5): 52–58.
13 Mijaji I, Ohno K and Mine H (1987). Solution method for partitioning students into groups. *Eur J Opl Res* **33**: 82–90.
14 Beheshtian-Ardekani M and Mahmood MA (1986). Development and validation of a tool for assigning students to groups for class projects. *Dec Sci* **17**: 92–113.
15 Muller T (1989). Assigning students to groups for class projects: an exploratory test of two methods. *Dec Sci* **20**: 623–634.
16 Donahue JM and Fox JB (1993). An investigation into the people-sequential heuristic method of group formation. *Dec Sci* **24**: 493–508.
17 Singh N (1993). Design of cellular manufacturing systems: an invited review. *Eur J Opl Res* **69**: 284–291.

18 Weitz R and Lakshminarayanan S (1996). On a heuristic for the final exam scheduling problem. *J Opl Res Soc* **47**: 599–600.

19 Carpaneto G and Toh P (1980). *ACM Trans Math Software* **6**: 104–111. (The program, ASSCT: a Fortran subroutine for solving the square assignment problem, is available via the internet at http://math.nist.gov/cgi-bin/gams-serve/list-module-components/TOMS/548/8587.html).

20 Weitz R and Lakshminarayanan S (1996). An empirical comparison of heuristic methods for creating maximally diverse groups, working paper. Stillman School of Business, Seton Hall Univerity, South Orange, NJ 07079, January.

21 Byrne JA and Bongiorno L (1994). The best B schools. *Bus Week* October 24.

22 Weitz R and Lakshminarayanan S (1997). An empirical comparison of heuristic and graph theoretic methods for creating maximally diverse groups, VLSI design, and exam scheduling. *Omega* **25**: 473–482.

Received February 1996;
accepted October 1997 after two revisions