J. Opl Res. Soc. Vol. 38, No. 9, pp. 785–795, 1987
Printed in Great Britain.

0160-5682/87 $3.00 + 0.00

Combinatorial Optimization Problems with Soft and Hard Requirements

HORST A. EISELT[1] and GILBERT LAPORTE[2]

[1]Faculty of Administration, University of New Brunswick and [2]GERAD, Ecole des Hautes Etudes Commerciales de Montréal, Canada

In this paper, the authors describe a heuristic principle for the solution of an important class of problems. Using real-world examples, it is then shown how some difficult combinatorial problems are modelled and solved by the suggested principle. In one case, a detailed discussion of the implementation of the solution is also provided.

Key words: heuristics, integer programming, multiobjective optimization, resource allocation, scheduling

INTRODUCTION

In this paper, we shall examine a class of difficult problems which frequently occur in practice. In particular, we refer to assignment problems (in the broad sense), i.e. problems where elements from one set are assigned to elements from a second set. Problems of this nature include the assignment of bus drivers to routes, assignments of students to course sections, or the assignment of a city to a salesman's route. There are various features which make this class of problems difficult to model and to solve. Here we would like to distinguish between mathematical difficulties and difficulties on the part of the decision-maker (DM) or the analyst. As an example, consider the aforementioned assignment of bus drivers to routes. In modelling the problem, the analyst will soon discover that the situation is by no means well defined. For instance, what constitutes a convenient schedule for a bus driver? What makes a schedule efficient? This *fuzziness* is largely the problem of the DM and the analyst. Furthermore, it is rare that there is only a single objective. Defining *multiple objectives* is again the DM's problem, whereas incorporating them into a solution procedure is also a mathematical difficulty. On the other hand, there are certain difficulties which are purely mathematical. These include the *combinatorial nature* of all these assignment problems. The *non-linearities* which occasionally occur fall into the same category. Finally, there is the fact that almost all practical problems—and the class discussed here is no exception—are of *large scale*. As an example, the models discussed in detail later in this paper include 200, 450, 7000 and 15,000 zero-one variables, respectively. Even if we restrict ourselves to deterministic models and exclude group decision-making, current knowledge prohibits the use of exact methods for the solution of such problems. Our heuristic principle is particularly suited for exactly these problems.

MODELLING THE REQUIREMENTS

Traditionally, decision models have two ingredients: one or more objective function(s) and a set of constraints that have to be satisfied. This distinction between objectives and constraints was, however, never as clear-cut as it sometimes appears. It formally vanished with the advent of goal programming and its target values, achievement levels and priority structures. But even before that, penalty and boundary methods have allowed the use of what were originally constraints in the objective function; the non-achievement of those constraints by a solution was penalized by very large costs. Early examples of such methods are the Big M method, attributed to Charnes (see Kolman and Beck[1]), and the sequential unconstrained minimization technique (SUMT—see Fiacco and McCormick[2]). Note that the principle of Lagrangean relaxation (see, for example, Fisher[3] and Geoffrion[4]) also belongs to this class of methods. It is obvious that in a practical problem, it is not always possible to label the various components uniquely as objectives or as constraints. In order to reflect this, the original 'crude' demands and needs that are to be modelled will be called

requirements. These requirements can formally be written as $g(\mathbf{x}) \leqslant 0$, where $\mathbf{x}$ is the vector of decision variables and $g(\mathbf{x})$ is some set of functions. The inequality signs of these relations are, however, misleading. Not only are there requirements which are very important for the decision-maker, but there is also no way to get around them; they must necessarily be satisfied. These requirements are termed 'hard'. Other requirements may be a little less crucial, even though their satisfaction is still important. Finally, at the other end of the scale, there are those requirements whose satisfaction is merely desirable. In other words, the '$\leqslant$' sign above may read anything from 'must be' via 'should really be' to 'would be nice if it were'. In order formally to divide the requirements (somewhat arbitrarily) into hard, medium and soft ones, $g(\mathbf{x})$ is subdivided into $[g_1(\mathbf{x}), g_2(\mathbf{x}), g_3(\mathbf{x})]$, reflecting the various degrees of strictness. In particular,

$g_1(\mathbf{x}) \leqslant 0$ are the hard (structural) requirements which have to be respected at all costs;
$g_2(\mathbf{x}) \leqslant 0$ are strict but nonetheless relaxable requirements; and
$g_3(\mathbf{x}) \leqslant 0$ are the soft (desirable) requirements.

As an example of this subdivision of requirements, consider a time-constrained travelling salesman problem. Then $g_1(\mathbf{x}) \leqslant 0$ includes a formulation prohibiting solutions in which the salesman must be present in two locations at the same time. This requirement cannot be relaxed, and hence it belongs in the hardest category. Consider now $g_2(\mathbf{x}) \leqslant 0$, the requirements of the second type. They will include a formulation which ensures 'sensible' visiting hours. For instance, a salesman's visit at 4:00 a.m. is highly undesirable but nevertheless possible, although at a high penalty cost. Finally, $g_3(\mathbf{x}) \leqslant 0$ consists of a formulation guaranteeing convenient working hours for the salesman. An appointment at, say, 8:00 a.m. a hundred miles away from where the salesman stayed overnight is possible, but there is a penalty associated with such a solution, even though it is not as steep as the ones in the preceding category.

Frequently the stricter requirements are quite well defined, whereas the softer requirements are more fuzzy. However, assigning a given requirement to one of the above three groups is a fuzzy problem in itself. As an example, if a certain course of action violates the law, it could be associated with $g_1(\mathbf{x})$ but it may also be seen as a part of $g_2(\mathbf{x})$, with the appropriate (large) penalty.

At this point, given requirements have to be cast into the mould of a mathematical programming problem. As a general principle, the hardest requirements (g_1) always appear as constraints, whereas the soft requirements (g_3) are modelled as objectives. The requirements of g_2 may be included in the constraints or in the objective, depending on the problem under study. Since the problem under consideration is assumed to be combinatorial in nature, this approach results in a discrete multiobjective programming problem. Formally, let g_C include those requirements which are so hard that they are considered as rigid constraints, whereas g_O are those requirements which are soft enough to be used as objectives. The resulting optimization problem can then be written as

$$(P) \text{ minimize } F = f(g_O(\mathbf{x})),$$

subject to

$$g_C(\mathbf{x}) \leqslant 0$$

$$\mathbf{x} \in \mathbf{X},$$

where $\mathbf{X}$ is usually a discrete, e.g. $\mathbf{X} = \{0, 1\}^n$.

ALTERNATIVE SOLUTION APPROACHES

Many different solution techniques for multiobjective programming problems have been devised. They can be categorized in various ways (see, for instance, Zimmermann[5] or Eiselt *et al.*[6]). Here we use and extend the latter classification scheme. In principle, the different approaches can be distinguished by the amount of specific information the DM is willing or able to specify prior to using any solution technique. There are two extreme cases: one in which the DM can specify exactly the value he associates with a certain objective, and the other in which he cannot say anything about the relations and trade-offs between the objectives. To formalize, let $f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_p(\mathbf{x})$ be the

objectives, with the criterion functions $f_k(\mathbf{x})$ being those in $g_O(\mathbf{x})$. Suppose now that the DM is able to specify weights $w_k (k = 1, \ldots, p)$, where w_k/w_l indicates the amount of the l-th objective he deems equivalent to a single unit of the k-th objective. Then the multiple objectives can be combined in the single objective

$$\sum_{k=1}^{p} w_k f_k(\mathbf{x}),$$

which is to be minimized, subject to the constraints $g_C(\mathbf{x}) \leqslant 0$, $\mathbf{x} \in \mathbf{X}$. Note that the problem is now a regular integer programming problem. On the other hand, if nothing is specified concerning the trade-offs between objectives, we are left with a vector optimization (VO) problem.

Various algorithms for continuous linear VO problems are known (see, for instance, Zeleny[7] and Cohon[8]). However, as far as discrete VO problems are concerned, recent reports such as those by Evans[9] or Bitran[10] indicate that only fairly small problems can be solved by exact methods.

In addition to this, the large number of non-dominated solutions which typically exist in VO problems usually requires a larger degree of interaction between the DM and the analyst who does the formal modelling. It is questionable whether decision-makers are willing or able to spend a considerable amount of time in the modelling and solution procedure.

Since specifying exact weights, as required by the first approach, is impossible for the type of problems under consideration and our problems are also discrete and of large scale, the above two approaches cannot be used. Another possibility would be to apply goal programming (see, for instance, Ignizio[11]). Here target values are specified, and then—on various pre-emptive priority levels—the weighted sums of the deviations of the requirements from the target values are minimized. It should be ensured that the optimal solution is non-dominated, which is not necessarily the case in general.[12] In a non-strictly algorithmic way, the same principle as that used in goal programming was applied by Lane[13] to the scheduling of the Canadian Football League: a hierarchy of requirements was constructed and then worked on 'top down'.

Finally there is the possibility of using fuzzy programming techniques. Work in this area has flourished over the last decade, as witnessed not only by the number of publications, but also by the fact that since 1978 there has been a journal, *Fuzzy Sets and Systems*, which deals exclusively with modelling and problem-solving in a fuzzy environment. Good surveys on these topics are those by Dubois and Prade,[14] Carlsson,[15] Zimmermann *et al.*[16] and Zimmermann.[17] At the core of fuzzy programming there is a set of fuzzy membership functions. Each of these functions is zero if the current solution falls short of a lower achievement level with respect to one of the given objectives, and it is one if an upper achievement level is reached. Between these two extreme cases, the fuzzy membership function is specified by the DM and the analyst. The general procedure is then to find a solution which maximizes the minimal value of any of the fuzzy membership functions. It can be shown[5] that solutions of fuzzy programming problems are non-inferior. But again, the selection of the fuzzy membership function is just as subjective as the selection of pre-emptive or non-pre-emptive weights in goal programming. It should be pointed out that all practical multiobjective programming approaches have in common that they 'satisfice' (for details of this concept, see Simon[18]).

Beyond the points just raised, another reason for us to use heuristic rather than exact methods for the solution of the problems described in this paper is that it makes little sense to employ a multiple-objective non-linear integer procedure (even if one were available) in the presence of fuzziness. In other words, why go through all the trouble of solving a multiobjective discrete programming problem if the structure and the parameters of the model are based on soft information?

THE HEURISTIC SOLUTION PRINCIPLE

All problems discussed in this paper can be viewed as assignment problems, where elements from a set I are assigned to elements of a set J. Each element $i \in I$ must be assigned to exactly one element $j \in J$, but many elements $i \in I$ may be assigned to the same $j \in J$. In other words, all cases discussed here are of the 'many-to-one' assignment type. In order to clarify this concept, consider the specific problems discussed below. In the case of political districting, the goal is to combine given

geographical sectors with constituencies. Formally, let I be the set of sectors and let J denote the set of constituencies. Then each sector is assigned to exactly one constituency, whereas a constituency can consist of any number of sectors. In exam scheduling, each element in I represents an exam, and each element in J stands for one time-slot. Again, each exam is assigned to exactly one time-slot, but different exams may be held at the same time. Finally, in nurse scheduling and course sectioning, the elements of I are nurses and students, respectively, and the elements of J are schedules but not individual time-slots. Then each student and nurse may be assigned to exactly one schedule, but each schedule may be shared by different nurses or students.

The heuristic procedure used in all problems discussed in this paper consists of two phases. In phase I we try to find a feasible solution, and in phase II the current feasible solution is improved in order to reach a local optimum. In particular, phase I is a greedy-type procedure, as follows. Initially, no assignments are made, and at any point during the algorithm, that assignment is introduced which—together with the assignments already in the solution—is feasible and which increases the objective function by the least amount. Many instances of such a procedure can be found in the literature. For example, the 'best fit' rule in bin packing,[19] the 'nearest neighbour' rule for the travelling salesman problem[20,21] and the algorithm developed by Minoux[22] for the optimal routeing of calls in a telephone network all belong to this class. If such a procedure terminates without having found a feasible solution, this does not, of course, mean that no feasible solution exists. One possibility is to re-arrange the assignments (e.g. by relaxing the requirement for the selection of the element to be assigned) or, if this still does not result in a feasible solution, to give the formulation back to the DM, who will then have to decide which constraints can be relaxed.

After a feasible solution has been found in phase I or by any other means, an attempt is made to improve the solution in the second phase of the procedure. In particular, any one assignment is temporarily removed from the current solution, thus leaving one element in J unassigned. We now try to assign the currently unassigned element j to some element in I, so that the assignment costs decrease. If such an assignment has been found, a new solution with lower cost is known and the procedure is repeated. Otherwise, if none of the assignments in a solution can be replaced by one with a lower objective-function value, a local optimum has been found. Popular examples of this method are the second phase of the primal simplex method and the 'pairwise exchange' method in the travelling salesman problem.[19] In summary, in a dynamic programming-like fashion, the first phase above constructs a solution element by element, while always 'keeping an eye on the objective'. In the second phase, we move from one feasible solution to an adjacent one, just as in the standard two-phase method of linear programming. The only difference is that since we do not work on convex sets, convergence towards a global optimum cannot be guaranteed. A local optimum will, however, always be found.

Instances of the above approach will be used in practical examples. In the next three sections of this paper, we briefly show how the above approach has been applied to model and to solve specific real-world problems. Then follows a more detailed example.

POLITICAL DISTRICTING

In Bourjolly *et al.*,[23] this problem is described as follows: "to divide the territory into suitably chosen basic units (or sectors) and then to group these together to form a set of constituencies which meet certain legal, political, geographical and demographic requirements". Here the requirements are:

g_1: each sector must be assigned to exactly one constituency;
g_2: (α) all constituencies must be connected;
(b) there should be no enclaves, i.e. no constituency should be completely surrounded by another constituency;
(γ) the magnitude of the population of each constituency must lie within certain bounds defined by law;
g_3: the constituencies must satisfy various geographic and demographic requirements: for example, natural boundaries such as large rivers, motorways, etc. will be respected whenever possible; all constituencies should have more or less the same population size;

they must be compact (a convenient compactness measure consists of taking the sum of Euclidean distances between the constituency centre of gravity and those of its sectors); constituencies should normally be homogeneous with respect to some socio-economic parameters such as annual income, etc.

Here is the solution approach developed by Kaiser,[24] Bourjolly *et al.*[23] and Laporte *et al.*[25]

Phase 0

Define an objective function F as the weighted sum of all criteria of $g_O(\mathbf{x}) = [g_{2\gamma}(\mathbf{x}), g_3(\mathbf{x})]$:

$$F = \sum_k w_k h_k(\mathbf{x}),$$

where each $h_k(\mathbf{x})$ is a positive valued function measuring, for criterion k, the deviation of the criterion function from its target value. For instance, in the case of $g_{2\gamma}$, this norm will be the ideal population size per constituency. The weight w_k is a positive constant chosen by the user. It is easy to weight the various criteria; here, for instance, the requirements associated with $g_{2\gamma}$ naturally carry considerable weight.

Phase 1

(i) Construct a gradual assignment of sectors to constituencies (thus satisfying g_1) by making at each step the selection which increases F the least and which satisfies $g_{2\alpha}$.

(ii) Eliminate enclaves (thus satisfying $g_{2\beta}$) created by including them in the surrounding constituency, and re-evaluate F.

Phase 2

(iii) Perform marginal improvements to the solution by elementary moves of sectors from one constituency to an adjacent one. If enclaves are created, they are treated as in (ii). Stop if a satisfactory solution has been attained.

A districting package (DECARTES) based on this method was developed and tested on the 1971 census data for the Island of Montreal. In the first tests, presented in Bourjolly *et al.*,[23] three criteria were considered under g_3: (i) small population size variance between the constituencies; (ii) compactness of the constituencies; (iii) socio-economic homogeneity (measured here by the annual income). Various combinations of weights for these three criteria were used. All solutions produced satisfied $g_1(\mathbf{x}) \leqslant 0$ and $g_2(\mathbf{x}) \leqslant 0$ and were also perfectly acceptable as far as g_3 was concerned. It was observed that assigning a large weight to compactness helped reduce the number of enclaves as a side effect. Also, in spite of attempts to obtain good socio-economic homogeneity by increasing the associated weight, the final solution always contained some constituencies made up of rich and poor sectors. This is due to the fact that on the Island of Montreal, there are several instances of poor areas adjoining wealthy ones.

The result of a districting study in 1981–82 for the Quebec Electoral Commission was ELECT, a graphical system (see Thalmann *et al.*[26]). This system was incorporated in DECARTES so that the maps produced could be drawn automatically on a screen or on paper. This enabled the Commission cartographers to test various weight combinations interactively in order to arrive at maps which were 'visually acceptable'. Using this graphical tool, tests were carried out in order to assess the significance of certain criteria such as age, schooling, ethnic origin, etc. For example, by allowing the weight assigned to a given criterion to vary between a very low value ($-M$) and a very large value (M), and by drawing the resulting maps, it could be 'seen' whether this criterion had any impact on the outcome.

NURSE SCHEDULING

In several North American hospitals, new nurse schedules are designed and implemented several times a year, for relatively short planning periods, lasting generally from a few weeks to a few months. These schedules have to meet certain strict conditions, be satisfactory to the nursing personnel involved and be such that the personnel needs of the hospital are met.

This problem has been addressed by several authors. Warner[27] described a two-phase method which fits very well into the framework described above: in the first phase, a feasible solution satisfying the staffing constraints is sought; in the second phase, this solution is modified in order to take staff preferences into account. Miller *et al.*[28] treated the same problem by considering two types of constraints: feasibility set constraints, which must always be satisfied, and non-binding schedule pattern constraints, whose violation incurs a high penalty cost. In his MSc. dissertation, Raymond[29] extended the work of Miller *et al.* in order to produce fairer schedules: employees who have been assigned an undesirable schedule at the beginning of the planning period should stand a better chance of obtaining a good schedule later. We now summarize Raymond's work.

The requirements can be described as follows:

g_1: every employee must be assigned exactly one schedule over the planning horizon;
g_2: (α) every working week must contain two days off;
(β) all employees should benefit from at least one week-end off out of two or three, depending on status;
g_3: (α) the number of employees on duty must be as close as possible to a target value (there exist five different target values, one for every category of nurses);
(β) nurses' preferences in the choice of their schedules should be met as far as possible.

The algorithm executes the following steps.

Phase 0

In this model, all requirements in g_1 and g_2 are considered as constraints, i.e. $g_C(\mathbf{x}) = [g_1(\mathbf{x}), g_2(\mathbf{x})]$. On the other hand, the objective function comprises all requirements in g_3. In particular, $F = w_1 g_{3\alpha}(\mathbf{x}) + w_2 g_{3\beta}(\mathbf{x})$. Generate all feasible two-week schedules and ask each employee to rank them.

Phase 1

For every two-week period, successively assign schedules to employees so as to increase F by the least amount. In order to be fair to employees, weights derived from the ranks specified by the nurses are updated by the program after every 14-day period.

This algorithm contains no interchange phase. It was tested on data provided by a Canadian hospital. The planning period consisted of six weeks, and there were 17 employees, belonging to five categories. Out of the 51 two-week schedules that were generated for these employees, all but three corresponded to the employee's first choice (the other three ranked 2, 2 and 5). Over the 42-day period, hospital targets were not met only five times, for only one category of employees.

EXAMINATION TIMETABLING

In this problem (see Laporte and Desroches[30] for a full description), the objective is to construct examination timetables for school or universities in such a way that: (i) there are no *conflicts*, i.e. students are never scheduled to take more than one examination at a time; (ii) the schedule does not extend over more than a prescribed number of days; (iii) room capacities are not exceeded; (iv) examinations are spread as evenly as possible in the schedule for most students; (v) some preferences as to the positioning of certain examinations are taken into account.

It is well known that the relaxed problem described by (i) and (ii) constitutes a graph-colouring problem (see, for example, Brelaz[31]). It is therefore not surprising that most solution strategies for examination timetabling are based on graph-colouring algorithms. However, the Laporte–Desroches approach, in the words of Carter,[32] "represents a dramatic departure from all the other algorithms".

Here is a summary of the hierarchy of requirements:

g_1: no conflicts are allowed;
g_2: (α) the time frame of the schedule must be respected;
(β) no room capacities may be exceeded;

g_3: (α) respect preferences for the positioning of certain examinations;
(β) spread examinations evenly for most students.

There are three main phases in the algorithm.

Phase 0

In this problem, the requirements in g_1 and g_2 were considered as constraints. The objective function F was defined as a sum of costs associated with $g_{3\alpha}$ and $g_{3\beta}$. These costs were defined in the following manner:

(*i*) *Position preferences.* Let i denote an examination taken by e_i students and let t be a period (time-slot) in the timetable. For every pair (i, t), the scheduler indicates whether or not he *wishes* examination i to be scheduled in period t. If so, set $w_{it} = 0$; otherwise, set w_{it} equal to some positive number (Laporte and Desroches[30] use $w_{it} = 0.75$). The solution to the examination timetabling problem can be represented by a binary matrix $\mathbf{x} = (x_{it})$, and the total cost of not respecting the scheduler's preferences is given by

$$F_1 = \sum_i \sum_t w_{it}\, x_{it}\, e_i.$$

(*ii*) *Even spread of examinations.* Consider two examinations i and j having c_{ij} students in common. If i and j are close to one another in the timetable, a high penalty should be generated. Let s be the number of periods separating i from j in the timetable. Then define weights w_s, taking large values for small s and small values for large s. (Laporte and Desroches[30] use $w_1 = 16$, $w_2 = 8$, $w_4 = 4$, $w_4 = 2$, $w_5 = 1$ and $w_s = 0$ for $s \geqslant 6$.) Then the total *proximity cost* is defined as

$$F_2 = \frac{1}{2} \sum_i \sum_j \sum_s \sum_t (x_{it} x_{j,t-s} + x_{it} x_{j,t+s})\, c_{ij} w_s.$$

Note that every student registered for an examination contributes towards F_1, but that only those students common to two given examinations are included in F_2.

Phase 1

All examinations are first placed in a waiting list. They are then gradually scheduled so as to create the smallest possible increase of $F = F_1 + F_2$. At some point, it may not be possible to schedule any more examinations without creating conflicts. Some examinations already scheduled may be bumped back into the waiting list. The process is pursued until all examinations have been scheduled. Considerable effort is devoted to the construction of this first feasible schedule. (If no schedule can be found—a very rare occurrence—the case is submitted to the user, who chooses which constraints to relax.)

Phase 2

As in the districting problem, the solution is then improved by simple moves of examinations in the schedule. This procedure is executed as long as F can be reduced.

In December 1981, this approach enabled Laporte and Desroches to produce the examination schedule at the Ecole des Hautes Etudes Commerciales de Montréal. There were then 168 examinations and 17,682 student-examinations. The program (HORHEC) was used to produce 10 alternative schedules in less than 30 sec on the University of Montreal CYBER 173 computer. HORHEC has since been implemented at the London School of Economics and Political Science in 1983 and at the University of Montreal Faculty of Arts and Science in 1985.

COURSE SECTIONING

This last application deals with the 'optimal' allocation of students to sections of a same course in a university. The problem is decribed in Laporte and Desroches,[33] and originates from the Ecole Polytechnique de Montréal (EPM), one of Canada's leading engineering schools, although it is by no means particular to this institution (see, for example, Abell[34]). Here is a brief description of the problem.

In some universities, complete course schedules are made available to students at the time of registration. Typically, these schedules, which are built up from enrolment forecasts, give the room number, instructor and time for each section of each course. At EPM, courses are offered in one of two campuses, located about two miles apart. Students use this information to select their courses. A computer program then allocates students to course sections in order to satisfy various constraints and criteria:

g_1: student course selections must be respected;
g_2: there should be no conflicts and no infeasible moves (i.e. two courses taken in succession in the two campuses);
g_3: (α) students should have 'pleasant schedules': a student should not have too many or too few classes on a given day; he should not have too many free periods between classes; his day should normally include a lunch break; the number of moves between the two campuses should be minimized, etc.;
(β) sections of a same course should be balanced (i.e. they should contain approximately the same number of students);
(γ) room capacities should be respected.

Satisfying the first requirement, i.e. respecting student course selection, was considered by EPM as critical, even if this meant the solution would contain conflicts. All other requirements were to be treated merely as objectives. It may seem surprising that the respect of room capacities should be treated only as a soft or desirable requirement: the reason is that at EPM, it is often possible to add extra chairs to overcrowded rooms.

The approach used in this case can be broken down into the following steps.

Phase 0

Here the constraints pose no problem since they simply state that every student must be assigned exactly one schedule.

The objective function F is the sum of three terms F_1, F_2 and F_3, measuring respectively the sum of costs associated with the selected student schedules, the total section disequilibrium and the total room overflow.

The cost associated with a student schedule is a sum of weights reflecting the various unpleasant aspects of the schedules. Some important defects, such as conflicts or infeasible moves, are weighted heavily, while other less important annoyances, such as the absence of a lunch break, receive a much smaller penalty. Here is a list of the penalties which were used:

	Situation	*Penalty*
(i)	Conflict	1000
(ii)	No lunch break between noon and 2:00 p.m.	30
(iii)	Each time interval between two classes	2
(iv)	Each free hour between two classes	2
(v)	Campus change during a free period	60
(vi)	Campus change during a class	1200
(vii)	Number of classes in a given day:	
	0 or 3 to 6	0
	2 or 7	6
	1 or 8	12
	9	18

F_1 is then set equal to the sum of all schedule costs over all students. To define F_2, let S_i be the set of sections associated with course i and y_{is} the number of students of course i registered in section s. The average number of students in sections of S_i is given by

$$\mu_i = \left(\sum_{s \in S_i} y_{is}\right) \Big/ |S_i|.$$

Then the total room disequilibrium can be measured by

$$F_2 = \sum_i \sum_{s \in S_i} (y_{is} - \mu_i)^2.$$

Finally, let C_r be the capacity of room r, and z_{rp} be the number of students assigned to room r in period p. Then the total room overflow is equal to

$$F_3 = \sum_r \sum_p \max(0, z_{rp} - C_r).$$

Phase 1

This phase is executed by temporarily assuming that the objective function consists only of F_1, i.e. section disequilibrium and room overflow are not considered at this stage. Then the problem simply reduces to assigning every student his or her best schedule. This is done by exploring, for every student, a branch and bound tree until one of the two following outcomes occurs: (i) all branches of the search tree have been fully explored, according to the usual branch and bound rules; (ii) 5000 schedules have been considered.

Phase 2

In order to arrive at a more satisfactory solution with respect to section disequilibrium and room overflow, it is now necessary to assign students schedules different from those they had been allocated in phase 1. This re-assignment may result in an increase in F_1, but this increase should be largely compensated by decreases in F_2 and in F_3. In order to accomplish this change with the maximum fairness possible to students, it is important that the increase in F_1 should not be realized at the expense of just a few students, but that it should be spread as equally as possible among them. This can be achieved by executing the following procedure, starting from the solution obtained at the end of phase 1.

Repeat this step for $t = 1, \ldots, 5$, using $\delta_1 = 5$, $\delta_2 = 10$, $\delta_3 = 20$, $\delta_4 = 70$, $\delta_5 = 400$.

Consider in turn all students i. If, at the beginning of iteration t, student i has a schedule of cost P_{it}, construct and assign to student i a new schedule of cost P'_{it} if the following conditions are satisfied:

(i) $P'_{it} \leqslant P_{it} + \delta_t$;
(ii) for $t \leqslant 4$, the new schedule must be such that F_2 decreases as a result; and
(iii) for $t = 5$, the new schedule must be such that F_3 decreases as a result.

The number of steps of this procedure and the corresponding values of δ_t were determined empirically after extensive testing.

At the time of writing, the program has been used six times at EPM, between January 1984 and September 1986. In order to illustrate some elements of the above discussion, we present the results obtained in January 1984. There were in total 2799 students, taking 326 courses offered in 518 sections. After elimination of students having selected courses containing only one section and merging of students having made identical course selections, there were 1818 students left.

In phase 1, the branch and bound algorithm examined an average of 500.34 schedules per student. In 1743 cases, fewer than 5000 schedules were examined. In the remaining 75 cases, the schedule may have been suboptimal. Therefore, an optimal schedule was found for at least 95.8% of all students. These schedules had an average cost of 719.21. Of all students, 34.5% had a schedule with a cost exceeding 1000, meaning that they probably had at least one conflict or one illegal move. This large percentage of 'infeasible' schedules may look high at first glance. However, the high proportion of optimal schedules indicates that the fault lies not with the algorithm but with the hand-made master schedule or with the students' selections of courses. A closer examination of this question revealed that around 75% of all conflicts were due to the fact that after their first or second year, many students are lagging behind in the normal curriculum and often have to take courses in two different years; the master schedule can hardly accommodate all possible situations. The remaining 25% of conflicts were caused by students wishing to take two optional courses overlapping in the master schedule. Here again, the master schedule cannot be

such that all selections of optional course are feasible for all students. As it turns out, about half the conflicts are tolerated by EPM; in all other cases, students are urged to make a different course selection. Given the situation that prevails at EPM, it is unlikely that all conflicts can be avoided. However, it is felt that a more satisfactory result could be achieved if an algorithm similar to the one described in this paper was used to construct the master schedule as well as optimal section assignment.

After the first iteration, the average section disequilibrium was equal to 19.67. After five extra iterations, this value was brought down to 2.69, whereas the average schedule cost increased to only 723.32 and still contained the same proportion of 'infeasible' schedules. After the last iteration, the average section disequilibrium was very low at 2.69, and only five sections out of 518 had an overflow exceeding 9 (the maximum overflow was equal to 26). The final solution was very satisfactory to the registrar's office and students alike.

CONCLUSION

A hierarchy of requirements is often present and can be indeed be dealt with in combinatorial problems. In this paper, a general framework for handling such problems was described. The four examples presented illustrate that although there is no universal algorithm for the problems, they have common principles, which may be summarized as follows:

(i) an objective function incorporating the fuzzy criteria is defined;
(ii) the *hard* constraints are satisfied through a step-by-step, build-up approach while 'keeping an eye on' the objective;
(iii) marginal modifications to the initial solution are then obtained.

This procedure results in a local minimum. From a practical point of view, it is preferable if several independent solutions with similar objective function values are generated so as to provide the decision-maker with some alternatives. Within these general guidelines, specific algorithms must be developed which take into account the particular structure of the problem to be tackled.

Acknowledgements—This work was supported by the Canadian Natural Sciences and Engineering Research Council (grants A5053 and A4747). The authors thank the referees for their valuable comments.

REFERENCES

1. B. Kolman and R. E. Beck (1980) *Elementary Linear Programming with Applications*. Academic Press, New York.
2. A. V. Fiacco and G. P. McCormick (1964) The sequential unconstrained minimization technique for nonlinear programming. A primal-dual method. *Mgmt Sci.* **10,** 360–366.
3. M. L. Fisher (1981) The Lagrangean relaxation method for solving integer programming problems. *Mgmt Sci.* **27,** 1–18.
4. A. M. Geoffrion (1974) Lagrangean relaxation and its uses in linear programming. *Mathemat. programmg Study* **2,** 82–114.
5. H. J. Zimmermann (1978) Fuzzy programming and L.P. with several objective functions. *Fuzzy Sets Systems* **1,** 45–55.
6. H. A. Eiselt, G. Pederzoli and C.-L. Sandblom (1987) *Continuous Optimization Models: Multiobjective, Nonlinear and Geometric Programming*. de Gruyter, Berlin.
7. M. Zeleny (1974) Linear multiobjective programming. In *Lecture Notes in Economics and Mathematical Systems*, Vol. 95. Springer, Berlin.
8. J. L. Cohon (1978) *Multiobjective Programming and Planning*. Academic Press, New York.
9. G. W. Evans (1984) An overview of techniques for solving multiobjective mathematical programs. *Mgmt Sci.* **30,** 1268–1282.
10. G. R. Bitran (1979) Theory and algorithms of linear multiple objective programs with zero-one variables. *Mathemat. Programmg* **17,** 362–390.
11. J. P. Ignizio (1982) *Linear Programming in Single- and Multiple-Objective Systems*. Prentice–Hall, Englewood Cliffs, N.J.
12. E. L. Hannan (1980) Nondominance in goal programming. *INFOR* **18,** 300–309.
13. D. E. Lane (1982) An algorithm for the construction of the regular schedule for the Canadian Football League. In *Recherche Opérationnelle/Management Science* (H. A. Eiselt, Ed.), Vol. 3, pp. 73–82. Administrative Sciences Association of Canada.
14. D. Dubois and H. Prade (1980) *Fuzzy Sets and Systems. Theory and Applications*. Academic Press, New York.
15. C. Carlsson (1981) Solving ill-structured problems through well-structured fuzzy programming. In *Operational Research '81* (J. P. Brans, Ed.). North-Holland, Amsterdam.
16. H. J. Zimmermann, L. A. Zadeh and B. R. Gaines (1984) Fuzzy sets and decision analysis. In *Studies in the Management Sciences*, Vol. 20. North-Holland, Amsterdam.

17. H. J. Zimmermann (1986) Fuzzy set theory—and its applications. In *Industrial Series in Management Science/Operations Research*. Kluwer-Nijhoff, Boston.
18. H. A. Simon (1964) On the concept of organization goal. *Admin. Sci. Q.* **9,** 1–22.
19. E. G. Coffman, M. R. Garey and D. S. Johnson (1984) Approximation algorithms for the bin-packing—an updated survey. In *Algorithm Design for Computer System Design* (G. Ausiello *et al.*, Eds). Springer, Vienna.
20. B. L. Golden and W. Stewart (1985) Empirical analysis of heuristics. In *The Traveling Salesman Problem: a Guided Tour of Combinatorial Optimization* (E. L. Lawler *et al.*, Eds), pp. 207–250. Wiley, Chichester.
21. M. Gondran and M. Minoux (1984) *Graphs and Algorithms*. Wiley, Chichester.
22. M. Minoux (1975) Résolution des problèmes de multiflots en nombres entiers dans les grands réseaux. *Rev. Franç. d'Automat. Inform. Recherche Opérat.* **3,** 21–40.
23. J.-M. Bourjolly, G. Laporte and J.-M. Rousseau (1981) Découpage électoral automatisé: application à l'Ile de Montréal. *INFOR* **19,** 113–124.
24. H. F. Kaiser (1967) An objective method for establishing legislative districts. *Midwest J. polit. Sci.* 200–213.
25. G. Laporte, J.-M. Rousseau and N. Thalmann (1983) A multicriteria approach to social and political districting. In *Recherche Opérationnellel Management Science* (C. Das, Ed.), Vol. 4, pp. 1–8. Administrative Sciences Association of Canada.
26. N. Thalmann, Y. Claude, G. Laporte and J.-M. Rousseau (1982) ELECT: an interactive graphical system for the automatic generation of electoral maps. *Cartographica* **19,** 28–40.
27. D. M. Warner (1976) Scheduling nursing personnel according to nursing preference: a mathematical programming approach. *Opns Res.* **24,** 842–856.
28. H. E. Miller, W. P. Pierskalla and G. J. Rath (1976) Nurse scheduling using mathematical programming. *Opns Res.* **24,** 857–870.
29. M.-P. Raymond (1978) Horaire du personnel infirmier. MSc. dissertation, Départment d'Informatique et de Recherche Opérationnelle, Université de Montréal.
30. G. Laporte and S. Desroches (1984) Examination timetabling by computer. *Comput. Opns Res.* **11,** 351–360.
31. D. Brelaz (1979) New methods to color the vertices of a graph. *CACM* **22,** 251–256.
32. M. W. Carter (1986) A survey of practical applications of examination timetabling algorithms. *Opns Res.* **34,** 193–202.
33. G. Laporte and S. Desroches (1986) The problem of assigning students to course sections in a large engineering school. *Comput. Opns Res.* **13,** 387–394.
34. V. A. Abell (1965) Purdue academic student scheduling. Mimeographed, Purdue University, Lafayette, Indiana.