J. Opl Res. Soc. Vol. 32, pp. 989 to 995, 1981
Printed in Great Britain. All rights reserved

0160-5682/81/110989-07$02.00/0
Copyright © 1981 Operational Research Society Ltd

An Algorithm for Solving 3-Dimensional Assignment Problems with Application to Scheduling a Teaching Practice

A. M. FRIEZE and J. YADEGAR
Department of Computer Science and Statistics, Queen Mary College, University of London

A scheduling problem associated with teaching practices at colleges of education is formulated as a 3-dimensional assignment problem. An efficient algorithm for its solution, based on Lagrangean relaxation, is described.

THE PROBLEM motivating the work described in this paper is the following: student teachers at colleges of education have periodically to undertake *teaching practices*. Each of a group of students will be assigned to one of a set of schools and his or her practices will be supervised by a tutor from the college.

Arranging which school each student is assigned to and which tutor will carry out a student's supervision is complicated by the fact that, quite naturally, students and tutors have requirements and preferences, and one should try to arrange things so that these are taken into account.

Formally suppose there are m students, n tutors and p schools. Let $I = \{1, \ldots m\}$, $J = \{1, \ldots n\}$, $K = \{1, \ldots p\}$. For student $i \in I$, tutor $j \in J$ and school $k \in K$ let v_{ijk} be a *satisfaction value* associated with an assignment of student i to school k under supervision by tutor j.

Tutor j is willing to supervise no more than t_j students for $j \in J$ and school $k \in K$ can have at most s_k students assigned to it.

A *schedule* is a pair y, z of integer m-vectors where $y_i \in J$ and $z_i \in K$ for $i \in I$. The triple (i, y_i, z_i) corresponds to student i being assigned to school z_i and supervised by tutor y_i. The schedule is *feasible*

if $$|\{i : y_i = j\}| \leqslant t_j \quad \text{for} \quad j \in J$$

and $$|\{i : z_i = k\}| \leqslant s_k \quad \text{for} \quad k \in K.$$

The *value* of a schedule is $\sum_{i \in I} v_{iy_iz_i}$. An important objective in arranging a teaching practice is to try and maximise the schedule value.

This problem can be formulated as an integer program in the following way:

Let the 0–1 variable x_{ijk}

$$i \in I, j \in J, k \in K$$

have the following significance:

$x_{ijk} = 1$ if student i is supervised at school k by tutor j—sometimes referred to as triple (i, j, k).
$= 0$ otherwise.

We then have the problem AP3:

$$\text{maximise} \quad \sum_{i \in I} \sum_{j \in J} \sum_{k \in K} v_{ijk} x_{ijk} \tag{1a}$$

subject to

$$\sum_{j\in J}\sum_{k\in K} x_{ijk} = 1 \qquad i \in I \tag{1b}$$

$$\sum_{i\in I}\sum_{k\in K} x_{ijk} \leqslant t_j \qquad j \in J \tag{1c}$$

$$\sum_{i\in I}\sum_{j\in J} x_{ijk} \leqslant s_k \qquad k \in K \tag{1d}$$

$$x_{ijk} = 0 \text{ or } 1. \qquad i \in I, j \in J, k \in K \tag{1e}$$

We use the notation $v(x)$ to denote the sum in (1a).

Constraint set (1b) says each student i occurs in exactly one triple. Constraint set (1c) says that each tutor j supervises no more than t_j students and constraint set (1d) says that each school k takes no more than s_k students.

AP3 is a *3-dimensional assignment* problem.

It is unfortunately an NP-hard problem—see Garey and Johnson[1] for a detailed analysis of this statement—and as such is unlikely to be solvable by an algorithm with a time bound which is polynomial in the size of the problem.

One is almost forced therefore to use a heuristic as in Frieze[2] or to try a branch and bound approach or possibly a cutting plane approach.

Very little material has been published on this problem. Branch and bound algorithms are described in Pierskalla[3] and Hansen and Kaufman,[4] where only small problems up to size $m = 8$ and $m = 16$ respectively were solved. (This latter problem took 674 secs on a CDC 6400.)

A branch and bound algorithm using Lagrange multipliers for constraints (1c) and (1d) was tested in Frieze[5] but the results were inferior to those proposed here. Burkard[6] also mentions the possibility of using Lagrangean relaxation on this problem.

We shall describe an approach using the idea of *Lagrangean relaxation* introduced by Held and Karp[7]—see also Fisher,[8] Geoffrion[9] or Shapiro.[10]

We relax constraints (1d) and use multipliers $u_k \geqslant 0$ for $k = 1, \ldots, p$ and consider the *dual function* $\phi : \mathscr{R}^p_+ \to \mathscr{R}$ defined by

$$\phi(u) = \max. \sum_{i\in I}\sum_{j\in J}\sum_{k\in K}(v_{ijk} - u_k)x_{ijk} + \sum_{k\in K} s_k u_k \tag{2}$$

subject to (1b), (1c), (1e).

If x^* is an optimum solution for AP3 it is easy to show that

$$\phi(u) \geqslant v(x^*) \qquad \text{for all } u \in \mathscr{R}^p_+.$$

The aim is to find a u that minimises or nearly minimises ϕ. This will generally give a good upper bound which can be used in the normal way in a branch and bound algorithm.

The procedure used to 'minimise' ϕ is the *sub-gradient algorithm*. We outline the bounding procedure before going into details.

PROCEDURE BOUND

Parameters: N = A limit to the number of iterations to be performed in computing the bound.

$\epsilon \geqslant 0$ = A tolerance, i.e. we will be satisfied with a feasible solution $\hat{x}$ satisfying $v(\hat{x}) \geqslant (1 - \epsilon)v(x^*)$

(a) (Initialisation)

$u = u_0$—starting value of u, $u_0 = 0$ seems remarkably good

$ub = \infty$—smallest upper bound computed so far

$lb = -\infty$—value of best solution to AP3 found so far

$t = 0$—iteration count

(b) $\quad t = t + 1$
$\quad$ if $t > N$ terminate bound.
(c) $\quad$ Compute $\phi(u)$, i.e. solve (2)
$\quad ub = \min(ub, \phi(u))$
(d) $\quad$ Use the solution to (2) to generate a solution $\hat{x}$.
$\quad lb = \max(lb, v(\hat{x}))$
$\quad$ If $lb \geqslant (1 - \epsilon)ub$ terminate bound.
(e) $\quad$ Compute a direction of search μ.
(f) $\quad$ Compute a step length $\sigma > 0$.
(g) $\quad u_k = \max(0, u_k + \sigma\mu_k) \qquad k \in K$
$\quad$ Go to (b).

Note that the algorithm will have the property that if $\mu = 0$ in step (e), then the current u minimises ϕ and bound can be terminated.

We next provide details for the various undefined steps.

COMPUTING ϕ

For each $i \in I$, $j \in J$ define $h_{ij} \in K$ and w_{ij} by

$$w_{ij} = v_{ijh_{ij}} - u_{h_{ij}} = \max_{k \in K} (v_{ijk} - u_k). \tag{3}$$

We argue next that

$$\phi(u) = \max \sum_{i \in I} \sum_{j \in J} w_{ij}\xi_{ij} + \sum_{k \in K} s_k u_k \tag{4a}$$

subject to

$$\sum_{j \in J} \xi_{ij} = 1 \qquad i \in I \tag{4b}$$

$$\sum_{i \in I} \xi_{ij} \leqslant t_j \qquad j \in J \tag{4c}$$

$$\xi_{ij} = 0 \text{ or } 1 \tag{4d}$$

This is proved formally as follows:

First let (x^*_{ijk}) be an optimum solution to (2). Let $\xi^*_{ij} = \sum_{k \in K} x^*_{ijk}$. (1b) and (1c) imply that (ξ^*_{ij}) satisfies (4b) and (4c). The integrality of (ξ^*_{ij}) plus (4b) implies (4d).

Also

$$\sum_{i \in I} \sum_{j \in J} \sum_{k \in K} (v_{ijk} - u_k)x^*_{ijk} \tag{5a}$$

$$\leqslant \sum_{i \in I} \sum_{j \in J} \sum_{k \in K} (v_{ijh_{ij}} - u_{h_{ij}})x^*_{ijk} \tag{5b}$$

$$= \sum_{i \in I} \sum_{j \in J} w_{ij}\xi^*_{ij} \tag{5c}$$

Suppose next that $(\hat{\xi}_{ij})$ is an optimum solution to (4). Define $(\hat{x}_{ijk})$ by

$$\hat{x}_{ijh_{ij}} = \hat{\xi}_{ij}$$
$$\hat{x}_{ijk} = 0 \text{ otherwise.}$$

Continuing the sequence of equations and inequalities (5) we get

$$\leqslant \sum_{i \in I} \sum_{j \in J} w_{ij}\hat{\xi}_{ij} \tag{5d}$$

$$= \sum_{i \in I} \sum_{j \in J} \sum_{k \in K} (v_{ijk} - u_k)\hat{x}_{ijk}. \tag{5e}$$

Now $(\hat{x}_{ijk})$ can be seen to satisfy (1b), (1c) and (1e). As (x^*_{ijk}) is optimal, this implies that the inequalities in (5) can be replaced by equations, which gives our result.

Problem (4) is relatively easy to solve. We used a primal-dual algorithm based on that of Ford and Fulkerson[11]. We had a program available for this algorithm and so we used it.

FINDING GOOD SOLUTIONS TO AP3

The vector (x_{ijk}^*) obtained from solving (2) does not necessarily satisfy (1d) but it does satisfy the remaining constraints of AP3.

The following decomposition of solutions to AP3 was noted in Frieze:[2] any 0–1 vector (x_{ijk}) satisfying (1b) can be decomposed into $x_{ijk} = \xi_{ij}\eta_{ik}$ where ξ, η satisfy (4b) and

$$\sum_{k\in K} \eta_{ik} = 1, \qquad \text{for } i \in I \tag{6a}$$

(1c) is then satisfied if and only if (4c) is, and (1d) is satisfied if and only if

$$\sum_{i\in I} \eta_{ik} \leq s_k, \qquad \text{for } k \in K \tag{6b}$$

In other words AP3 can be shown equivalent to the problem

$$\text{maximise} \qquad \sum_{i\in I}\sum_{j\in J}\sum_{k\in K} v_{ijk}\xi_{ij}\eta_{ik} \tag{7a}$$

subject to (4b), (4c), (6a), (6b):

$$\xi_{ij}, \eta_{ik} = 0 \text{ or } 1, \qquad i \in I, j \in J, k \in K \tag{7b}$$

(One can in fact drop the integrality requirement for ξ, η but it does not help very much.)

Having solved (4), we have (hopefully) a good value for ξ. Let this be ξ^*. We can construct the η that maximises (7) given $\xi = \xi^*$ as follows:

$$\text{Let } a_{ik} = \sum_{j\in J} v_{ijk}\xi^*_{ij} \qquad \text{for } i \in I, k \in K.$$

The η that maximises (7) under the constraint $\xi = \xi^*$ is one that maximises

$$\sum_{i\in I}\sum_{k\in K} a_{ik}\eta_{ik} \tag{8}$$

subject to (6a), (6b), (7b).

Thus solving 2 problems of the form given in (4) giving an upper bound and a good solution to AP3. Our experiments show that these are of high quality.

CHANGING u—THE SUBGRADIENT ALGORITHM

The function ϕ can be shown to be convex. A vector $v \in \mathscr{R}^p$ is said to be a *sub-gradient of* ϕ at u—written $v \in \partial\phi(u)$ if

$$\phi(u') - \phi(u) \geq v.(u' - u) \qquad \text{for all } u' \in \mathscr{R}^p \tag{9}$$

The sub-gradient generalises the idea of a gradient. (Note that ϕ is not generally differentiable everywhere.) The sub-gradient algorithm generalises the classical steepest descent algorithm—see Held, Wolfe and Crowder,[12] Poljak[13] or Camerini, Fratta and Maffioli.[14]

So in step (e) of our algorithm the direction taken μ is such that $-\mu \in \partial\phi(u)$. (Note that in (9) if $\phi(u') < \phi(u)$ and $-\mu \in \partial\phi(u)$, then

$$0 > \phi(u') - \phi(u) \geq -\mu\cdot(u' - u) \rightarrow \mu\cdot(u' - u) > 0.$$

and so $u' - u$ makes an acute angle with the direction μ chosen.)

COMPUTING μ

Finding a sub-gradient is fortunately a 'by-product' of the calculation of ϕ. In fact if (x^*_{ijk}) solves (2), then it is not difficult to show that $v = (v_1, \ldots v_p)$ where

$$v_k = s_k - \sum_{i \in I} \sum_{j \in J} x^*_{ijk} \qquad k \in K \tag{10}$$

is a sub-gradient of ϕ at u.

Having obtained a sub-gradient, one needs to decide on the step length σ. The following result is useful: suppose u^* minimises ϕ and $v \in \partial\phi(u)$. Then for $0 < \lambda < 2$ we have

$$\|u^* - (u - \sigma v)\| < \|u^* - u\| \qquad \text{Euclidean norm} \tag{11a}$$

if

$$\sigma < \lambda(\phi(u) - \phi(u^*))/\|v\|^2. \tag{11b}$$

As $\phi(u^*)$ is not known, we underestimate it by using lb. This works satisfactorily in practice. We used $\lambda = 1$ in our experiments.

If the procedure (a)–(g) described above fails to find a satisfactory solution after N steps, one branches, i.e. split the problem into a number of subproblems and apply the bounding procedure to each subproblem and so on.

Our code does not have this facility. We believe that we will usually get good enough solutions for teaching practice problems without branching.

Some experimental results and implementational details are discussed next.

COMPUTATIONAL CONSIDERATIONS

We have up to the present time only tackled one problem arising from a teaching practice. The rest of our experience is on randomly generated problems.

In a teaching practice problem the values v_{ijk} are generated in the following way: the input data consists of 3 matrices A, B, C. $A = \|a_{ij}\|$ is $m \times n$, $B = \|b_{ik}\|$ is $m \times p$ and $C = \|c_{jk}\|$ is $n \times p$. The matrix a_{ij} has the following significance:

$a_{ij} = 0$ if student i cannot be supervised by tutor j,
$\quad > 0$ the 'satisfaction level' is a_{ij}. In practice we allow $a_{ij} = 1$ or 2.

The values b_{ik} for student-school and c_{jk} for tutor-school are defined similarly. The values for v_{ijk} can then be defined as

$$\begin{aligned} v_{ijk} &= a_{ij} + b_{ik} + c_{kj} && \text{if} \quad a_{ij}, b_{ik}, c_{kj} \neq 0, \\ &= -\infty && \text{otherwise} \end{aligned} \tag{12}$$

i.e. the primary task is to find as large as possible a set of triples (i, y_i, z_i) such that $a_{iy_i} \neq 0$, $b_{iz_i} \neq 0$ and $c_{y_i z_i} \neq 0$. The value for ∞ in (12) was taken as $M = 6m + 1$. In practice we did not use (12) but

$$v_{ijk} := M + (v_{ijk} \text{ as defined in (12)}). \tag{13}$$

Using (13) we have that at any stage $\lfloor ub/m \rfloor$ is a valid upper bound to the number of students that can be found suitable assignments.

We found it useful having input the matrices A, B, C to generate a list of L of triples (i, j, k) for which $v_{ijk} \neq -\infty$ in (12). In the practical problem we tackled where $m = 57$, $n = 29$ and $p = 106$ only 2430 out of the 175218 possible triples were included in L. This speeded up the construction of the matrix $W = \|w_{ij}\|$ in (3) enormously. In fact construction of W at each stage was a bottleneck that had to be overcome if the method was to be practicable. Construction directly from (3) was initially taking up approximately 80% of the time for each iteration.

Now one pass through L is all that is needed to update W.

It may also be worth mentioning that we do not need to keep room for 0–1 vectors (x_{ijk}), (ξ_{ij}), (η_{ik}). We work throughout with corresponding integer m-vectors y, z.

RESULTS

In Tables 1 and 2 we summarise our computational experience with the method described previously. All problems were randomly generated except for the last problem in Table 1.

TABLE 1

Problem	m	n	p	nz	ub	lb	nit	t
1	10	10	7	19	7	7	9 (4)	4.1 (1.8)
2	20	25	15	100	18 (17)	17	10 (4)	8.9 (10.6)
3	30	30	30	861	30	30	3	14.9 (23.5)
4	30	30	30	861	30	30	1	13.5 (8.7)
5	50	50	40	2432	50	50	2	24.3 (55.6)
6	50	50	50	382	48	43 (42)	3 (2)	40.3 (85.2)
7	60	60	60	388	58	51	9	198.6 (677.7)
8	57	29	106	2430	48	48	1	22.0

m, n, p are as previously described.
nz = number of non-zero v_{ijk}; ub = final upper bound computed; lb = value of best solution found; nit = number of iterations of subgradient algorithm; t = c.p.u. time in seconds on an ICL 1905 computer; (quantity) refers to a version of the program that does not keep the non-zero v_{ijk}'s in a list—where significantly different.

The problems in Table 1 have the following form: given 3 0–1 matrices A, B, C we take

$$v_{ijk} = a_{ij} \times b_{ik} \times c_{kj} \text{ for } i \in I, j \in J, k \in K.$$

The most impressive result is that for the practical problem 8, where after one iteration the algorithm assigned the maximum possible number of students.

The quality of this result explains why we have not programmed a branch and bound phase. This would be justified however for problems 6 and 7.

In the problems of Table 2 we took the value $v_{ijk} = a_{ij} + b_{ik} + c_{jk}$ where a_{ij}, b_{ik}, c_{kj} were randomly generated integers from a uniform distribution on the interval [0–10].

TABLE 2

Problem	m	n	p	ub	lb	nit	t
1	10	10	7	198	198	4	2.0
2	20	25	15	98	98	17	35.7
3	30	30	30	210	208	1	8.0
4	30	30	30	210	210	1	8.0
5	50	50	40	400	398	3	74.0
6	50	50	50	650	649	8	225.6
7	50	50	50	1325	1302	5	219.0
8	50	50	50	400	400	1	32.0
9	50	50	50	600	599	8	224.0
10	64	64	64	1344	1337	5	321.0

For these problems ϵ = 0.02, step (d).

ACKNOWLEDGEMENT

We are grateful to Paul Rolfe and Charles Leedham-Green for bringing this problem to our attention in the first place. J. Yadegar's work was supported by the University of London Scholarship Fund.

REFERENCES

[1] M. R. GAREY and D. S. JOHNSON (1979) *Computers and intractibility: a Guide to NP-Completeness*. W. H. Freeman and Co. 1979.

[2] A. M. FRIEZE (1974) A bilinear programming formulation of the 3-dimensional assignment problem. *Math. Prog.* **7**, 376–379.

[3] W. P. PIERSKALLA (1968) The multidimensional assignment problem. *Opns Res.* **16**, 422–431.

[4] P. HANSEN and L. KAUFMAN (1973) A primal dual algorithm for the three-dimensional assignment problem. *Cah. Cent. Etud. Rech. Oper.* **15**, 327–336.

A. M. FRIEZE (1975) Studies in
R. E. BURKARD (1979) *Annals*
Holland, Amsterdam.
M. HELD and R. M. KARP (1
Math. Programm. **1**, 6–25.
M. L. FISHER (1978) Lagrange
ings of *Summer School in Com*
A. M. GEOFFRION (1974) Lagr
[10] J. F. SHAPIRO (1977) A survey
OR Centre, MIT.
[11] L. R. FORD and D. R. FULKER
[12] M. HELD, P. WOLFE and H. D
62–88.
[13] B. T. POLJAK (1967) A genera
[14] P. M. CAMERINI, L. FRATTA a
techniques. *Math. Programm.*

[5] A. M. Frieze (1975) Studies in integer programming. Ph.D. Thesis. The University of London.

[6] R. E. Burkard (1979) *Annals of Discrete Mathematics*. Vol. 4 (P. L. Hammer *et al*., Eds), p. 293. North Holland, Amsterdam.

[7] M. Held and R. M. Karp (1971) The travelling salesman problem and minimum spanning tree: Part II. *Math. Programm.* **1**, 6–25.

[8] M. L. Fisher (1978) Lagrangean relaxation methods for combinatorial optimisation. To appear in proceedings of *Summer School in Combinatorial Optimisation*. Urbino, Italy.

[9] A. M. Geoffrion (1974) Lagrangean relaxation for integer programming. *Math. Programm. Stud.* **2**, 82–114.

[10] J. F. Shapiro (1977) A survey of Lagrangean techniques for discrete optimization. Technical Report No. 133, OR Centre, MIT.

[11] L. R. Ford and D. R. Fulkerson (1962) *Flows in Networks*. Princeton Univ. Press, NJ.

[12] M. Held, P. Wolfe and H. D. Crowder (1974) Validation of sub-gradient optimization. *Math. Programm.* **6**, 62–88.

[13] B. T. Poljak (1967) A general method for solving extremum problems. *Soviet Math. Dokl.* **8**, 593–597.

[14] P. M. Camerini, L. Fratta and F. Maffioli (1975) On improving relaxation methods by modified gradient techniques. *Math. Programm. Stud.* **3**, 26–34.