Chapter 19
Operations Research

1 Scope of the Chapter

This chapter provides procedures for the numerical solution of certain integer programming and shortest path problems.

2 Available Modules

Module 19.1: `nag_ip` — **Integer Programming**

This module provides a procedure for solving

- 'zero-one', 'general', 'mixed' or 'all' integer linear programming problems using a branch and bound method.

Module 19.2: `nag_short_path` — **Shortest Path Problems**

This module provides a procedure for finding

- the shortest path through a directed or undirected acyclic network using Dijkstra's algorithm.

3 Background

3.1 Integer Programming Problems

General *linear programming* (LP) problems (see Dantzig [1]) are of the form:

find $x = (x_1, x_2, \ldots, x_n)^T$ to maximize $F(x) = \sum_{j=1}^{n} c_j x_j$

subject to linear constraints which may have the forms:

$$\begin{array}{lll} \sum_{j=1}^{n} a_{ij}x_j = b_i, & i = 1, 2, \ldots, m_1 & \text{(equality)} \\ \sum_{j=1}^{n} a_{ij}x_j \leq b_i, & i = m_1 + 1, \ldots, m_2 & \text{(inequality)} \\ \sum_{j=1}^{n} a_{ij}x_j \geq b_i, & i = m_2 + 1, \ldots, m & \text{(inequality)} \\ x_j \geq l_j, & j = 1, 2, \ldots, n & \text{(simple bound)} \\ x_j \leq u_j, & j = 1, 2, \ldots, n & \text{(simple bound).} \end{array}$$

This chapter is concerned with *integer programming* (IP) problems in which some (or all) of the elements of the vector x are further constrained to be *integers*. For general LP problems where x takes only real (i.e., non-integer) values, refer to Chapter 9. Note that IP problems may or may not have a solution, which may or may not be unique.

Consider for example the following problem:

$$\begin{array}{ll} \text{minimize} & 3x_1 + 2x_2 \\ \text{subject to} & 4x_1 + 2x_2 \geq 5, \\ & \phantom{4x_1 + {}} 2x_2 \leq 5, \\ & x_1 - x_2 \leq 2, \\ \text{and} & x_1 \geq 0,\, x_2 \geq 0. \end{array}$$

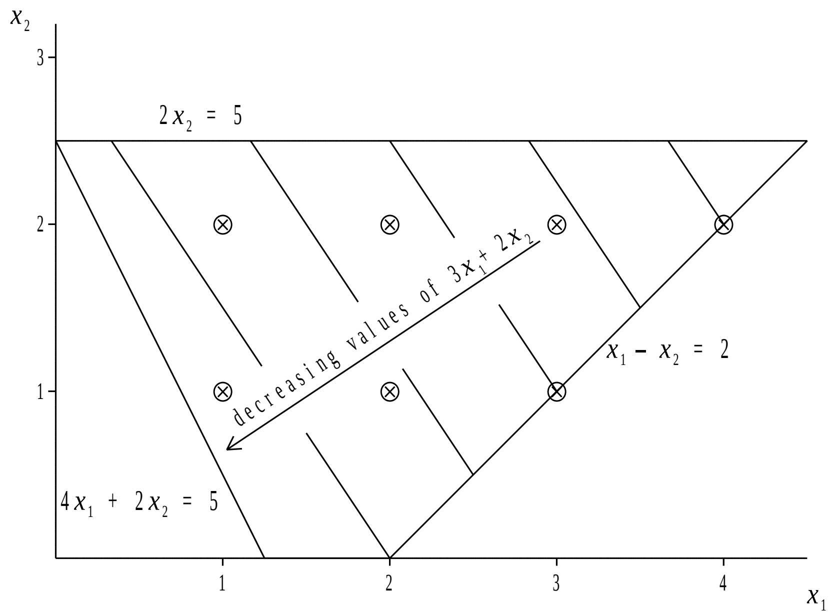

Figure 1.

The hatched area in Figure 1 is the *feasible region*, the region where all the constraints are satisfied, and the points within it which have integer co-ordinates are circled. The lines of hatching are in fact contours of decreasing values of the objective function $3x_1 + 2x_2$, and it is clear from Figure 1 that the optimum IP solution is at the point (1,1). For this problem the solution is unique.

However, there are other possible situations:

there may be more than one solution; e.g., if the objective function in the above problem were changed to $x_1 + x_2$, both (1,1) and (2,0) would be IP solutions;

the feasible region may contain no points with integer co-ordinates; e.g., if the bound constraint

$$3x_1 \leq 2$$

were added to the above problem;

there may be no feasible region; e.g., if the linear constraint

$$x_1 + x_2 \leq 1$$

were added to the above problem;

the objective function may have no finite minimum within the feasible region; this means that the feasible region is unbounded in the direction of decreasing values of the objective function, e.g., if the constraints

$$4x_1 + 2x_2 \geq 5, \quad x_1 \geq 0 \text{ and } x_2 \geq 0$$

were deleted from the above problem.

Algorithms for IP problems are usually based on algorithms for general LP problems, together with some procedure for constructing additional constraints which exclude non-integer solutions (see Beale [2]).

The Branch and Bound (B&B) method is a well-known and widely used technique for solving IP problems (see Beale [2] or Mitra [3]). It involves subdividing the optimum solution to the original LP problem into two mutually exclusive sub-problems by branching an integer variable that currently has a fractional optimal value. Each sub-problem can now be solved as an LP problem, using the objective function of the original problem. The process of branching continues until a solution for one of the sub-problems is feasible with respect to the integer problem. In order to prove the optimality of this solution, the rest of the sub-problems in the B&B tree must also be solved. Naturally, if a better integer feasible solution is found for any sub-problem, it should replace the one at hand. Computational efficiency is enhanced by discarding inferior sub-problems. These are problems in the B&B search tree whose LP solutions are lower than (in the case of maximization) the best integer solution at hand.

3.2 Shortest Path Problems

The *shortest path* problem is that of finding a path of minimum length between two distinct vertices n_s and n_e through a network. Suppose the vertices in the network are labelled by the integers $1, 2, \ldots, n$. Let (i, j) denote an ordered pair of vertices in the network (where i is the origin vertex and j the destination vertex of the arc), x_{ij} the amount of flow in arc (i, j) and d_{ij} the length of the arc (i, j). The LP formulation of the problem is thus given as

$$\text{minimize} \quad \sum \sum d_{ij} x_{ij} \text{ subject to } Ax = b, \;\; 0 \leq x \leq 1, \qquad (1)$$

where

$$a_{ij} = \begin{cases} +1 & \text{if arc } j \text{ is directed away from vertex } i, \\ -1 & \text{if arc } j \text{ is directed towards vertex } i, \\ 0 & \text{otherwise} \end{cases}$$

and

$$b_i = \begin{cases} +1 & \text{for } i = n_s, \\ -1 & \text{for } i = n_e, \\ 0 & \text{otherwise.} \end{cases}$$

The above formulation only yields a meaningful solution if $x_{ij} = 0$ or 1; i.e., arc (i, j) forms part of the shortest route only if $x_{ij} = 1$. In fact since the optimal LP solution will (in theory) always yield $x_{ij} = 0$ or 1, (1) can also be solved as an IP problem. Note that the problem may also be solved directly (and more efficiently) using a variant of Dijkstra's algorithm (see Ahuja *et al* [4]).

4 References

[1] Dantzig G B (1963) *Linear Programming and Extensions* Princeton University Press

[2] Beale E M (1977) Integer Programming *The State of the Art in Numerical Analysis* (ed D A H Jacobs) Academic Press

[3] Mitra G (1973) Investigation of some branch and bound strategies for the solution of mixed integer linear programs *Math. Programming* **4** 155–170

[4] Ahuja R K, Magnanti T L and Orhin J B (1993) *Network Flows: Theory, Algorithms and Applications* Prentice-Hall