3

Well-Solved Problems

3.1 PROPERTIES OF EASY PROBLEMS

Here we plan to study some integer and combinatorial optimization problems that are "well-solved" in the sense that an "efficient" algorithm is known for solving all instances of the problem. Clearly an instance with 1000 variables or data values ranging up to 10^{20} can be expected to take longer than an instance with 10 variables and integer data never exceeding 100. So we need to define what we mean by efficient.

For the moment we will be very imprecise and say that an algorithm on a graph $G = (V, E)$ with n nodes and m edges is *efficient* if, in the worst case, the algorithm requires $0(m^p)$ elementary calculations (such as additions, divisions, comparisons, etc) for some integer p, where we assume that $m \geq n$.

In considering the COP $\max\{cx : x \in X \subseteq R^n\}$, it is not just of interest to find a dual problem, but also to consider a related problem, called the separation problem .

Definition 3.1 The *Separation Problem* associated with COP is the problem: Given $x^* \in R^n$, is $x^* \in conv(X)$? If not, find an inequality $\pi x \leq \pi_0$ satisfied by all points in X, but violated by the point x^*.

Now, in examining a problem to see if it has an efficient algorithm, we will see that the following four properties often go together:

(i) *Efficient Optimization Property:* For a given class of optimization problems (P) $\max\{cx : x \in X \subseteq R^n\}$, there exists an efficient (polynomial) algorithm.

(ii) *Strong Dual Property:* For the given problem class, there exists a strong dual problem (D) $\min\{\omega(u) : u \in U\}$ allowing us to obtain optimality conditions that can be quickly verified:
$x^* \in X$ is optimal in P if and only if there exists $u^* \in U$ with $cx^* = \omega(u^*)$.

(iii) *Efficient Separation Property:* There exists an efficient algorithm for the separation problem associated with the problem class.

(iv) *Explicit Convex Hull Property:* A compact description of the convex hull $conv(X)$ is known, which in principle allows us to replace every instance by the linear program: $\max\{cx : x \in conv(X)\}$.

Note that if a problem has the Explicit Convex Hull Property, then the dual of the linear program $\max\{cx : x \in conv(X)\}$ suggests that the Strong Dual Property should hold, and also using the description of conv(X), there is some likelihood that the Efficient Separation Property holds. So some ties between the four properties are not surprising. The precise relationship will be discussed later. In the next sections we examine several classes of problems for which we will see that typically all four properties hold.

3.2 IPS WITH TOTALLY UNIMODULAR MATRICES

A natural starting point in solving integer programs :

$$(IP) \qquad \max\{cx : Ax \le b, x \in Z^n_+\}$$

with integral data (A, b) is to ask when one will be lucky, and the linear programming relaxation (LP) $\max\{cx : Ax \le b, x \in R^n_+\}$ will have an optimal solution that is integral.

From linear programming theory, we know that basic feasible solutions take the form: $x = (x_B, x_N) = (B^{-1}b, 0)$ where B is an $m \times m$ nonsingular submatrix of (A, I) and I is an $m \times m$ identity matrix.

Observation 3.1 (Sufficient Condition) If the optimal basis B has $det(B) = \pm 1$, then the linear programming relaxation solves IP.

Proof. From Cramer's rule, $B^{-1} = B^*/det(B)$ where B^* is the adjoint matrix. The entries of B^* are all products of terms of B. Thus B^* is an integral matrix, and as $det(B) = \pm 1$, B^{-1} is also integral. Thus $B^{-1}b$ is integral for all integral b. ∎

The next step is to ask when one will always be lucky. When do all bases or all optimal bases satisfy $det(B) = \pm 1$?

Definition 3.2 A matrix A is *totally unimodular (TU)* if every square sub-

$$\begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix} \qquad \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix}$$

Table 3.1 Matrices that are not TU

$$\begin{pmatrix} 1 & -1 & -1 & 0 \\ -1 & 0 & 0 & 1 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \qquad \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Table 3.2 Matrices that are TU

First we consider whether such matrices exist and how we can recognize them. Some simple observations follow directly from the definition.

Observation 3.2 If A is TU, $a_{ij} \in \{+1, -1, 0\}$ for all i, j.

Observation 3.3 The matrices in Table 3.1 are not TU. The matrices in Table 3.2 are TU.

Proposition 3.1 *A matrix A is TU if and only if*
(i) the transpose matrix A^T is TU if and only if
(ii) the matrix (A, I) is TU.

There is a simple and important sufficient condition for total unimodularity, that can be used to show that the first matrix in Table 3.2 is TU.

Proposition 3.2 *(Sufficient Condition). A matrix A is TU if*
(i) $a_{ij} \in \{+1, -1, 0\}$ for all i, j.
(ii) Each column contains at most two nonzero coefficients ($\sum_{i=1}^m |a_{ij}| \le 2$).
(iii) There exists a partition (M_1, M_2) of the set M of rows such that each column j containing two nonzero coefficients satisfies $\sum_{i \in M_1} a_{ij} - \sum_{i \in M_2} a_{ij} = 0$.

Proof. Assume that A is not TU, and let B be the smallest square submatrix of A for which $\det(A) \notin \{0, 1, -1\}$. B cannot contain a column with a single nonzero entry, as otherwise B would not be minimal. So B contains two nonzero entries in each column. Now by condition (iii), adding the rows in M_1 and subtracting the rows in M_2 gives the zero vector, and so $\det(B) = 0$, and we have a contradiction. ∎

Note that condition (iii) means that if the nonzeros are in rows i and k, and

and $k \in M_2$, or vice versa. This leads to a simple algorithm to test whether the conditions of Proposition 3.2 hold. In the next section we will see an important class of matrices arising from network flow problems that satisfy this sufficient condition.

Now returning to IP, it is clear that when A is TU, the linear programming relaxation solves IP. In some sense the converse holds.

Proposition 3.3 *The linear program* $\max\{cx : Ax \leq b, x \in R^n_+\}$ *has an integral optimal solution for all integer vectors b for which it has a finite optimal value if and only if A is totally unimodular.*

On the question of efficient algorithms, we have essentially proved that for the IP: $\max\{cx : Ax \leq b, x \in Z^n_+\}$ with A totally unimodular:

(a) The Strong Dual Property holds: the linear program $(D) : \min\{ub : uA \geq c, u \geq 0\}$ is a strong dual.
(b) The Explicit Convex Hull Property holds: the convex hull of the set of feasible solutions $conv(X) = \{Ax \leq b, x \geq 0\}$ is known.
(c) The Efficient Separation Property holds: the separation problem is easy as it suffices to check if $Ax^* \leq b$ and $x^* \geq 0$.

Given that these three properties hold, we have suggested that the Efficient Optimization Property should also hold, so there should be an efficient algorithm for IP. This turns out to be true, but it is a nontrivial result beyond the scope of this text. This is in turn related to the fact that efficient algorithms to recognize whether a matrix A is TU are also nontrivial.

3.3 MINIMUM COST NETWORK FLOWS

Here we consider an important class of problems with many applications lying at the frontier between linear and integer programming.

Given a digraph $D = (V, A)$ with arc capacities h_{ij} for all $(i, j) \in A$, demands b_i (positive inflows or negative outflows) at each node $i \in V$, and unit flow costs c_{ij} for all $(i, j) \in A$, the minimum cost network flow problem is to find a feasible flow that satisfies all the demands at minimum cost. This has the formulation:

$$\min \sum_{(i,j)\in A} c_{ij}x_{ij} \quad (3.1)$$

$$\sum_{k\in V^+(i)} x_{ik} - \sum_{k\in V^-(i)} x_{ki} = b_i \text{ for } i \in V \quad (3.2)$$

$$0 \leq x_{ij} \leq h_{ij} \quad \text{for } (i,j) \in A \quad (3.3)$$

where x_{ij} denotes the flow in arc (i, j), $V^+(i) = \{k : (i, k) \in A\}$ and $V^-(i) = \{k : (k, i) \in A\}$.

It is evident that for the problem to be feasible the total sum of all the demands must be zero (i.e., $\sum_{i\in V} b_i = 0$).

Example 3.1 The digraph in Figure 3.1 leads to the following set of balance

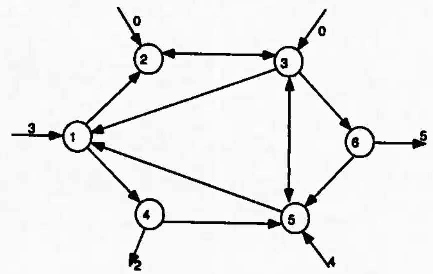

Fig. 3.1 Digraph for minimum cost network flow

equations:

x_{12}	x_{14}	x_{23}	x_{31}	x_{32}	x_{35}	x_{36}	x_{45}	x_{51}	x_{53}	x_{65}		
1	1	0	−1	0	0	0	0	−1	0	0	=	3
−1	0	1	0	−1	0	0	0	0	0	0	=	0
0	0	−1	1	1	1	1	0	0	−1	0	=	0
0	−1	0	0	0	0	0	1	0	0	0	=	−2
0	0	0	0	0	−1	0	−1	1	1	−1	=	4
0	0	0	0	0	0	−1	0	0	0	1	=	−5

The additional constraints are the bound constraints: $0 \leq x_{ij} \leq h_{ij}$. ■

Proposition 3.4 *The constraint matrix A arising in a minimum cost network flow problem is totally unimodular.*

Proof. The matrix A is of the form $\begin{pmatrix} C \\ I \end{pmatrix}$ where C comes from the flow conservation constraints, and I from the upper bound constraints. Therefore it suffices to show that C is TU. The sufficient conditions of Proposition 3.2 are satisfied with $M_1 = M$ and $M_2 = \phi$. ■

Corollary In a minimum cost network flow problem, if the demands $\{b_i\}$ and the capacities $\{h_{ij}\}$ are integral,
(i) Each extreme point is integral.
(ii) The constraints (3.2)-(3.3) describe the convex hull of the integral feasible flows.