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 :

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

with integral data (A, b) is to ask when one will be lucky, and the linear programming relaxation (LP) $\max\{cx : Ax \leq b, x \in R^n_+\}$ 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 submatrix of A has determinant $+1, -1$ or 0.

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

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 ($\sum_{i=1}^{m} |a_{ij}| \leq 2$).
(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 if $a_{ij} = -a_{kj}$, then $\{i, k\} \in M_1$ or $\{i, k\} \in M_2$, whereas if $a_{ij} = a_{kj}$, $i \in M_1$

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:

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

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

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

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

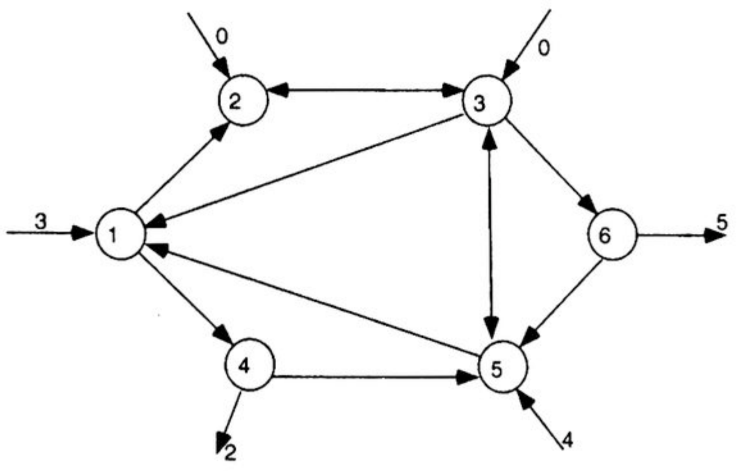

Fig. 3.1 Digraph for minimum cost network flow

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: $0 \le x_{ij} \le h_{ij}$. ■

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.

3.4 SPECIAL MINIMUM COST FLOWS

The Shortest Path Problem. Given a digraph $D = (V, A)$, two distinguished nodes $s, t \in V$, and nonnegative arc costs c_{ij} for $(i,j) \in A$, find a minimum cost $s - t$ path.

The Max Flow Problem. Given a digraph $D = (V, A)$, two distinguished nodes $s, t \in V$, and nonnegative capacities h_{ij} for $(i,j) \in A$, find a maximum flow from s to t.

Both these problems are special cases of the minimum cost network flow problem so we can use total unimodularity to analyze them. However, the reader has probably already seen combinatorial polynomial algorithms for these two problems in a course on network flows.

What are the associated dual problems?

3.4.1 Shortest Path

Observe first that the shortest path problem can be formulated as:

$$z = \min \sum_{(i,j)\in A} c_{ij}x_{ij} \tag{3.4}$$

$$\sum_{k\in V^+(i)} x_{ik} - \sum_{k\in V^-(i)} x_{ki} = 1 \text{ for } i = s \tag{3.5}$$

$$\sum_{k\in V^+(i)} x_{ik} - \sum_{k\in V^-(i)} x_{ki} = 0 \text{ for } i \in V \setminus \{s,t\} \tag{3.6}$$

$$\sum_{k\in V^+(i)} x_{ik} - \sum_{k\in V^-(i)} x_{ki} = -1 \text{ for } i = t \tag{3.7}$$

$$x_{ij} \geq 0 \text{ for } (i,j) \in A \tag{3.8}$$

$$x \in Z^{|A|} \tag{3.9}$$

where $x_{ij} = 1$ if arc (i,j) is in the minimum cost (shortest) $s - t$ path.

Theorem 3.5 *z is the length of a shortest $s - t$ path if and only if there exist values π_i for $i \in V$ such that $\pi_s = 0, \pi_t = z$, and $\pi_j - \pi_i \leq c_{ij}$ for $(i,j) \in A$.*

Proof. The linear programming dual of (3.4)–(3.8) is precisely

$$w^{LP} = \max \pi_t - \pi_s$$
$$\pi_j - \pi_i \leq c_{ij} \text{ for } (i,j) \in A.$$

Replacing π_j by $\pi_j + \alpha$ for all $j \in V$ does not change the dual, so we can fix $\pi_s = 0$ without loss of generalty. As the primal matrix is totally unimodular, strong duality holds and the claim follows. ∎

We note that one particular dual solution is obtained by taking π_i to be the cost of a shortest path from s to i.

3.4.2 Maximum $s - t$ Flow

Adding a backward arc from t to s, the maximum $s - t$ flow problem can be formulated as:

$$\begin{gathered}\max x_{ts}\\ \textstyle\sum_{k\in V^+(i)} x_{ik} - \sum_{k\in V^-(i)} x_{ki} = 0 \text{ for } i \in V\\ 0 \le x_{ij} \le h_{ij} \text{ for } (i,j) \in A.\end{gathered}$$

The dual is:

$$\begin{gathered}\min \textstyle\sum_{(i,j)\in A} h_{ij}w_{ij}\\ u_i - u_j + w_{ij} \ge 0 \text{ for } (i,j) \in A\\ u_t - u_s \ge 1.\end{gathered}$$

From total unimodularity, an optimal solution is integer. Also as the dual is unchanged if we replace u_j by $u_j + \alpha$ for all $j \in V$, we can set $u_s = 0$. Given such a solution, let $X = \{j \in V : u_j \le 0\}$ and $\bar{X} = V \setminus X = \{j \in V : u_j \ge 1\}$. Now

$$\sum_{(i,j)\in A} h_{ij}w_{ij} \ge \sum_{(i,j)\in A, i\in X, j\in\bar{X}} h_{ij}w_{ij} \ge \sum_{(i,j)\in A, i\in X, j\in\bar{X}} h_{ij}$$

as $w_{ij} \ge u_j - u_i \ge 1$ for $(i,j) \in A$ with $i \in X$ and $j \in \bar{X}$.

However, this lower bound $\sum_{(i,j)\in A, i\in X, j\in\bar{X}} h_{ij}$ is attained by the solution $u_j = 0$ for $j \in X$, $u_j = 1$ for $j \in \bar{X}, w_{ij} = 1$ for $(i,j) \in A$ with $i \in X$ and $j \in \bar{X}$, and $w_{ij} = 0$ otherwise. So there is an optimal 0-1 solution.

We see that $s \in X, t \in \bar{X}$, $\{(i,j) : w_{ij} = 1\}$ is the set of arcs of the $s - t$ cut $(X, V \setminus X)$, and we obtain the standard result that the maximum value of an $s - t$ flow equals the minimum capacity of an $s - t$ cut.

Theorem 3.6 *A strong dual to the max $s - t$ flow problem is the minimum $s - t$ cut problem:*

$$\min_X\{\sum_{(i,j)\in A: i\in X, j\notin X} h_{ij} : s \in X \subset V \setminus \{t\}\}.$$

3.5 OPTIMAL TREES

Definition 3.3 Given a graph $G = (V, E)$, a *forest* is a subgraph $G' = (V, E')$ containing no cycles.

Definition 3.4 Given a graph $G = (V, E)$, a *tree* is a subgraph $G' = (V, E')$ that is a forest and is *connected* (contains a path between every pair of nodes of V).

Some well-known consequences of these definitions are listed below:

Proposition 3.7 *A graph $G = (V, E)$ is a tree if and only if*
(i) it is a forest containing exactly $n - 1$ edges, if and only if
(ii) it is an edge-minimal connected graph spanning V, if and only if
(iii) it contains a unique path between every pair of nodes of V, if and only if
(iv) the addition of an edge not in E creates a unique cycle.

The Maximum Weight Forest (Tree) Problem. Given a graph $G = (V, E)$ and edge weights c_e for $e \in E$, find a maximum weight subgraph that is a forest (tree).

This problem arises naturally in many telecommunications and computer network applications where it is necessary that there is at least one path between each pair of nodes. When one wishes to minimize installation costs, the optimal solution is clearly a minimum weight tree.

Remember from the previous chapter that the idea of a *greedy algorithm* is to take the best element and run. It is very shortsighted. It just chooses one after the other whichever element gives the maximum profit and still gives a feasible solution. For a graph $G = (V, E)$, we let $n = |V|$ and $m = |E|$.

Greedy Algorithm for a Maximum Weight Tree

Initialization. Set $E^0 = E, T^0 = \phi$. Order the edges by nonincreasing weight $c_1 \geq c_2 \geq \ldots \geq c_m$, where c_t is the cost of edge e_t.
Iteration t. If $T^{t-1} \cup \{e_t\}$ contains no cycle, set $T^t = T^{t-1} \cup \{e_t\}$. Otherwise $T^t = T^{t-1}$.
Set $E^t = E^{t-1} \setminus \{e_t\}$. If $| T^t |= n - 1$, stop with T^t is optimal. If $t = m$, stop with no feasible solution.

To obtain a maximum weight **forest**, it suffices to modify the greedy algorithm to stop as soon as $c_{t+1} \leq 0$.

Example 3.2 Consider the graph shown in Figure 3.2 with the weights on the edges as shown.

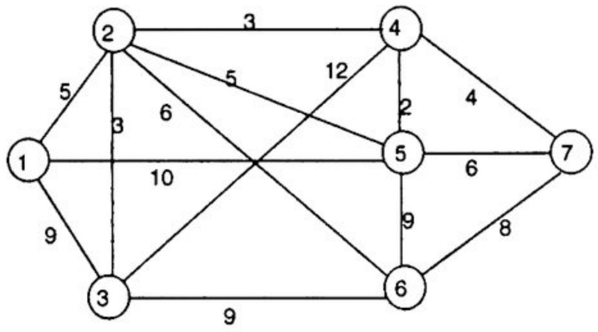

Fig. 3.2 Graph for optimal weight tree

We consider the edges in the order:

	e_1	e_2	e_3	e_4	e_5	e_6	e_7
(i,j)	$(3,4)$	$(1,5)$	$(1,3)$	$(3,6)$	$(5,6)$	$(6,7)$	$(5,7)$
c_e	12	10	9	9	9	8	6
	$+$	$+$	$+$	$+$		$+$	

e_8	e_9	e_{10}	e_{11}	e_{12}	e_{13}	e_{14}
$(2,6)$	$(1,2)$	$(2,5)$	$(4,7)$	$(2,3)$	$(2,4)$	$(4,5)$
6	5	5	4	3	3	2 .
	$+$					

The algorithm chooses the edges marked with a $+$ and rejects the others. For example, the first edge rejected is $e_5 = (5,6)$ because it forms a cycle with e_2, e_3, and e_4, which have already been selected. ■

Theorem 3.8 *The greedy algorithm terminates with an optimal weight tree.*

Proof. Suppose for simplicity that all the edge weights are different. Let $T = \{g_1, \ldots, g_{n-1}\}$ be the edges chosen by the greedy algorithm with $c_{g_1} > \ldots > c_{g_{n-1}}$. Let $F = \{f_1, \ldots, f_{n-1}\}$ be the edges of an optimal solution with $c_{f_1} > \ldots > c_{f_{n-1}}$.

If the solutions are the same, the result is proved. So suppose the two solutions differ with $g_1 = f_1, \ldots, g_{k-1} = f_{k-1}$ but $g_k \neq f_k$.

(i) We observe that $c_{g_k} > c_{f_k}$ because the greedy algorithm chooses g_k and not f_k and neither edge creates a cycle with $\{g_1, \ldots, g_{k-1}\}$. Also by construction $c_{f_1} > \ldots > c_{f_{n-1}}$, and so $g_k \notin F$.
(ii) Now consider the edge set $F \cup \{g_k\}$. As F is a tree, it follows from Proposition 3.7 that $F \cup \{g_k\}$ contains exactly one cycle C. Note however that the set of edges $\{f_1, \ldots, f_{k-1}, g_k\} = \{g_1, \ldots, g_{k-1}, g_k\}$ forms part of the tree T and thus does not contain a cycle. Therefore one of the other edges $f_k, \ldots, f_{n-1}$ must be in the cycle C. Suppose f^* is one such edge.
(iii) As C contains a unique cycle and f^* is in this cycle, $T' = F \cup \{g_k\} \setminus \{f^*\}$ is cycle free. As it has $n-1$ edges, it is a tree.
(iv) Finally, as $c_{g_k} > c_{f_k}$ and $c_{f_k} \geq c_{f^*}$, the weight of T' exceeds that of F.
(v) As F is an optimal tree, we have arrived at a contradiction, and so T and F cannot differ. ■

As the greedy algorithm is easily seen to be polynomial, the Efficient Optimization Property holds for the maximum weight tree problem. So we can again consider the other three properties. To do this, we need a formulation of the problem as an integer program. In modeling the traveling salesman problem in Section 1.2 we saw how to avoid cycles. Thus the maximum weight forest problem can be formulated as:

$$\max \textstyle\sum_{e \in E} c_e x_e \tag{3.10}$$
$$\textstyle\sum_{e \in E(S)} x_e \leq |S| - 1 \text{ for } 2 \leq |S| \leq n \tag{3.11}$$

$$x_e \geq 0 \text{ for } e \in E \tag{3.12}$$

$$x \in Z^{|E|}. \tag{3.13}$$

Theorem 3.9 *The convex hull of the incidence vectors of the forests in a graph is given by the constraints (3.11)-(3.12).*

This result says that the Explicit Convex Hull Property holds for the Maximum Weight Forest Problem. We generalize and prove this result in the next section, and in Chapter 9 we show that the Efficient Separation Property holds for this problem.

To terminate this section we introduce an important and more difficult generalization of the optimal tree problem. Given a graph $G = (V, E)$ and a set of terminals $T \subseteq V$, a *Steiner tree on* T is an edge-minimal acyclic subgraph of G containing a path joining every pair of nodes in T. Such a subgraph may or may not have edges incident to the nodes in $V \setminus T$. Given weights c_e for $e \in E$, the *Optimal Steiner Tree Problem* is to find a minimum weight Steiner tree. Observe that when $T = V$, this is the optimal tree problem, and when $| T |= 2$, it is the shortest path problem.

3.6 SUBMODULARITY AND MATROIDS*

Here we examine a larger class of problems for which a greedy algorithm provides an optimal solution. This generalizes the maximum weight forest problem examined in the last section. $\mathcal{P}(N)$ denotes the set of subsets of N.

Definition 3.5 (i) A set function $f : \mathcal{P}\ (N) \to R^1$ is *submodular* if

$$f(A) + f(B) \geq f(A \cap B) + f(A \cup B) \text{ for all } A, B \subseteq N.$$

(ii) A set function f is *nondecreasing* if

$$f(A) \leq f(B) \text{ for all } A, B \text{ with } A \subset B \subseteq N.$$

An alternative representation of such functions is useful.

Proposition 3.10 *A set function f is non-decreasing and submodular if and only if*

$$f(A) \leq f(B) + \sum_{j \in A \setminus B} [f(B \cup \{j\}) - f(B)] \text{ for all } A, B \subseteq N.$$

Proof. Suppose f is non-decreasing and submodular. Let $A \setminus B = \{j_1, \ldots, j_r\}$. Then $f(A) \leq f(A \cup B) = \sum_{i=1}^{r} [f(B \cup \{j_1, \ldots, j_i\}) - f(B \cup \{j_1, \ldots, j_{i-1}\})] \leq$

$\sum_{i=1}^{r}[f(B \cup \{j_i\}) - f(B)] = \sum_{j \in A \setminus B}[f(B \cup \{j\}] - f(B)]$, where the first inequality follows from f nondecreasing and the second from submodularity. The other direction is immediate. ∎

Now given a nondecreasing submodular function f on N with $f(\emptyset) = 0$, we consider the *submodular polyhedron*:

$$P(f) = \{x \in R^n_+ : \sum_{j \in S} x_j \leq f(S) \text{ for } S \subseteq N\},$$

and the associated *submodular optimization problem*:

$$\max\{cx : x \in P(f)\}.$$

The Greedy Algorithm for the Submodular Optimization Problem

(i) Order the variables so that $c_1 \geq c_2 \geq \ldots \geq c_r > 0 \geq c_{r+1} \geq \ldots \geq c_n$.
(ii) Set $x_i = f(S^i) - f(S^{i-1})$ for $i = 1, \ldots, r$ and $x_j = 0$ for $j > r$, where $S^i = \{1, \ldots, i\}$ for $i = 1, \ldots, r$ and $S^0 = \emptyset$.

Theorem 3.11 *The greedy algorithm solves the submodular optimization problem.*

Proof. As f is nondecreasing, $x_i = f(S^i) - f(S^{i-1}) \geq 0$ for $i = 1, \ldots, r$. Also for each $T \subseteq N$,

$$\begin{aligned}
\sum_{j \in T} x_j &= \sum_{j \in T \cap S^r} [f(S^j) - f(S^{j-1})] \\
&\leq \sum_{j \in T \cap S^r} [f(S^j \cap T) - f(S^{j-1} \cap T)] \\
&\leq \sum_{j \in S^r} [f(S^j \cap T) - f(S^{j-1} \cap T)] \\
&= f(S^r \cap T) - f(\emptyset) \leq f(T),
\end{aligned}$$

where the first inequality follows from the submodularity of f, and the others as f is nondecreasing. So the greedy solution is feasible with value $\sum_{i=1}^{r} c_i[f(S^i) - f(S^{i-1})]$.

Now consider the linear programming dual:

$$\begin{gathered}
\min \textstyle\sum_{S \subseteq N} f(S) y_S \\
\textstyle\sum_{S: j \in S} y_S \geq c_j \text{ for } j \in N \\
y_S \geq 0 \text{ for } S \subseteq N.
\end{gathered}$$

Let $y_{S^i} = c_i - c_{i+1}$ for $i = 1, \ldots, r-1$, $y_{S^r} = c_r$, and $y_S = 0$ otherwise. Clearly $y_S \geq 0$ for all $S \subseteq N$. Also for $j \leq r$, $\sum_{S: j \in S} y_S \geq \sum_{i=j}^{r} y_{S^i} =$

$\sum_{i=j}^{r-1}(c_i - c_{i+1}) + c_r = c_j$, and for $j > r$, $\sum_{S:j\in S} y_S \geq 0 \geq c_j$. Thus the solution y is dual feasible. Finally the dual objective value is

$$\sum_{i=1}^{r} f(S^i)y_{S^i} = \sum_{i=1}^{r-1} f(S^i)(c_i - c_{i+1}) + f(S^r)c_r = \sum_{i=1}^{r} c_i[f(S^i) - f(S^{i-1})].$$

So the value of the dual feasible solution has the same value as the greedy solution, and so from linear programming duality, the greedy solution is optimal. ∎

Note that when f is integer-valued, the greedy algorithm provides an integral solution. In the special case when $f(S \cup \{j\}) - f(S) \in \{0, 1\}$ for all $S \subset N$ and $j \in N \setminus S$, we call f a *submodular rank function*, and the greedy solution is a $0 - -1$ vector. What is more, we now show that the feasible 0–1 points in the submodular rank polyhedron generate an interesting combinatorial structure, called a *matroid.*

Proposition 3.12 *Suppose that r is a submodular rank function on a set N with $r(\emptyset) = 0$.*
(i) $r(A) \leq |A|$ for all $A \subseteq N$.
(ii) If $r(A) = |A|$, then $r(B) = |B|$ for all $B \subset A \subseteq N$.
(iii) If x^A is the incidence vector of $A \subseteq N$, $x^A \in P(r)$ if and only if $r(A) = |A|$.

Proof. (i) Using Proposition 3.10 and the property of a submodular rank function, $r(A) \leq r(\emptyset) + \sum_{j\in A}[r(\{j\}) - r(\emptyset)] \leq |A|$.
(ii) Again using the same properties, $|A| = r(A) \leq r(B) + \sum_{j\in A\setminus B}[r(B \cup \{j\}) - r(B)] \leq |B| + |A \setminus B| = |A|$. Equality must hold throughout, and thus $r(B) = |B|$.
(iii) If $r(A) < |A|, \sum_{j\in A} x_j^A = |A| > r(A)$ and $x^A \notin P(r)$. If $r(A) = |A|$, $\sum_{j\in S} x_j^A = |A \cap S| = r(A \cap S) \leq r(S)$ where the second equality uses (ii). This inequality holds for all $S \subseteq N$, and thus $x^A \in P(r)$. ∎

Definition 3.6 Given a submodular rank function r, a set $A \subseteq N$ is *independent* if $r(A) = |A|$. The pair $(N,\mathcal{F})$, where $\mathcal{F}$ is the set of independent sets, is called a *matroid* .

Based on Theorem 3.11, we know how to optimize on matroids.

Theorem 3.13 *The greedy algorithm solves the maximum weight independent set problem in a matroid.*

Given a connected graph $G = (V, E)$, it is not difficult to verify that the edge sets of forests form a matroid and that the function $r : \mathcal{P}(E) \rightarrow R^1$, where $r(E')$ is the size of the largest forest in (V, E'), is a submodular rank function. Specifically when $S \subseteq V$, $E' = E(S)$ and the subgraph $(S, E(S))$

is connected, we clearly have $r(E(S)) = |S| - 1$, so the forest polyhedron (3.11)-(3.12) is a special case of a submodular polyhedron.

The constraint set associated to a submodular polyhedron has another interesting property.

Definition 3.7 A set of linear inequalities $Ax \leq b$ is called *Totally Dual Integral (TDI)* if, for all $c \in Z^n$ for which the linear program $\max\{cx : Ax \leq b\}$ has a finite optimal value, the dual linear program

$$\min\{yb : yA = c, y \geq 0\}$$

has an optimal solution with y integral.

We have seen in the proof of Theorem 3.11 that the linear system $\{\sum_{j \in S} x_j \leq f(S)$ for $S \subseteq N, x_j \geq 0$ for $j \in N\}$ is TDI. Based on the following result, the TDI property provides another useful way of showing that certain linear programs always have integer solutions.

Theorem 3.14 *If $Ax \leq b$ is TDI, b is an integer vector, and $P = \{x \in R^n : Ax \leq b\}$ has vertices, then all vertices of P are integral.*

Note also that if A is a TU matrix, then $Ax \leq b$ is TDI.

3.7 NOTES

3.1 The theoretical importance of the separation problem is discussed at the end of Chapter 6. Its practical importance was brought out in the first computational studies using strong cutting planes; see Chapter 9.

3.2 Totally unimodular matrices have been studied since the fifties. The characterization of Proposition 3.1 is due to [HofKru56]. The interval or consecutive 1's property of Exercise 3.3 is due to [FulGro65], and the stronger necessary condition is from [Gho62]. A complete characterization of TU matrices is much more difficult; see [Sey80] or the presentation in [Sch86].

3.3 We again refer to [AhuMagOrl93] for network flows, as well as shortest path and max flow problems. This book also contains a large number of applications and a wealth of exercises. Note also the chapter of Ahuja on flows and paths in [DelAMafMar97], which also indicates where some of the latest software for network flow problems can be obtained.

3.4 The max flow min cut theorem was already part of the max flow algorithm of [ForFul56]. The min cut problem arises as a separation problem in solving TSP and other network design problems. The problem of finding all minimum cuts was answered in [GomHu61]. Recently new algorithms have appeared that find minimum cuts directly without using flows; see Ch. 3 in

[CooCunetal97].

3.5 The greedy algorithm for finding minimum weight trees is from [Kru56]. A faster classical algorithm is that of [Prim57]. Special algorithms based on Delaunay triangulations can be used for two-dimensional Euclidean problems, [PreSha85]. [Goe94] and [MagWol95] contain a discussion of many alternative formulations for tree and Steiner tree problems.

3.6 Submodular polyhedra and the greedy algorithm for matroids are studied in [Edm70] and [Edm71], see also [Law76]. [Wel76] is a book devoted to matroids. For total dual integrality, see [EdmGil77].

3.8 EXERCISES

1. Are the following matrices totally unimodular or not?

$$A_1 = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 \end{pmatrix}, A_2 = \begin{pmatrix} -1 & & 1 & & -1 & & \\ & 1 & & 1 & 1 & & 1 \\ -1 & 1 & & & & & \\ & & 1 & & & 1 & 1 \\ & & & 1 & & -1 & \end{pmatrix}.$$

2. Prove that the polyhedron $P = \{(x_1, \ldots, x_m, y) \in R_+^{m+1} : y \leq 1, x_i \leq y$ for $i = 1, \ldots, m\}$ has integer vertices.

3. A 0–1 matrix B has the *consecutive 1's property* if for any column j, $b_{ij} = b_{i'j} = 1$ with $i < i'$ implies $b_{lj} = 1$ for $i < l < i'$.

A more general sufficient condition for total unimodularity is: Matrix A is TU if

(i) $a_{ij} \in \{+1, -1, 0\}$ for all i, j.
(ii) For any subset M of the rows, there exists a partition (M_1, M_2) of M such that each column j satisfies

$$\mid \sum_{i \in M_1} a_{ij} - \sum_{i \in M_2} a_{ij} \mid \leq 1.$$

Use this to show that a matrix with the consecutive 1's property is TU.

4. Consider a scheduling model in which a machine can be switched on at most k times: $\sum_t z_t \leq k, z_t - y_t + y_{t-1} \geq 0, z_t \leq y_t, 0 \leq y_t, z_t \leq 1$ for all t, where $y_t = 1$ if the machine is on in period t, and $z_t = 1$ if it is switched on in period t. Show that the resulting matrix is TU.

5. Prove Proposition 3.3.

6. Use linear programming to find the length of a shortest path from node s to node t in the directed graph of Figure 3.3. Use an optimal dual solution to prove that your solution is optimal.

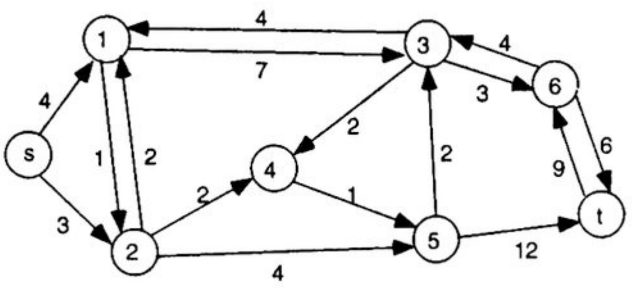

Fig. 3.3 Shortest path instance

7. Use linear programming to find a minimum $s - t$ cut in the capacitated network of Figure 3.4.

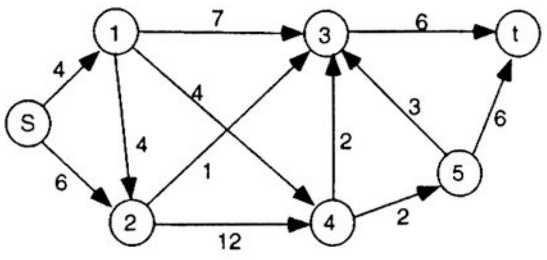

Fig. 3.4 Network instance

8. Find a minimum weight spanning tree in the graph shown in Figure 3.5.

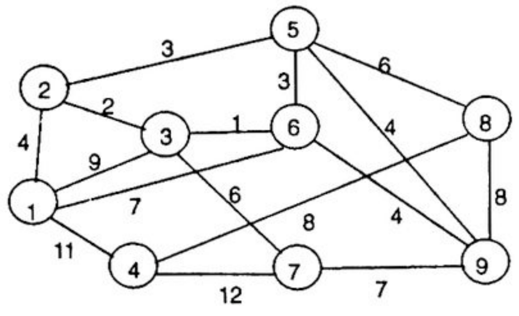

Fig. 3.5 Tree instance

9. Prove that the greedy algorithm produces an optimal weight tree when edge weights can be equal.

10. Formulate the optimal Steiner tree problem as an integer program.

11. (i) For their annual Christmas party the thirty members of staff of the thriving company Ipopt were invited/obliged to dine together and then spend the night in a fancy hotel. The boss's secretary had the unenviable task of allocating the staff two to a room. Knowing the likes and dislikes of everyone, she drew up a list of all the compatible pairs. How could you help her to fill all fifteen rooms?
(ii) Recently a summer camp was organized for an equal number of English and French children. After a few days, the children had to participate in an orienteering competition in pairs, each pair made up of one French and one English child. To allocate the pairs, each potential pair was asked to give a weight from 1 to 10 representing their willingness to form a pair. Formulate the problem of choosing the pairs so as to maximize the sum of the weights.
(iii) If you have a linear programming code available, can you help either the boss's secretary or the camp organizer or both?

12. Consider a real matrix C with n columns. Let $N = \{1, \ldots, n\}$ and $\mathcal{F} = \{S \subseteq N : \text{the columns } \{c_j\}_{j \in S} \text{ are linearly independent}\}$. Show that $(N,\mathcal{F})$ is a matroid. What is the associated rank function r?

13. Given a matroid, show that
(i) if A and B are independent sets with $|A| > |B|$, then there exists $j \in A \setminus B$ such that $A \cup \{j\}$ is independent, and
(ii) for an arbitrary set $A \subseteq N$, every maximal independent set in A has the same cardinality.