11

Column Generation Algorithms

11.1 INTRODUCTION

One of the recurring ideas in optimization and in this text is that of decomposition. We have several times considered what happens when the integer programming problem *(IP)* $\max\{cx : x \in X\}$ has a feasible region X that can be written as the intersection of two or more sets with structure $X = \cap_{k=0}^{K} X^k$ for some $K \geq 1$. Even more particular is the case where the constraints take the form:

$$\begin{array}{lllllll}
A^1x^1 & +A^2x^2 & +\ldots & +A^Kx^K & = & b \\
D^1x^1 & & & & \leq & d_1 \\
 & \ldots & & & \leq & . \\
 & & \ldots & & \leq & . \\
 & & & D^Kx^K & \leq & d_K \\
x^1 \in Z_+^{n_1}, & \ldots & \ldots & ,x^K \in Z_+^{n_K} & &
\end{array}$$

so that the sets $X^k = \{x^k \in Z_+^{n_k} : D^kx^k \leq d_k\}$ are independent for $k = 1,\ldots,K$, and only the *joint* constraints $\sum_{k=1}^{K} A^kx^k = b$ link together the different sets of variables.

Given an objective function $\max \sum_{k=1}^{K} c^kx^k$, two earlier approaches that would permit us to benefit from such structure are cut generation, in which we would try to generate valid inequalities for each subset $X^k, k = 1,\ldots,K$, and Lagrangian relaxation, in which we would dualize the joint constraints so as to obtain a dual problem:

$$\min_u L(u),$$

where

$$L(u) = \max\{\sum_{k=1}^{K}(c^k - uA^k)x^k + ub : x^k \in X^k \text{ for } k = 1, \ldots, K\},$$

and the calculation of $L(u)$ breaks up into K distinct subproblems:

$$L(u) = \sum_{k=1}^{K} \max\{(c^k - uA^k)x^k : x^k \in X^k\} + ub.$$

In this chapter we examine a third way to exploit the structure of integer programs of the above form. Throughout we assume that each of the sets X^k is bounded for $k = 1, \ldots, K$. The approach essentially involves solving an equivalent problem of the form:

$$\max\{\sum_{k=1}^{K} \gamma^k \lambda^k : \sum_{k=1}^{K} B^k \lambda^k = \beta, \lambda^k \geq 0 \text{ integer for } k = 1, \ldots, K\}$$

where each matrix B^k has a very large number of columns, one for each of the feasible points in X^k, and each vector λ^k contains the corresponding variables.

For example, we now derive an alternative formulation of this type for the uncapacitated facility location problem UFL. Here the locations $j = 1, \ldots, n$ correspond to the indices $k = 1, \ldots, K$. For each nonempty subset $S \subseteq M$ of clients, let $\lambda_S^j = 1$ if depot j satisfies the demand of client set S. This then leads to the formulation:

$$\min \textstyle\sum_{j \in N} \sum_{S \neq \phi} (\sum_{i \in S} c_{ij} + f_j)\lambda_S^j \tag{11.1}$$

$$\textstyle\sum_{j \in N} \sum_{S \neq \phi, i \in S} \lambda_S^j = 1 \quad \text{for } i \in M \tag{11.2}$$

$$\textstyle\sum_{S \neq \phi} \lambda_S^j \leq 1 \quad \text{for } j \in N \tag{11.3}$$

$$\lambda_S^j \in \{0,1\} \text{for } \phi \neq S \subseteq M, j \in N. \tag{11.4}$$

Here the cost of λ_S^j is the cost of opening depot j and serving the clients in S from depot j. The first constraints again impose that each client is served, while the second set of constraints ensure that at most one subset of clients is assigned to depot j. In practice the latter constraints are typically unnecessary. Why?

Thus the problems we wish to solve here are integer programs with an enormous number of variables, where the columns are often described implicitly as the incidence vectors of certain subsets of a set, that is tours, client subsets, and so on. Below we show how such (large) formulations of an integer program, called *Master Problems*, also arise by reformulation. We then consider how to solve the linear programming relaxation of these Master Problems, and relate the strength of this relaxation to that obtained by Lagrangian duality, or by the use of cutting planes. Finally we consider what to do when the linear programming solution is not integral, and we must resort to enumeration, leading to *IP Column Generation* or *Branch-and-Price* algorithms.

11.2 DANTZIG-WOLFE REFORMULATION OF AN IP

Consider the problem in the form:

$$(IP) \quad z = \max\{\sum_{k=1}^{K} c^k x^k : \sum_{k=1}^{K} A^k x^k = b, x^k \in X^k \text{ for } k = 1, \ldots, K\} \tag{11.5}$$

where $X^k = \{x^k \in Z_+^{n_k} : D^k x^k \leq d_k\}$ for $k = 1, \ldots, K$. Assuming that each set X^k contains a large but finite set of points $\{x^{k,t}\}_{t=1}^{T_k}$, we have that $X^k =$

$$\{x^k \in R^{n_k} : x^k = \sum_{t=1}^{T_k} \lambda_{k,t} x^{k,t}, \sum_{t=1}^{T_k} \lambda_{k,t} = 1, \lambda_{k,t} \in \{0,1\} \text{ for } t = 1, \ldots, T_k\}.$$

Now substituting for x^k leads to an equivalent *IP Master Problem*:

$$(IPM) \quad \begin{array}{c} z = \max \sum_{k=1}^{K} \sum_{t=1}^{T_k} (c^k x^{k,t})\lambda_{k,t} \\ \sum_{k=1}^{K} \sum_{t=1}^{T_k} (A^k x^{k,t})\lambda_{k,t} = b \\ \sum_{t=1}^{T_k} \lambda_{k,t} = 1 \quad \text{for } k = 1, \ldots, K \\ \lambda_{k,t} \in \{0,1\} \quad \text{for } t = 1, \ldots, T_k \text{ and } k = 1, \ldots, K. \end{array}$$

Continuing with problem UFL, suppose we start from the weak formulation

$$\min \textstyle\sum_{i \in M} \sum_{j \in N} c_{ij} x_{ij} + \sum_{j \in N} f_j y_j \tag{11.6}$$

$$\textstyle\sum_{j \in N} x_{ij} = 1 \quad \text{for } i \in M \tag{11.7}$$

$$\textstyle\sum_{i \in M} x_{ij} \leq m y_j \quad \text{for } j \in N \tag{11.8}$$

$$x \in B^{|M| \times |N|}, y \in B^{|N|}. \tag{11.9}$$

Here we can take (11.7) as the joint constraints, and $X^k = \{(x_{1k}, \ldots, x_{mk}, y_k) : \sum_{i \in M} x_{ik} \leq m y_k, \ x_{ik} \in B^1 \text{ for } i \in M, \ y_k \in \{0,1\}\}$. The points in X^k are $\{(x_S^k, 1)\}_{S \subseteq M}$, where x_S^k is the incidence vector of $S \subseteq M$, and $(0,0)$ with associated variables λ_S^k, ν^k respectively leading to the *IP* Master Problem:

$$\begin{array}{c} \min \sum_{j \in N} [\sum_{S \neq \phi} (\sum_{i \in S} c_{ij} + f_j)\lambda_S^j + f_j \lambda_\emptyset^j] \\ \sum_{j \in N} \sum_{S \neq \phi, i \in S} \lambda_S^j = 1 \quad \text{for } i \in M \\ \sum_{S \neq \phi} \lambda_S^j + \lambda_\emptyset^j + \nu^j = 1 \quad \text{for } j \in N \\ \lambda_S^j \in \{0,1\} \text{ for } S \subseteq M, j \in N, \nu^j \in \{0,1\} \quad \text{for } j \in N. \end{array}$$

Observe that as $f_j \geq 0$, variable $\lambda_\emptyset^j$ is dominated by ν^j and can be dropped. Now this formulation and (11.1)–(11.4) are identical if we take ν^j to be the slack variable in (11.3).

11.3 SOLVING THE MASTER LINEAR PROGRAM

Here we use a column generation algorithm to solve the linear programming relaxation of the Integer Programming Master Problem, called the *Linear Programming Master Problem:*

$$(LPM)\qquad \begin{aligned} z^{LPM} = \max \textstyle\sum_{k=1}^{K} \sum_{t=1}^{T_k} (c^k x^{k,t})\lambda_{k,t} \\ \textstyle\sum_{k=1}^{K} \sum_{t=1}^{T_k} (A^k x^{k,t})\lambda_{k,t} = b \\ \textstyle\sum_{t=1}^{T_k} \lambda_{k,t} = 1 \text{ for } k = 1,\ldots,K \\ \lambda_{k,t} \geq 0 \quad \text{for } t = 1,\ldots,T^k, k = 1,\ldots,K \end{aligned}$$

where there is a column $\begin{pmatrix} c^k x \\ A^k x \\ e_k \end{pmatrix}$ for each $x \in X^k$. Below we will use $\{\pi_i\}_{i=1}^m$ as the dual variables associated with the joint constraints, and $\{\mu_k\}_{k=1}^K$ as dual variables for the second set of constraints, known as *convexity* constraints.

The idea is to solve the linear program by the primal simplex algorithm. However, the pricing step of choosing a column to enter the basis must be modified because of the enormous number of columns. Rather than pricing the columns one by one, the problem of finding a column with the largest reduced price is itself a set of K optimization problems.

Initialization. We suppose that a subset of columns (at least one for each k) is available, providing a feasible *Restricted Linear Programming Master Problem*

$$(RLPM)\qquad \begin{aligned} \tilde{z}^{LPM} = \max \tilde{c}\tilde{\lambda} \\ \tilde{A}\tilde{\lambda} = \tilde{b} \\ \tilde{\lambda} \geq 0 \end{aligned}$$

where $\tilde{b} = \begin{pmatrix} b \\ 1 \end{pmatrix}$, $\tilde{A}$ is generated by the available set of columns and is a submatrix of

$$\begin{pmatrix} A^1x^{1,1} & .. & A^1x^{1,T_1} & A^2x^{2,1} & .. & A^2x^{2,T_2} & .. & A^Kx^{K,1} & .. & A^Kx^{K,T_K} \\ 1 & .. & 1 & & & & & & & \\ & & & 1 & .. & 1 & & & & \\ & & & & & & .. & & & \\ & & & & & & & 1 & .. & 1 \end{pmatrix},$$

and $\tilde{c}, \tilde{\lambda}$ are the corresponding costs and variables. Solving $RLPM$ gives an optimal primal solution $\tilde{\lambda}^*$ and an optimal dual solution $(\pi,\mu) \in R^m \times R^K$.

Primal Feasibility. Any feasible solution of $RLPM$ is feasible for LPM. In particular, $\tilde{\lambda}^*$ is a feasible solution of LPM, and so $\tilde{z}^{LPM} = \tilde{c}\tilde{\lambda}^* = \sum_{i=1}^m \pi_i b_i + \sum_{k=1}^K \mu_k \leq z^{LPM}$.

Optimality Check for LPM. We need to check whether (π,μ) is dual feasible for LPM. This involves checking for each column, that is for each k, and for each $x \in X^k$ whether the reduced price $c^k x - \pi A^k x - \mu_k \leq 0$. Rather than examining each point separately, we treat all points in X^k implicitly by solving an optimization subproblem:

$$\zeta_k = \max\{(c^k - \pi A^k)x - \mu_k : x \in X^k\}.$$

Stopping Criterion. If $\zeta_k = 0$ for $k = 1,\ldots,K$, the solution (π,μ) is dual feasible for LPM, and so $z^{LPM} \leq \sum_{i=1}^m \pi_i b_i + \sum_{k=1}^K \mu_k$. As the value of the primal feasible solution $\tilde{\lambda}$ equals that of this upper bound, $\tilde{\lambda}$ is optimal for LPM.

Generating a New Column. If $\zeta_k > 0$ for some k, the column corresponding to the optimal solution $\tilde{x}^k$ of the subproblem has positive reduced price. Introducing the column $\begin{pmatrix} c^k\tilde{x}^k \\ A^k\tilde{x}^k \\ e_k \end{pmatrix}$ leads to a new Restricted Linear Programming Master Problem that can be easily reoptimized (e.g., by the primal simplex algorithm).

A Dual (Upper) Bound. From the subproblem, we have that $\zeta_k \geq (c^k - \pi A^k)x - \mu_k$ for all $x \in X^k$. It follows that $(c^k - \pi A^k)x - \mu_k - \zeta_k \leq 0$ for all $x \in X^k$. Therefore setting $\zeta = (\zeta_1,\ldots,\zeta_K)$, we have that $(\pi, \mu + \zeta)$ is dual feasible in LPM. Therefore

$$z^{LPM} \leq \pi b + \sum_{k=1}^K \mu_k + \sum_{k=1}^K \zeta_k.$$

These different observations lead directly to an algorithm for LPM that terminates when $\zeta_k = 0$ for $k = 1,\ldots,K$. However, as in Lagrangian relaxation, it may be possible to terminate earlier.

An Alternative Stopping Criterion. If the subproblem solutions $(\tilde{x}^1,\ldots,\tilde{x}^K)$ satisfy the original joint constraints $\sum_{k=1}^K A^k x^k = b$, then $(\tilde{x}^1,\ldots,\tilde{x}^K)$ is optimal.

This follows because $\zeta_k = (c^k - \pi A^k)\tilde{x}^k - \mu_k$ implies that $\sum_k c^k\tilde{x}^k = \sum_k \pi A^k \tilde{x}^k + \sum_k \mu_k + \sum_k \zeta_k = \pi b + \sum_k \mu_k + \sum_k \zeta_k$. Therefore the primal feasible solution has the same value as the upper bound on z^{LPM}.

11.3.1 STSP by Column Generation

Here we consider the application of the above algorithm to solve the Master Linear Program of a problem in which there is just a single subproblem. We again consider the symmetric traveling salesman problem, which can be written as

$$\min\{\sum_{e\in E} c_e x_e : \sum_{e\in\delta(i)} x_e = 2 \text{ for } i\in N, x\in X^1\}$$

where

$$X^1 = \{x\in Z_+^m : \textstyle\sum_{e\in\delta(1)} x_e = 2, \sum_{e\in E(S)} x_e \leq |S|-1 \text{ for } \phi\subset S\subset N\setminus\{1\},$$

$$\textstyle\sum_{e\in E} x_e = n\}$$

is the set of incidence vectors of 1-trees.

Writing $x_e = \sum_{t:e\in E^t}\lambda_t$, where $G^t = (N,E^t)$ is the t^{th} 1-tree, the degree constraints become $\sum_{e\in\delta(i)} x_e = \sum_{e\in\delta(i)}\sum_{t:e\in E^t}\lambda_t = \sum_t d_i^t\lambda_t = 2$ where d_i^t is the degree of node i in the 1-tree G^t. Thus the corresponding Linear Programming Master is

$$(LPM)\qquad \begin{array}{c} \min \sum_{t=1}^{T_1}(cx^t)\lambda_t \\ \sum_{t=1}^{T_1} d_i^t\lambda_t = 2 \text{ for } i\in N \\ \sum_{t=1}^{T_1}\lambda_t = 1 \\ \lambda\in R_+^T \end{array}$$

with which we associate dual variables $\{u_i\}_{i=1}^n$ to the degree constraints, and dual variable μ to the convexity constraint. The corresponding single subproblem is

$$\zeta_1 = \min\{\sum_{e\in E}(c_e - u_i - u_j)x_e - \mu : x\in X^1\}$$

as the 1-tree G^t has reduced cost $cx^t - \sum_{i\in N} d_i^t u_i - \mu = cx^t - \sum_{i\in N} u_i \sum_{e\in\delta(i)} x_e^t - \mu = \sum_{e\in E}(c_e - u_i - u_j)x_e^t - \mu$, where x_e^t for $e\in E$ are the edge variables of the 1-tree G^t, and $e = (i,j)$ for $e\in E$.

Note that because we are dealing with 1-trees, $d_1^t = 2$ for all t, and so the first equation in LPM is twice the convexity constraint. As a result we can drop the convexity constraint.

Example 11.1 Consider an instance of $STSP$ with distance matrix

$$c_e = \begin{pmatrix} \cdot & 7 & 2 & 1 & 5 \\ & \cdot & 3 & 6 & 8 \\ & & \cdot & 4 & 2 \\ & & & \cdot & 9 \\ & & & & \cdot \end{pmatrix}.$$

We initialize with a restricted LPM having 7 columns, corresponding to a tour of length 28 and six 1-trees chosen arbitrarily

$$\begin{array}{lccccccccc} \min & 28\lambda_1 & +25\lambda_2 & +21\lambda_3 & +19\lambda_4 & +22\lambda_5 & +18\lambda_6 & +28\lambda_7 & & \\ & 2\lambda_1 & +2\lambda_2 & +2\lambda_3 & +2\lambda_4 & +2\lambda_5 & +2\lambda_6 & +2\lambda_7 & = & 2 \\ & 2\lambda_1 & +2\lambda_2 & +2\lambda_3 & +1\lambda_4 & +1\lambda_5 & +2\lambda_6 & +3\lambda_7 & = & 2 \\ & 2\lambda_1 & +3\lambda_2 & +2\lambda_3 & +3\lambda_4 & +2\lambda_5 & +3\lambda_6 & +1\lambda_7 & = & 2 \\ & 2\lambda_1 & +2\lambda_2 & +3\lambda_3 & +3\lambda_4 & +3\lambda_5 & +1\lambda_6 & +1\lambda_7 & = & 2 \\ & 2\lambda_1 & +1\lambda_2 & +1\lambda_3 & +1\lambda_4 & +2\lambda_5 & +2\lambda_6 & +3\lambda_7 & = & 2 \\ & & & \lambda & \geq & 0. & & & & \end{array}$$

The resulting linear programming solution is $\lambda = (0,0,\frac{1}{4},0,\frac{1}{4},\frac{1}{4},\frac{1}{4})$ with cost 22.5 and dual solution $u = (\frac{151}{8},-1,-\frac{11}{2},-\frac{5}{4},0)$. The corresponding reduced cost matrix for the subproblem is

$$\begin{pmatrix} \cdot & -\frac{87}{8} & -\frac{91}{8} & -\frac{133}{8} & -\frac{111}{8} \\ & \cdot & \frac{19}{2} & \frac{33}{4} & 9 \\ & & \cdot & \frac{43}{4} & \frac{15}{2} \\ & & & \cdot & \frac{41}{4} \\ & & & & \cdot \end{pmatrix}.$$

The optimal 1-tree is $x_{14} = x_{15} = x_{24} = x_{25} = x_{35} = 1$ with $\zeta = -\frac{23}{4}$. Therefore $22.5 + \zeta = 16.75 \leq z^{LP} \leq 22.5$.

We start a new iteration by introducing this 1-tree as a new column in the restricted master with cost 22, and degrees (2,2,1,2,3). The new linear programming solution is $\lambda = (0,0,\frac{1}{3},0,0,\frac{1}{3},0,\frac{1}{3})$ with cost 20.333 and dual solution $u = (\frac{65}{6},\frac{1}{3},-\frac{5}{3},\frac{2}{3},0)$.

The corresponding reduced cost matrix for the subproblem is

$$\begin{pmatrix} \cdot & -\frac{25}{6} & -\frac{43}{6} & -\frac{21}{2} & -\frac{35}{6} \\ & \cdot & \frac{13}{3} & 5 & \frac{23}{3} \\ & & \cdot & 5 & \frac{11}{3} \\ & & & \cdot & \frac{25}{3} \\ & & & & \cdot \end{pmatrix}.$$

The optimal 1-tree is $x_{13} = x_{14} = x_{23} = x_{24} = x_{35} = 1$ with $\zeta = -\frac{14}{3}$. The lower bound of $20.333 - \frac{14}{3} = 15.667$ is not as good as that obtained before. Therefore we now have $16.75 \leq z^{LP} \leq 20.333$.

Again we introduce this 1-tree as a new column in the restricted master with cost 14, and degrees (2,2,3,2,1). The new linear programming solution is $\lambda = (0,0,0,0,0,0,0,\frac{1}{2},\frac{1}{2})$ with cost 18 and dual solution $u = (13,0,-4,0,0)$.

The corresponding reduced cost matrix for the subproblem is

$$\begin{pmatrix} \cdot & -6 & -7 & -12 & -8 \\ & \cdot & 7 & 6 & 8 \\ & & \cdot & 8 & 6 \\ & & & \cdot & 9 \\ & & & & \cdot \end{pmatrix}.$$

The optimal 1-tree is $x_{14} = x_{15} = x_{23} = x_{24} = x_{35} = 1$ with $\zeta = -1$. The lower bound on z^{LPM} increases to $18 - 1 = 17$. As this 1-tree is a tour, it follows from the alternative stopping criterion that it is optimal. Alternatively one can check that its real cost is 17. ∎

11.3.2 Strength of the Linear Programming Master

How strong is the linear programming relaxation of the Master Problem? Is there some hope that it will solve the original problem IP?

Proposition 11.1

$$z^{LPM} = \max\{\sum_{k=1}^{K} c^k x^k : \sum_{k=1}^{K} A^k x^k = b, x^k \in conv(X^k) \text{ for } k = 1,\ldots,K\}.$$

Proof. LPM can be obtained from the original problem IP by substituting $x^k = \sum_{t=1}^{T_k} x^{k,t}\lambda_{k,t}, \sum_{t=1}^{T_k} \lambda_{k,t} = 1, \lambda_{k,t} \geq 0$ for $t = 1,\ldots,T_k$. This is equivalent to substituting $x^k \in \text{conv}(X^k)$. ∎

As discussed in the introduction to this chapter, when IP is decomposable, Lagrangian relaxation and cutting plane algorithms are two possible alternative approaches. Specifically let w_{LD} be the value of the Lagrangian dual when the joint constraints $\sum_{k=1}^{K} A^k x^k = b$ are dualized, and let z^{CUT} be the value obtained when cutting planes are added to the linear programming relaxation of IP using an exact separation algorithm for each of the sets $\text{conv}(X^k)$ for $k = 1,\ldots,K$.

The next result, showing that all three approaches are in some sense equivalent as they lead to the same dual bounds, is based on Theorem 10.3, Proposition 11.1, and the fact that an exact separation algorithm for $\text{conv}(X^k)$ implicitly generates $\text{conv}(X^k)$.

Theorem 11.2 $z^{LPM} = w_{LD} = z^{CUT}$.

As the subproblems solved in both the column generation and Lagrangian dual approaches are optimization problems over X^k, column generation can be viewed as an algorithm for solving the Lagrangian dual in which the dual variables π are updated using linear programming by solving the Restricted Linear Programming Master. This is in comparison with the subgradient algorithm often used to solve the Lagrangian dual that is based on a much simpler updating procedure.

On the other hand, if we use the cutting plane approach, though the bound obtained is potentially the same, separation problems over $\text{conv}(X^k)$ have to be solved instead of optimization problems.

As the theoretical complexity of the optimization and separation problems for $\text{conv}(X^k)$ is the same, the choice of approach depends on the relative difficulty in solving the two problems as well as on the convergence of the column generation and cutting plane algorithms in practice.

11.4 IP COLUMN GENERATION FOR 0–1 IP

If when the column generation algorithm terminates, the optimal solution vector $\tilde{\lambda} = (\tilde{\lambda}^1,\ldots,\tilde{\lambda}^K)$ of LPM is not integer, then IPM is not yet solved. However, $z^{LPM} \geq z$, which suggests the possibility of using such upper bounds in a branch-and-bound algorithm. In this section we present an algorithm for 0–1 problems using this bound, called an *IP column generation* or *branch-and-price* algorithm.

Again we have the original problem

$$(IP) \qquad \begin{aligned} z = \max\{\textstyle\sum_{k=1}^{K} c^k x^k : \sum_{k=1}^{K} A^k x^k = b,\\ D^k x^k \leq d^k \text{ for } k = 1,\ldots,K,\ x^k \in B^{n_k} \text{ for } k = 1,\ldots,K\},\end{aligned}$$

and its reformulation

$$(IPM) \qquad \begin{aligned} z = \max \textstyle\sum_{k=1}^{K} \sum_{t=1}^{T_k} (c^k x^{k,t})\lambda_{k,t}\\ \textstyle\sum_{k=1}^{K} \sum_{t=1}^{T_k} (A^k x^{k,t})\lambda_{k,t} = b\\ \textstyle\sum_{t=1}^{T_k} \lambda_{k,t} = 1 \quad \text{for } k = 1,\ldots,K\\ \lambda_{k,t} \in \{0,1\} \text{ for } t = 1,\ldots,T_k, k = 1,\ldots,K.\end{aligned}$$

whose linear programming relaxation has optimal solution $\tilde{\lambda}$.

Because the points $x^{k,t} \in X^k$ are distinct 0–1 vectors, note that $\tilde{x}^k = \sum_{t=1}^{T_k} \tilde{\lambda}_{k,t} x^{k,t} \in \{0,1\}^{n_k}$ if and only if $\tilde{\lambda}$ is integer. Therefore if $\tilde{\lambda}$ is not integer, there is some κ and j such that the corresponding 0–1 variable x_j^κ has linear programming value $\tilde{x}_j^\kappa$ that is fractional, and on which one can branch.

This suggests the branching scheme shown in Figure 11.1(a), in which the set S of all feasible solutions is split into $S_0 = S \cap \{x : x_j^\kappa = 0\}$ and $S_1 = S \cap \{x : x_j^\kappa = 1\}$. Note that this is exactly the same type of scheme used in the basic branch-and-bound algorithm in Chapter 7.

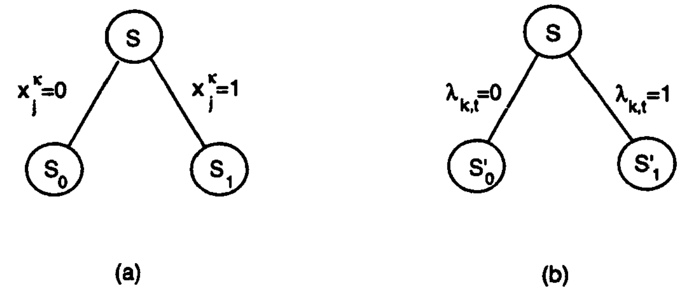

Fig. 11.1 Branching for 0–1 column generation: (a) original (b) column variables

It is important now to make sure that it will still be possible to solve the new linear programming master problems without difficulty. To do this we need to define the new Master Problems associated with S_i for $i = 0, 1$, and the new subproblems.

Now as $x_j^k = \sum_{t=1}^{T_k} \lambda_{k,t} x_j^{k,t} \in \{0,1\}$, $x_j^k = \delta \in \{0,1\}$ implies that $x_j^{k,t} = \delta$ for all k, t with $\lambda_{k,t} > 0$. So the Master Problem at node $S_i = S \cap \{x : x_j^\kappa = i\}$ for $i = 0, 1$ is

$$
(IPM(S_i)) \quad
\begin{array}{c}
z(S_i) = \max \sum_{k \neq \kappa} \sum_t (c^k x^{k,t}) \lambda_{k,t} + \sum_{t: x_j^{\kappa,t} = i} (c^\kappa x^{\kappa,t}) \lambda_{\kappa,t} \\
\sum_{k \neq \kappa} \sum_t (A^k x^{k,t}) \lambda_{k,t} + \sum_{t: x_j^{\kappa,t} = i} (A^\kappa x^{\kappa,t}) \lambda_{\kappa,t} = b \\
\sum_t \lambda_{k,t} = 1 \text{ for } k \neq \kappa \\
\sum_{t: x_j^{\kappa,t} = i} \lambda_{\kappa,t} = 1 \\
\lambda_{k,t} \in \{0,1\} \text{ for } t = 1, \ldots, T_k, k = 1, \ldots, K.
\end{array}
$$

This has the same form as the original Master Problem except that a set of columns are excluded on each branch, and the previous LPM solution is now infeasible. Turning to the column generation subproblems, the subproblem is unchanged if $k \neq \kappa$. However, for subproblem κ and $i = 0, 1$, we have

$$\zeta_\kappa(S_i) = \max\{(c^\kappa - \pi A^\kappa)x - \mu_\kappa : x \in X^\kappa, x_j = i\},$$

which is very similar to the original subproblem.

Another idea is to branch on some fractional $\lambda_{k,t}$ variable, fixing it to 0 and 1 respectively, see Figure 11.1(b). Note, however, that on the branch in which $\lambda_{k,t} = 0$, just one column, corresponding to the t^{th} solution of subproblem k, is excluded, so the resulting problem is almost identical to the original one. This means that the resulting enumeration tree has the undesirable property of being highly unbalanced. In addition it is often difficult to impose the condition $\lambda_{k,t} = 0$, and thus to prevent the same solution being generated again as optimal solution after branching.

One potential advantage of the column generation approach, visible in Example 11.2, is that the optimal solutions to $RLPM$ are often integral or close to integral. In the first case this gives a feasible integer solution, and in the second such a solution can often be obtained by a simple rounding heuristic.

11.5 IMPLICIT PARTITIONING/PACKING PROBLEMS

An important subclass of decomposable 0–1 IPs are packing and partitioning problems. Given a finite set $M = \{1, \ldots, m\}$, there are K implicitly described sets of feasible subsets, and the problem is to find a maximum value packing or partition of M consisting of certain of these subsets.

In terms of the original IP (11.5) of Section 11.2, we set $x^k = (y^k, w^k)$ with $y^k \in \{0,1\}^m$ the incidence vector of subset k of M, $c^k = (e^k, f^k)$, $A^k = (I, 0)$ and $b = 1$. One should think of the variables w^k as auxiliary variables needed to define whether the subset with incidence vector y^k is feasible, and to define the possibly nonlinear objective value of the corresponding subset. So we have the formulation

$$z = \max\{\sum_{k=1}^K (e^k y^k + f^k w^k) : \sum_{k=1}^K y^k \leq 1, (y^k, w^k) \in X^k \text{ for } k = 1, \ldots, K\}.$$

Now if $(y^{k,t}, w^{k,t})$ corresponds to the t^{th} feasible solution in the set X^k, and $\lambda_{k,t}$ is the corresponding variable, we obtain an equivalent Integer Programming Master

$$
\begin{array}{c}
z = \max \sum_{k=1}^K \sum_{t=1}^{T_k} (e^k y^{k,t} + f^k w^{k,t}) \lambda_{k,t} \\
\sum_{k=1}^K \sum_{t: y_i^{k,t} = 1} \lambda_{k,t} = 1 \text{ for } i \in M \\
\sum_{t=1}^{T_k} \lambda_{k,t} \leq 1 \text{ for } k = 1, \ldots, K \\
\lambda_{k,t} \in \{0,1\} \text{ for } t = 1, \ldots, T_k, k = 1, \ldots, K.
\end{array}
$$

We now present several problems of this type. Clearly as the partitioning problem is a special case of (11.5), the algorithm of the previous section can be applied.

Multi-Item Lot-Sizing. Suppose we are given demands d_t^k for items $k = 1, \ldots, K$ over a time horizon $t = 1, \ldots, T$. All items must be produced on a single machine; the machine can produce only one item in each period and has a capacity C_t^k if item k is produced in period t. Given production, storage, and set-up costs for each item in each period, we wish to find a minimum cost production plan. This problem can be formulated as

$$
\begin{array}{c}
\min \sum_{k=1}^K \sum_{t=1}^T (p_t^k x_t^k + h_t^k s_t^k + f_t^k y_t^k) \\
\sum_{k=1}^K y_t^k \leq 1 \text{ for } t = 1, \ldots, n \\
(x^k, s^k, y^k) \in X^k \text{ for } k = 1, \ldots, K
\end{array}
$$

where $X^k = \{(x^k, s^k, y^k) \in R_+^n \times R_+^n \times B^n : s_{t-1}^k + x_t^k = d_t^k + s_t^k,\ \ x_t^k \leq C_t^k y_t^k \text{ for } t = 1, \ldots, n\}$.

Clustering. Given a graph $G = (V, E)$, edge costs c_e for $e \in E$, node weights d_i for $i \in V$, and a cluster capacity C, we wish to split the node set V into K (possibly empty) clusters satisfying the property that the sum of the node weights in each cluster does not exceed C, in a way that minimizes the sum of the weights of edges between clusters (maximizes the sum of weights of edges within clusters). Figure 11.2 shows a feasible solution for an instance with 3 clusters and a capacity of 9. The thick edges are those between clusters. The problem can be formulated as

$$
\begin{array}{c}
\max \sum_{k=1}^K \sum_{e \in E} c_e w_e^k \\
\sum_{k=1}^K y_i^k \leq 1 \text{ for } i \in V \\
(w^k, y^k) \in X^k \text{ for } k = 1, \ldots, K
\end{array}
$$

where $X^k = \{(w^k, y^k) \in B^m \times B^n : w_e^k \leq y_i^k, w_e^k \leq y_j^k, w_e^k \geq y_i^k + y_j^k - 1$ for $e = (i,j) \in E$, $\sum_{i \in V} d_i y_i^k \leq C\}$ with $y_i^k = 1$ if node i is in cluster k and $w_e^k = 1$ if edge e has both endpoints in cluster k.

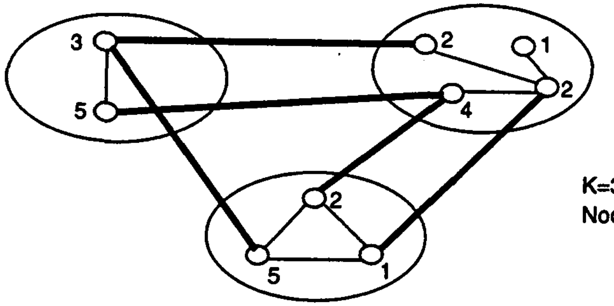

Fig. 11.2 Clustering solution with three clusters

Capacitated Vehicle Routing. Given a graph $G = (V, E)$, a depot node 0, edge costs c_e for each $e \in E$, K identical vehicles of capacity C, and client orders d_i for $i \in V \setminus \{0\}$, we wish to find a set of subtours (cycles) for each vehicle such that (i) each subtour contains the depot, (ii) together the subtours contain all the nodes, (iii) the subtours are disjoint on the node set $V \setminus \{0\}$, and (iv) the total demand on each subtour (the total amount delivered by each vehicle) does not exceed C.

Another problem with such a decomposable structure is the generalized assignment problem. An instance of GAP is treated by branch-and-cut in Section 9.6.

11.6 PARTITIONING WITH IDENTICAL SUBSETS*

The clustering and vehicle routing problems of the last section both have the property that the clusters or vehicles are interchangeable (independent of k). This means that the numbering of the subsets is arbitrary, and exchanging any two sets leads to an essentially identical solution.

Here we consider how the integer programming column generation algorithm of Section 11.4 can be specialized to take account of this symmetry. As $X^k = X$, $(e^k, f^k) = (e, f)$ and $T_k = T$ for all k, we can set $\lambda_t = \sum_{k=1}^K \lambda_{k,t}$ and IPM now takes the form:

$$\max \sum_{t=1}^T (ey^t + fw^t)\lambda_t$$
$$\sum_{t: y_i^t = 1} \lambda_t = 1 \text{ for } i \in M$$
$$\sum_{t=1}^T \lambda_t \leq K$$
$$\lambda \in B^T.$$

There is now just a single column generation subproblem. Letting the dual variables associated with the linear programming relaxation be $\{\pi_i\}_{i \in M}$ and μ, the subproblem is:

$$\zeta = \max\{(e - \pi)y + fw - \mu : (y, w) \in X\}$$

and LPM can be solved as in Section 11.3.

What happens if the solution $\tilde{\lambda}$ of LPM is not integral? It is now not at all obvious how to recover the original variables x^k or the $\lambda_{k,t}$ variables, so the branching scheme proposed in Section 11.4 must be modified. We now consider two possibilities.

Branching Rules

(i) If $\sum_{t=1}^T \lambda_t = \alpha \notin Z$, then form two branches with $\sum_{t=1}^T \lambda_t \leq \lfloor \alpha \rfloor$ and $\sum_{t=1}^T \lambda_t \geq \lceil \alpha \rceil$ respectively.

(ii) A second possibility is based on the simple observation that if we take two elements of M, either they appear together in some subset, or not. So we choose a pair of elements (rows) i and j in M for which

$$0 < \sum_{t: y_i^t = y_j^t = 1} \lambda_t < 1,$$

and we then form two branches with $\sum_{t: y_i^t = y_j^t = 1} \lambda_t = 1$ and $\sum_{t: y_i^t = y_j^t = 1} \lambda_t = 0$ respectively.

In the first case (i) we impose that i and j lie in the same subset, and in the second case (ii) that they lie in different subsets. In case (i) all columns corresponding to subsets Q containing either i or j but not both are eliminated from the Master Problem, and the constraint $y_i = y_j$ is added to the subproblem to ensure that any new column generated does not generate a subset containing i but not j, or vice versa. In case (ii) columns containing both i and j are eliminated from the Master, and the constraint $y_i + y_j \leq 1$ is added to the subproblem.

So, imposing the constraints $y_i = y_j$ or $y_i + y_j \leq 1$ on each subproblem permits us to branch as shown in Figure 11.3.

The following result says that this second branching scheme is sufficient.

Proposition 11.3 *If $\tilde{\lambda} \notin B^T$, there exist rows $i, j \in M$ such that*

$$0 < \sum_{t: y_i^t = y_j^t = 1} \lambda_t < 1.$$

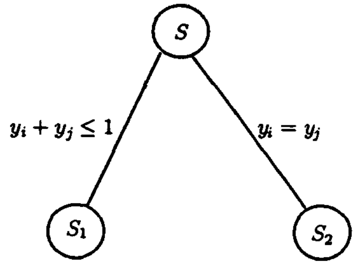

Fig. 11.3 A branching scheme for partitioning

Such a pair is also not difficult to find.

Example 11.2 Consider an instance of the clustering problem of Section 11.5 with $G = (V, E)$ the complete graph on 3 nodes, $K = 3$ clusters, the objective of choosing as many edges as possible within the clusters, and at most 2 nodes allowed per cluster, that is, node weights $d_i = 1$ for all $i \in V$, edge weights $c_e = 1$ for all $e \in E$, and cluster capacity $C = 2$.

1. *Solving LPM.* Starting the Restricted LPM with clusters consisting of single nodes leads to

$$\begin{array}{rcl} \max 0\lambda_1 + 0\lambda_2 + 0\lambda_3 & & \\ 1\lambda_1 + 0\lambda_2 + 0\lambda_3 & = & 1 \\ 0\lambda_1 + 1\lambda_2 + 0\lambda_3 & = & 1 \\ 0\lambda_1 + 0\lambda_2 + 1\lambda_3 & = & 1 \\ 1\lambda_1 + 1\lambda_2 + 1\lambda_3 & \leq & 3 \\ \lambda \geq 0 & & \end{array}$$

with $RLPM$ value 0, primal solution $\lambda = (1,1,1)$, and dual solution $\pi = (0,0,0), \mu = 0$. This provides a feasible solution to the original problem of objective value 0, and so we set $\underline{z} = 0$.

2. *Solving the Subproblem.* The subproblem of selecting a feasible cluster of maximum reduced price is

$$\begin{array}{c} \zeta = \min w_{12} + w_{13} + w_{23} - 0y_1 - 0y_2 - 0y_3 - 0 \\ w_{12} \leq y_1, w_{12} \leq y_2, w_{12} \geq y_1 + y_2 - 1 \\ w_{13} \leq y_1, w_{13} \leq y_3, w_{13} \geq y_1 + y_3 - 1 \\ w_{23} \leq y_2, w_{23} \leq y_3, w_{23} \geq y_2 + y_3 - 1 \\ y_1 + y_2 + y_3 \leq 2 \\ w \in B^{|E|}, y \in B^{|V|} \end{array}$$

giving $\zeta = 1$ and an optimal solution $w_{12} = y_1 = y_2 = 1$.

3. *Solution of LPM.* After three iterations LPM is solved in the form

$$\begin{array}{rcl} \max 0\lambda_1 + 0\lambda_2 + 0\lambda_3 + 1\lambda_4 + 1\lambda_5 + 1\lambda_6 & & \\ 1\lambda_1 + 0\lambda_2 + 0\lambda_3 + 1\lambda_4 + 1\lambda_5 + 0\lambda_6 & = & 1 \\ 0\lambda_1 + 1\lambda_2 + 0\lambda_3 + 1\lambda_4 + 0\lambda_5 + 1\lambda_6 & = & 1 \\ 0\lambda_1 + 0\lambda_2 + 1\lambda_3 + 0\lambda_4 + 1\lambda_5 + 1\lambda_6 & = & 1 \\ 1\lambda_1 + 1\lambda_2 + 1\lambda_3 + 1\lambda_4 + 1\lambda_5 + 1\lambda_6 & \leq & 3 \\ \lambda \geq 0 & & \end{array}$$

with optimal solution $\lambda_4 = \lambda_5 = \lambda_6 = \frac{1}{2}$ and $z^{LPM} = \frac{3}{2}$.

4. *Branching.* Taking rows $i = 1$ and $l = 2$, we use the second branching scheme and split the problem into two subproblems:

S_1 is the set of solutions in which nodes 1 and 2 do not lie in the same cluster, so S_1 is obtained by setting $\lambda_4 = 0$ cutting off the existing solution. All new clusters containing both nodes 1 and 2 are excluded.

S_2 is the set of solutions in which any cluster containing either 1 or 2 must contain the other, so S_2 is obtained by setting $\lambda_2 = \lambda_3 = \lambda_5 = \lambda_6 = 0$ cutting off the existing solution. Any new clusters containing just one of the nodes 1,2 are excluded.

5. *Reoptimizing for S_1.* With $\lambda_4 = 0$, the new $RLPM$ is

$$\begin{array}{rcl} \max 0\lambda_1 + 0\lambda_2 + 0\lambda_3 + 1\lambda_5 + 1\lambda_6 & & \\ 1\lambda_1 + 0\lambda_2 + 0\lambda_3 + 1\lambda_5 + 0\lambda_6 & = & 1 \\ 0\lambda_1 + 1\lambda_2 + 0\lambda_3 + 0\lambda_5 + 1\lambda_6 & = & 1 \\ 0\lambda_1 + 0\lambda_2 + 1\lambda_3 + 1\lambda_5 + 1\lambda_6 & = & 1 \\ 1\lambda_1 + 1\lambda_2 + 1\lambda_3 + 1\lambda_5 + 1\lambda_6 & \leq & 3 \\ \lambda \geq 0 & & \end{array}$$

with optimal primal solution $\lambda_1 = \lambda_6 = 1$, and dual solution $\pi = (0,0,1), \mu = 0$. The incumbent is updated, $\underline{z} = 1$.

6. *Subproblem for S_1.* The subproblem is

$$\begin{array}{c} \zeta = \min w_{12} + w_{13} + w_{23} - 1y_1 - 0y_2 - 0y_3 - 0 \\ w_{12} \leq y_1, w_{12} \leq y_2, w_{12} \geq y_1 + y_2 - 1 \\ w_{13} \leq y_1, w_{13} \leq y_3, w_{13} \geq y_1 + y_3 - 1 \\ w_{23} \leq y_2, w_{23} \leq y_3, w_{23} \geq y_2 + y_3 - 1 \\ y_1 + y_2 + y_3 \leq 2 \\ y_1 + y_2 \leq 1 \\ w \in B^{|E|}, y \in B^{|V|} \end{array}$$

giving $\zeta = 0$. So $LPM(S_1)$ is solved with $z^{LPM}(S_1) = 1$. The node is pruned by bound.

7. *Node* S_2. When setting $\lambda_2 = \lambda_3 = \lambda_5 = \lambda_6 = 0$, the new $RLPM$ has unique optimal solution $\lambda_3 = \lambda_4 = 1$. We continue iterating between subproblem and the restricted Master till LPM is solved with $z^{LPM}(S_2) = 1$. Then the node is pruned by bound, and as there are no outstanding nodes, the incumbent solution $\lambda_1 = \lambda_6 = 1$ is optimal. This corresponds to one cluster containing node 1, another containing nodes 2,3, and the third necessarily empty. ∎

11.7 NOTES

11.1 The fundamental paper on the decomposition of linear programs, known as Dantzig-Wolfe decomposition, is [DanWol60]. Recent surveys in this area include [Barnetal94] and [Desetal95].

11.3 The first use of column generation to solve the Master linear program arising from an integer programming problem is probably the work on the cutting stock problem [GilGom61], [GilGom63]. The equivalence of the bounds provided by the linear programming Master and the Lagrangian dual has been known since [Geo74].

11.4 The first papers on integer programmingt column generation appeared in the eighties [DesSouDes84],[DesSou89] on routing problems in which the subproblems are constrained shortest path problems that are solved by dynamic programming.

11.5 The multi-item lot-sizing and clustering problems have been tackled by integer programming decomposition in [Vdbeck94], and the clustering problem in [JohMehNem93], the generalized assignment problem in [Sav93], and binary and integer cutting stock problems in [Vancetal94] and [Vdbeck96] respectively.

11.6 The branching rule (ii) is from [RyaFos81]. Recent more general branching rules that are not restricted to 0–1 problems appear in [Barnetal94] and [VdbeckWol96].

In [Ben62] an alternative resource-based reformulation and decomposition approach is proposed; see Exercise 11.5.

11.8 EXERCISES

1. Consider the following instance of UFL with $m = 4, n = 3$,

$$(c_{ij}) = \begin{pmatrix} 2 & 1 & 5 \\ 3 & 4 & 2 \\ 6 & 4 & 1 \\ 1 & 3 & 7 \end{pmatrix} \text{ and } f = (8, 6, 5).$$

Reformulate using an Integer Programming Master Problem. Solve the Linear Programming Master by column generation.

2. Solve the following instance of $STSP$ by column generation

$$(c_e) = \begin{pmatrix} - & 3 & 4 & 2 \\ - & - & 5 & 6 \\ - & - & - & 12 \\ - & - & - & - \end{pmatrix}.$$

3. Consider GAP with equality constraints

$$\begin{aligned} \max \textstyle\sum_{i=1}^m \sum_{j=1}^n c_{ij}x_{ij} \\ \textstyle\sum_{j=1}^n x_{ij} = 1 \text{ for } i = 1, \ldots, m \\ \textstyle\sum_{i=1}^m a_{ij}x_{ij} \leq b_j \text{ for } j = 1, \ldots, n \\ x \in B^{mn}. \end{aligned}$$

Solve an instance with $m = 3, n = 2$, $(c_{ij}) = (a_{ij}) = \begin{pmatrix} 2 & 1 \\ 1 & 1 \\ 1 & 2 \end{pmatrix}$, and $b = \begin{pmatrix} 2 \\ 2 \end{pmatrix}$ by inter programming decomposition. Solve also by Lagrangian relaxation and by cutting planes and compare.

4. Formulate the Integer Programming Master and subproblems for the three problems presented in Section 11.5.

5.* (Benders' Reformulation). Use the results of Exercise 1.15 to show that

$$(MIP) \qquad z = \max\{cx + hy : Ax + Gy \leq b, x \in R^n_+, y \in Y \subseteq R^p_+\}$$

has the equivalent formulation

$$\begin{aligned} z = \max \eta \\ \eta \leq u^s(b - Gy) + hy \text{ for } s = 1, \ldots, S \\ v^t(b - Gy) \geq 0 \text{ for } t = 1, \ldots, T \\ y \in Y. \end{aligned}$$