To appear in *Column Generation*, G. Desaulniers, J. Desrosiers, and M.M. Solomon (Eds.), Kluwer, 2005.

Chapter 1

A PRIMER IN COLUMN GENERATION

Jacques Desrosiers
Marco E. Lübbecke

Abstract We give a didactic introduction to the use of the column generation technique in linear and in particular in integer programming. We touch on both, the relevant basic theory and more advanced ideas which help in solving large scale practical problems. Our discussion includes embedding Dantzig-Wolfe decomposition and Lagrangian relaxation within a branch-and-bound framework, deriving natural branching and cutting rules by means of a so-called compact formulation, and understanding and influencing the behavior of the dual variables during column generation. Most concepts are illustrated via a small example. We close with a discussion of the classical cutting stock problem and some suggestions for further reading.

1. Hands-On Experience

Let us start right away by solving a constrained shortest path problem. Consider the network depicted in Figure 1.1. Besides a cost c_{ij} there is a resource consumption t_{ij} attributed to each arc $(i, j) \in A$, say a traversal time. Our goal is to find a shortest path from node 1 to node 6 such that the total traversal time of the path does not exceed 14 time units.

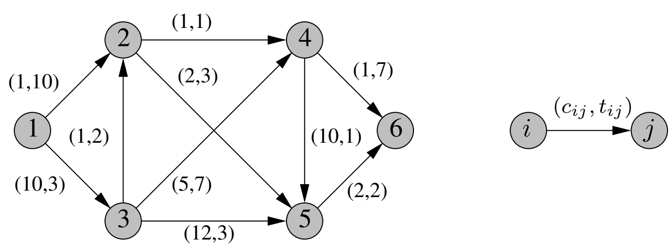

Figure 1.1. Time constrained shortest path problem, Ahuja et al., 1993, p. 599

One way to state this particular network flow problem is as the integer program (1.1)–(1.6). One unit of flow has to leave the source (1.2) and has to enter the sink (1.4), while flow conservation (1.3) holds at all other nodes. The time resource constraint appears as (1.5).

$$z^\star := \min \sum_{(i,j)\in A} c_{ij}x_{ij} \tag{1.1}$$

$$\text{subject to} \sum_{j:(1,j)\in A} x_{1j} = 1 \tag{1.2}$$

$$\sum_{j:(i,j)\in A} x_{ij} - \sum_{j:(j,i)\in A} x_{ji} = 0 \qquad i = 2,3,4,5 \tag{1.3}$$

$$\sum_{i:(i,6)\in A} x_{i6} = 1 \tag{1.4}$$

$$\sum_{(i,j)\in A} t_{ij}x_{ij} \leq 14 \tag{1.5}$$

$$x_{ij} = 0 \text{ or } 1 \quad (i,j) \in A \tag{1.6}$$

An inspection shows that there are nine possible paths, three of which consume too much time. The optimal integer solution is path 13246 of cost 13 with a traversal time of 13. How would we find this out? First note that the resource constraint (1.5) prevents us from solving our problem with a classical shortest path algorithm. In fact, no polynomial time algorithm is likely to exist since the resource constrained shortest path problem is $\mathcal{NP}$-hard. However, since the problem is *almost* a shortest path problem, we would like to exploit this embedded well-studied structure algorithmically.

1.1 An Equivalent Reformulation: Arcs vs. Paths

If we ignore the complicating constraint (1.5), the easily tractable remainder is $X = \{x_{ij} = 0 \text{ or } 1 \mid (1.2)\text{–}(1.4)\}$. It is a well-known result in network flow theory that an extreme point $\mathbf{x}_p = (x_{pij})$ of the polytope defined by the convex hull of X corresponds to a path $p \in P$ in the network. This enables us to express any arc flow as a convex combination of path flows:

$$x_{ij} = \sum_{p\in P} x_{pij}\lambda_p \quad (i,j) \in A \tag{1.7}$$

$$\sum_{p\in P} \lambda_p = 1 \tag{1.8}$$

$$\lambda_p \geq 0 \quad p \in P. \tag{1.9}$$

If we substitute for $\mathbf{x}$ in (1.1) and (1.5) we obtain the so-called *master problem*:

$$z^\star = \min \sum_{p\in P} (\sum_{(i,j)\in A} c_{ij} x_{pij}) \lambda_p \tag{1.10}$$

$$\text{subject to } \sum_{p\in P} (\sum_{(i,j)\in A} t_{ij} x_{pij}) \lambda_p \leq 14 \tag{1.11}$$

$$\sum_{p\in P} \lambda_p = 1 \tag{1.12}$$

$$\lambda_p \geq 0 \qquad p \in P \tag{1.13}$$

$$\sum_{p\in P} x_{pij} \lambda_p = x_{ij} \qquad (i,j) \in A \tag{1.14}$$

$$x_{ij} = 0 \text{ or } 1 \quad (i,j) \in A\,. \tag{1.15}$$

Loosely speaking, the structural information X that we are looking for a path is hidden in "$p \in P$." The cost coefficient of λ_p is the cost of path p and its coefficient in (1.11) is path p's duration. Via (1.14) and (1.15) we explicitly preserve the linking of variables (1.7) in the formulation, and we may recover a solution $\mathbf{x}$ to our *original* problem (1.1)–(1.6) from a master problem's solution. Always remember that integrality must hold for the original $\mathbf{x}$ variables.

1.2 The Linear Relaxation of the Master Problem

One starts with solving the linear programming (LP) relaxation of the master problem. If we relax (1.15), there is no longer a need to link the $\mathbf{x}$ and $\boldsymbol{\lambda}$ variables, and we may drop (1.14) as well. There remains a problem with nine path variables and two constraints. Associate with (1.11) and (1.12) dual variables π_1 and π_0, respectively. For large networks, the cardinality of P becomes prohibitive, and we cannot even explicitly state all the variables of the master problem. The appealing idea of *column generation* is to work only with a sufficiently meaningful subset of variables, forming the so-called *restricted master problem* (RMP). More variables are added only when needed: like in the simplex method we have to find in every iteration a promising variable to enter the basis. In column generation an iteration consists (a) of optimizing the restricted master problem in order to determine the current optimal objective function value $\bar{z}$ and dual multipliers $\boldsymbol{\pi}$, and (b) of finding, if there still is one, a variable λ_p with *negative reduced cost*

$$\bar{c}_p = \sum_{(i,j)\in A} c_{ij} x_{pij} - \pi_1 (\sum_{(i,j)\in A} t_{ij} x_{pij}) - \pi_0 \ < 0. \tag{1.16}$$

The implicit search for a minimum reduced cost variable amounts to optimizing a *subproblem*, precisely in our case: a shortest path problem in the network of

Figure 1.1 with a modified cost structure:

$$\bar{c}^\star = \min_{(1.2)\text{–}(1.4),\,(1.6)} \sum_{(i,j)\in A} (c_{ij} - \pi_1 t_{ij}) x_{ij} - \pi_0. \tag{1.17}$$

Clearly, if $\bar{c}^\star \geq 0$ there is no improving variable and we are done with the linear relaxation of the master problem. Otherwise, the variable found is added to the RMP and we repeat.

In order to obtain integer solutions to our original problem, we have to embed column generation within a branch-and-bound framework. We now give full numerical details of the solution of our particular instance. We denote by BB$n.i$ iteration number i at node number n ($n = 0$ represents the root node). The summary in Table 1.1 for the LP relaxation of the master problem also lists the cost c_p and duration t_p of path p, respectively, and the solution in terms of the value of the original variables $\mathbf{x}$.

Table 1.1. BB0: The linear programming relaxation of the master problem

Iteration	Master Solution	$\bar{z}$	π_0	π_1	$\bar{c}^\star$	p	c_p	t_p
BB0.1	$y_0 = 1$	100.0	100.00	0.00	-97.0	1246	3	18
BB0.2	$y_0 = 0.22, \lambda_{1246} = 0.78$	24.6	100.00	-5.39	-32.9	1356	24	8
BB0.3	$\lambda_{1246} = 0.6, \lambda_{1356} = 0.4$	11.4	40.80	-2.10	-4.8	13256	15	10
BB0.4	$\lambda_{1246} = \lambda_{13256} = 0.5$	9.0	30.00	-1.50	-2.5	1256	5	15
BB0.5	$\lambda_{13256} = 0.2, \lambda_{1256} = 0.8$	7.0	35.00	-2.00	0			
Arc flows: $x_{12} = 0.8, x_{13} = x_{32} = 0.2, x_{25} = x_{56} = 1$								

Since we have no feasible initial solution at iteration BB0.1, we adopt a big-M approach and introduce an artificial variable y_0 with a large cost, say 100, for the convexity constraint. We do not have any path variables yet and the RMP contains two constraints and the artificial variable. This problem is solved by inspection: $y_0 = 1, \bar{z} = 100$, and the dual variables are $\pi_0 = 100$ and $\pi_1 = 0$. The subproblem (1.17) returns path 1246 at reduced cost $\bar{c}^\star = -97$, cost 3 and duration 18. In iteration BB0.2, the RMP contains two variables: y_0 and λ_{1246}. An optimal solution with $\bar{z} = 24.6$ is $y_0 = 0.22$ and $\lambda_{1246} = 0.78$, which is still infeasible. The dual variables assume values $\pi_0 = 100$ and $\pi_1 = -5.39$. Solving the subproblem gives the feasible path 1356 of reduced cost -32.9, cost 24, and duration 8.

In total, four path variables are generated during the column generation process. In iteration BB0.5, we use 0.2 times the feasible path 13256 and 0.8 times the infeasible path 1256. The optimal objective function value is 7, with $\pi_0 = 35$ and $\pi_1 = -2$. The arc flow values provided at the bottom of Table 1.1 are identical to those found when solving the LP relaxation of the original problem.

1.3 Branch-and-Bound: The Reformulation Repeats

Except for the integrality requirement (1.6) (or 1.15) all constraints of the original (and of the master) problem are satisfied, and a subsequent branch-and-bound process is used to compute an optimal integer solution. Even though it cannot happen for our example problem, in general the generated set of columns may not contain an integer feasible solution. To proceed, we have to start the reformulation and column generation again in each node.

Let us first explore some "standard" ways of branching on fractional variables, e.g., branching on $x_{12} = 0.8$. For $x_{12} = 0$, the impact on the RMP is that we have to remove path variables λ_{1246} and λ_{1256}, that is, those paths which contain arc $(1, 2)$. In the subproblem, this arc is removed from the network. When the RMP is re-optimized, the artificial variable assumes a positive value, and we would have to generate new $\boldsymbol{\lambda}$ variables. On branch $x_{12} = 1$, arcs $(1, 3)$ and $(3, 2)$ cannot be used. Generated paths which contain these arcs are discarded from the RMP, and both arcs are removed from the subproblem.

There are also many strategies involving more than a single arc flow variable. One is to branch on the sum of all flow variables which currently is 3.2. Since the solution is a path, an integer number of arcs has to be used, in fact, at least three and at most five in our example. Our freedom of making branching decisions is a powerful tool when properly applied.

Alternatively, we branch on $x_{13} + x_{32} = 0.4$. On branch $x_{13} + x_{32} = 0$, we simultaneously treat two flow variables; impacts on the RMP and the subproblem are similar to those described above. On branch $x_{13} + x_{32} \geq 1$, this constraint is first added to the original formulation. We exploit again the path substructure X, go through the reformulation process via (1.7), and obtain a new RMP to work with. Details of the search tree are summarized in Table 1.2.

At node BB1, we set $x_{13} + x_{32} = 0$. In iteration BB1.1, paths 1356 and 13256 are discarded from the RMP, and arcs $(1, 3)$ and $(3, 2)$ are removed from the subproblem. The resulting RMP with $y_0 = 0.067$ and $\lambda_{1256} = 0.933$ is infeasible. The objective function assumes a value $\bar{z} = 11.3$, and $\pi_0 = 100$ and $\pi_1 = -6.33$. Given these dual multipliers, no column with negative reduced cost can be generated!

Here we face a drawback of the big-M approach. Path 12456 is feasible, its duration is 14, but its cost of 14 is larger than the current objective function value, computed as $0.067M$ + 0.933×5. The constant $M = 100$ is too small, and we have to increase it, say to 1000. (A different phase I approach, that is, minimizing the artificial variable y_0, would have easily prevented this.) Re-optimizing the RMP in iteration BB1.2 now results in $\bar{z} = 71.3$, $y_0 = 0.067$, $\lambda_{1256} = 0.933$, $\pi_0 = 1000$, and $\pi_1 = -66.33$. The subproblem returns path 12456 with a reduced cost of -57.3. In iteration BB1.3, the new RMP has an integer solution $\lambda_{12456} = 1$, with $\bar{z} = 14$, an upper bound on the optimal

Table 1.2. Details of the branch-and-bound decisions

Iteration	Master Solution	$\bar{z}$	π_0	π_1	π_2	$\bar{c}^\star$	p	c_p	t_p
BB1: BB0 and $x_{13}+x_{32}=0$									
BB1.1	$y_0=0.067, \lambda_{1256}=0.933$	11.3	100	-6.33	–	0			
BB1.2	Big-M increased to 1000								
	$y_0=0.067, \lambda_{1256}=0.933$	71.3	1000	-66.33	–	-57.3	12456	14	14
BB1.3	$\lambda_{12456}=1$	**14**	1000	-70.43	–	0			
BB2: BB0 and $x_{13}+x_{32}\geq 1$									
BB2.1	$\lambda_{1246}=\lambda_{13256}=0.5$	9	15	-0.67	3.33	0			
Arc flows: $x_{12}=x_{13}=x_{24}=x_{25}=x_{32}=x_{46}=x_{56}=0.5$									
BB3: BB2 and $x_{12}=0$									
BB3.1	$\lambda_{13256}=1$	**15**	15	0	0	-2	13246	13	13
BB3.2	$\lambda_{13246}=1$	**13**	13	0	0	0			
BB4: BB2 and $x_{12}=1$									
BB4.1	$y_0=0.067, \lambda_{1256}=0.933$	111.3	100	-6.33	100	0			
Infeasible arc flows									

path cost. The dual multipliers are $\pi_0 = 1000$ and $\pi_1 = -70.43$, and no new variable is generated.

At node BB2, we impose $x_{13}+x_{32} \geq 1$ to the original formulation, and again, we reformulate these $\mathbf{x}$ variables in terms of the $\boldsymbol{\lambda}$ variables. The resulting new constraint (with dual multiplier π_2) in the RMP is $\sum_{p\in P}(x_{p13}+x_{p32})\lambda_p \geq 1$. From the value of $(x_{p13}+x_{p32})$ we learn how often arcs $(1,3)$ and $(3,2)$ are used in path p. The current problem at node BB2.1 is the following:

$$
\begin{array}{lrcrcrcrcrcll}
\text{min} & 100y_0 & + & 3\lambda_{1246} & + & 24\lambda_{1356} & + & 15\lambda_{13256} & + & 5\lambda_{1256} & & & \\
\text{subject to:} & & & 18\lambda_{1246} & + & 8\lambda_{1356} & + & 10\lambda_{13256} & + & 15\lambda_{1256} & \leq & 14 & [\pi_1] \\
 & & & & & \lambda_{1356} & + & 2\lambda_{13256} & & & \geq & 1 & [\pi_2] \\
 & y_0 & + & \lambda_{1246} & + & \lambda_{1356} & + & \lambda_{13256} & + & \lambda_{1256} & = & 1 & [\pi_0] \\
 & y_0, & & \lambda_{1246}, & & \lambda_{1356}, & & \lambda_{13256}, & & \lambda_{1256} & \geq & 0 &
\end{array}
$$

From solving this linear program we obtain an increase in the objective function $\bar{z}$ from 7 to 9 with variables $\lambda_{1246} = \lambda_{13256} = 0.5$, and dual multipliers $\pi_0 = 15, \pi_1 = -0.67$, and $\pi_2 = 3.33$. The new subproblem is given by

$$
\bar{c}^\star = \min_{(1.2)\text{–}(1.4),\,(1.6)} \sum_{(i,j)\in A} (c_{ij} - \pi_1 t_{ij})x_{ij} - \pi_0 - \pi_2(x_{13}+x_{32}). \qquad (1.18)
$$

For these multipliers no path of negative reduced cost exists. The solution of the flow variables is $x_{12} = x_{13} = x_{24} = x_{25} = x_{32} = x_{46} = x_{56} = 0.5$.

Next, we arbitrarily choose variable $x_{12} = 0.5$ to branch on. Two iterations are needed when x_{12} is set to zero. In iteration BB3.1, path variables λ_{1246}

and λ_{1256} are discarded from the RMP and arc $(1, 2)$ is removed from the subproblem. The RMP is integer feasible with $\lambda_{13256} = 1$ at cost 15. Dual multipliers are $\pi_0 = 15, \pi_1 = 0$, and $\pi_2 = 0$. Path 13246 of reduced cost -2, cost 13 and duration 13 is generated and used in the next iteration BB3.2. Again the RMP is integer feasible with path variable $\lambda_{13246} = 1$ and a new best integer solution at cost 13, with dual multipliers $\pi_0 = 15, \pi_1 = 0$, and $\pi_2 = 0$ for which no path of negative reduced cost exists.

On the alternative branch $x_{12} = 1$ the RMP is optimal after a single iteration. In iteration BB4.1, variable x_{13} can be set to zero and variables $\lambda_{1356}, \lambda_{13256}$, and λ_{13246} are discarded from the current RMP. After the introduction of an artificial variable y_2 in the second row, the RMP is infeasible since $y_0 > 0$ (as can be seen also from the large objective function value $\bar{z} = 111.3$). Given the dual multipliers, no columns of negative reduced cost can be generated, and the RMP remains infeasible. The optimal solution (found at node BB3) is path 13246 of cost 13 with a duration of 13 as well.

2. Some Theoretical Background

In the previous example we already saw all the necessary building blocks for a column generation based solution approach to integer programs: (1) an original formulation to solve which acts as the control center to facilitate the design of natural branching rules and cutting planes; (2) a master problem to determine the currently optimal dual multipliers and to provide a lower bound at each node of the branch-and-bound tree; (3) a pricing subproblem which explicitly reflects an embedded structure we wish to exploit. In this section we detail the underlying theory.

2.1 Column Generation

Let us call the following linear program the *master problem* (MP).

$$\begin{aligned} z^\star_{MP} := \min \quad & \sum_{j \in J} c_j \lambda_j \\ \text{subject to} \quad & \sum_{j \in J} \mathbf{a}_j \lambda_j \geq \mathbf{b} \\ & \lambda_j \geq 0, \; j \in J. \end{aligned} \tag{1.19}$$

In each iteration of the simplex method we look for a non-basic variable to price out and enter the basis. That is, given the non-negative vector $\boldsymbol{\pi}$ of dual variables we wish to find a $j \in J$ which minimizes $\bar{c}_j := c_j - \boldsymbol{\pi}^t \mathbf{a}_j$. This explicit pricing is a too costly operation when $|J|$ is huge. Instead, we work with a reasonably small subset $J' \subseteq J$ of columns—the restricted master problem (RMP)—and evaluate reduced costs only by implicit enumeration. Let $\boldsymbol{\lambda}$ and $\boldsymbol{\pi}$ assume primal and dual optimal solutions of the current RMP, respectively. When columns $\mathbf{a}_j$, $j \in J$, are given as elements of a set $\mathcal{A}$, and the cost coefficient c_j can be computed from $\mathbf{a}_j$ via a function c then the subproblem

$$\bar{c}^\star := \min \left\{ c(\mathbf{a}) - \boldsymbol{\pi}^t \mathbf{a} \mid \mathbf{a} \in \mathcal{A} \right\} \tag{1.20}$$

performs the pricing. If $\bar{c}^\star \geq 0$, there is no negative $\bar{c}_j$, $j \in J$, and the solution $\boldsymbol{\lambda}$ to the restricted master problem optimally solves the master problem as well. Otherwise, we add to the RMP the column derived from the optimal subproblem solution, and repeat with re-optimizing the RMP. The process is initialized with an artificial, a heuristic, or a previous ("warm start") solution. In what regards convergence, note that each $\mathbf{a} \in \mathcal{A}$ is generated at most once since no variable in an optimal RMP has negative reduced cost. When dealing with some finite set $\mathcal{A}$ (as is practically always true), the column generation algorithm is exact. In addition, we can make use of bounds. Let $\bar{z}$ denote the optimal objective function value to the RMP. When an upper bound $\kappa \geq \sum_{j \in J} \lambda_j$ holds for the optimal solution of the master problem, we have not only an upper bound $\bar{z}$ on $z^\star_{MP}$ in each iteration, but also a lower bound: we cannot reduce $\bar{z}$ by more than κ times the smallest reduced cost $\bar{c}^\star$:

$$\bar{z} + \kappa \bar{c}^\star \leq z^\star_{MP} \leq \bar{z}. \tag{1.21}$$

Thus, we may verify the solution quality at any time. In the optimum of (1.19), $\bar{c}^\star = 0$ for the basic variables, and $\bar{z} = z^\star_{MP}$.

2.2 Dantzig-Wolfe Decomposition for Integer Programs

In many applications we are interested in optimizing over a discrete set X. For $X = \{\mathbf{x} \in \mathbb{Z}^n_+ \mid D\mathbf{x} \geq \mathbf{d}\} \neq \emptyset$ we have the special case of integer linear programming. Consider the following (*original* or *compact*) program:

$$\begin{array}{rrcl} z^\star := \min & \mathbf{c}^t \mathbf{x} & & \\ \text{subject to} & A\mathbf{x} & \geq & \mathbf{b} \\ & \mathbf{x} & \in & X. \end{array} \tag{1.22}$$

Replacing X by $\mathrm{conv}(X)$ in (1.22) does not change $z^\star$ which we assume to be finite. The Minkowski and Weyl theorems (see Schrijver, 1986) enable us to represent each $\mathbf{x} \in X$ as a convex combination of extreme points $\{\mathbf{x}_p\}_{p \in P}$ plus a non-negative combination of extreme rays $\{\mathbf{x}_r\}_{r \in R}$ of $\mathrm{conv}(X)$, i.e.,

$$\mathbf{x} = \sum_{p \in P} \mathbf{x}_p \lambda_p + \sum_{r \in R} \mathbf{x}_r \lambda_r, \quad \sum_{p \in P} \lambda_p = 1, \quad \boldsymbol{\lambda} \in \mathbb{R}^{|P|+|R|}_+ \tag{1.23}$$

where the index sets P and R are finite. Substituting for $\mathbf{x}$ in (1.22) and applying the linear transformations $c_j = \mathbf{c}^t \mathbf{x}_j$ and $\mathbf{a}_j = A\mathbf{x}_j$, $j \in P \cup R$ we obtain an equivalent *extensive formulation*

$$z^\star := \min \sum_{p\in P} c_p\lambda_p + \sum_{r\in R} c_r\lambda_r \tag{1.24}$$

$$\text{subject to } \sum_{p\in P} \mathbf{a}_p\lambda_p + \sum_{r\in R} \mathbf{a}_r\lambda_r \geq \mathbf{b} \tag{1.25}$$

$$\sum_{p\in P} \lambda_p = 1 \tag{1.26}$$

$$\boldsymbol{\lambda} \geq \mathbf{0} \tag{1.27}$$

$$\sum_{p\in P} \mathbf{x}_p\lambda_p + \sum_{r\in R} \mathbf{x}_r\lambda_r = \mathbf{x} \tag{1.28}$$

$$\mathbf{x} \in \mathbb{Z}_+^n. \tag{1.29}$$

Equation (1.26) is referred to as the *convexity constraint*. When we relax the integrality of $\mathbf{x}$, there is no need to link $\mathbf{x}$ and $\boldsymbol{\lambda}$, and we may also relax (1.28). The columns of this special master problem are defined by the extreme points and extreme rays of $\mathrm{conv}(X)$. We solve the master by column generation to get its optimal objective function value $z^\star_{MP}$. Given an optimal dual solution $\boldsymbol{\pi}$ and π_0 to the current RMP, where variable π_0 corresponds to the convexity constraint, the subproblem is to determine $\min_{j\in P}\{c_j - \boldsymbol{\pi}^t\mathbf{a}_j - \pi_0\}$ and $\min_{j\in R}\{c_j - \boldsymbol{\pi}^t\mathbf{a}_j\}$. By our previous linear transformation and since π_0 is a constant, this results in

$$\bar{c}^\star \quad := \quad \min\left\{(\mathbf{c}^t - \boldsymbol{\pi}^t A)\mathbf{x} - \pi_0 \mid \mathbf{x} \in X\right\}. \tag{1.30}$$

This subproblem is an integer linear program. When $\bar{c}^\star \geq 0$, there is no negative reduced cost column, and the algorithm terminates. When $\bar{c}^\star < 0$ and finite, an optimal solution to (1.30) is an extreme point $\mathbf{x}_p$ of $\mathrm{conv}(X)$, and we add the column $[\mathbf{c}^t\mathbf{x}_p, (A\mathbf{x}_p)^t, 1]^t$ to the RMP. When $\bar{c}^\star = -\infty$ we identify an extreme ray $\mathbf{x}_r$ of $\mathrm{conv}(X)$ as a solution $\mathbf{x} \in X$ to $(\mathbf{c}^t - \boldsymbol{\pi}^t A)\mathbf{x} = 0$, and add the column $[\mathbf{c}^t\mathbf{x}_r, (A\mathbf{x}_r)^t, 0]^t$ to the RMP.

From (1.21) together with the convexity constraint we obtain in each iteration

$$\bar{z} + \bar{c}^\star \leq z^\star_{MP} \leq \bar{z}, \tag{1.31}$$

where $\bar{z} = \boldsymbol{\pi}^t\mathbf{b} + \pi_0$ is again the optimal objective function value of the RMP. Since $z^\star_{MP} \leq z^\star$, $\bar{z} + \bar{c}^\star$ is also a lower bound on $z^\star$. In general, $\bar{z}$ is not a valid upper bound on $z^\star$, except if the current $\mathbf{x}$ variables are integer. The algorithm is exact and finite as long as finiteness is ensured in optimizing the RMP.

The original formulation is the starting point to obtain integer solutions in the $\mathbf{x}$ variables. Branching and cutting constraints are added there, the reformulation as in Section 1.1.1 is re-applied, and the process continues with an updated master problem. It is important to see that it is our choice as to whether the additional constraints remain in the master problem (as in the previous section) or go into the subproblem (as we will see later).

Pricing Out the Original $\mathbf{x}$ Variables. Assume that in (1.22) we have a linear subproblem $X = \{\mathbf{x} \in \mathbb{R}^n_+ \mid D\mathbf{x} \geq \mathbf{d}\} \neq \emptyset$. Column generation then essentially solves the linear program

$$\min\ \mathbf{c}^t\mathbf{x} \quad \text{subject to} \quad A\mathbf{x} \geq \mathbf{b},\ D\mathbf{x} \geq \mathbf{d},\ \mathbf{x} \geq \mathbf{0}.$$

We obtain an optimal primal solution $\mathbf{x}$ but only the dual multipliers $\boldsymbol{\pi}$ associated with the constraint set $A\mathbf{x} \geq \mathbf{b}$. However, following an idea of Walker, 1969 we can also retrieve the dual variables $\boldsymbol{\sigma}$ associated with $D\mathbf{x} \geq d$: it is the vector obtained from solving the linear subproblem in the last iteration of the column generation process. This full dual information allows for a pricing of the original variables, and therefore a possible elimination of some of them. Given an upper bound on the integer optimal objective function value of the original problem, one can eliminate an x variable if its reduced cost is larger than the optimality gap.

In the general case of a linear integer or even non-linear pricing subproblem, the above procedure does not work. Poggi de Aragão and Uchoa, 2003 suggest to directly use the extensive formulation: if we keep the coupling constraint (1.28) in the master problem, it suffices to impose the constraint $\mathbf{x} \geq \boldsymbol{\epsilon}$, for a small $\boldsymbol{\epsilon} > \mathbf{0}$, at the end of the process. The shadow prices of these constraints are the reduced costs of the $\mathbf{x}$ vector of original variables. Note that there is no need to apply the additional constraints to already positive variables. Computational experiments underline the benefits of this procedure.

Block Diagonal Structure. For practical problems Dantzig-Wolfe decomposition can typically exploit a block diagonal structure of D, i.e.,

$$D = \begin{pmatrix} D^1 & & & \\ & D^2 & & \\ & & \ddots & \\ & & & D^\kappa \end{pmatrix} \qquad \mathbf{d} = \begin{pmatrix} \mathbf{d}^1 \\ \mathbf{d}^2 \\ \vdots \\ \mathbf{d}^\kappa. \end{pmatrix} \tag{1.32}$$

Each $X^k = \{D^k\mathbf{x}^k \geq \mathbf{d}^k,\ \mathbf{x}^k \geq \mathbf{0} \text{ and integer}\}$, $k \in K := \{1, \ldots, \kappa\}$, gives rise to a representation as in (1.23). The decomposition yields κ subproblems, each with its own convexity constraint and associated dual variable:

$$\bar{c}^{k\star} := \min\{(\mathbf{c}^{kT} - \boldsymbol{\pi}^t A^k)\mathbf{x}^k - \pi_0^k \mid \mathbf{x}^k \in X^k\}, \quad k \in K. \tag{1.33}$$

The superscript k to all entities should be interpreted in the canonical way. The algorithm terminates when $\bar{c}^{k\star} \geq 0$, for all $k \in K$. Otherwise, extreme points and rays identified in (1.33) give rise to new columns to be added to the RMP. By linear programming duality, $\bar{z} = \boldsymbol{\pi}^t \mathbf{b} + \sum_{k=1}^{\kappa} \pi_0^k$, and we obtain the following bounds, see Lasdon, 1970:

$$\bar{z} + \sum_{k=1}^{\kappa} \bar{c}^{k\star} \leq z^\star_{MP} \leq \bar{z}. \tag{1.34}$$

2.3 Useful Working Knowledge

When problems get larger and computationally much more difficult than our small constrained shortest path problem it is helpful to know more about mechanisms, their consequences, and how to exploit them.

Infeasible paths. One may wonder why we kept infeasible paths in the RMP during column generation. Here, as for the whole process, we cannot overemphasize the fact that knowledge about the *integer* solution usually does not help us much in solving the *linear relaxation* program. Figure 1.2 illustrates the domain of the RMP (shaded) and the domain X of the subproblem. In part (a), the optimal solution $\mathbf{x}$, symbolized by the dot, is uniquely determined as a convex combination of the three extreme points of the triangle X, even though all of them are not feasible for the intersection of the master and subproblem. In our example, in iteration BB0.5, any convex combination of feasible paths which have been generated, namely 13256 and 1356, has cost larger than 7, i.e., is suboptimal for the linear relaxation of the master problem. Infeasible paths are removed only if needed during the search for an integer solution. In Figure 1.2 (a), $\mathbf{x}$ can be integer and no branch-and-bound search is needed.

In part (b) there are many ways to express the optimal solution as a convex combination of three extreme points. This is a partial explanation of the slow convergence (*tailing off*) of linear programming column generation.

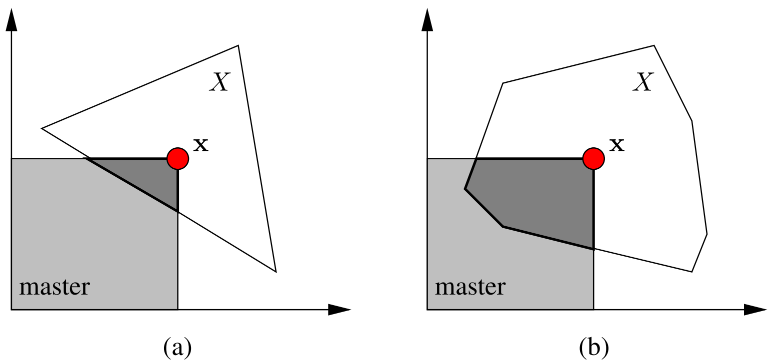

Figure 1.2. Schematic illustration of the domains of master and subproblem X

Lower and Upper Bounds. Figure 1.3 gives the development of upper ($\bar{z}$) and lower ($\bar{z} + \bar{c}^{*}$) bounds on $z^{\star}_{MP}$ in the root node for our small constrained shortest path example. The values for the lower bound are 3.0, -8.33, 6.6, 6.5, and finally 7. While the upper bound decreases monotonically (as expected when minimizing a linear program) there is no monotony for the lower bound. Still, we can use these bounds to evaluate the quality of the current solution by computing the optimality gap, and could stop the process when a preset quality is reached. Is there any use of the bounds beyond that? Note first that $UB = \bar{z}$

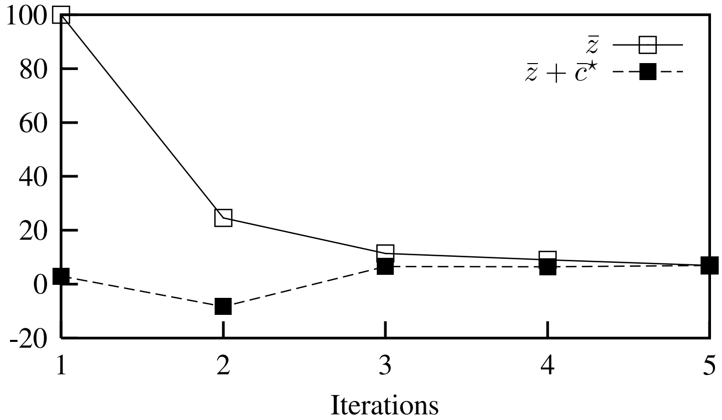

Figure 1.3. Development of lower and upper bounds on z_{MP} in BB0

is not an upper bound on $z^{\star}$. The currently (over all iterations) best lower bound LB, however, is a lower bound on $z^{\star}_{MP}$ and on $z^{\star}$. Even though there is no direct use of LB or UB in the master problem we can impose the additional constraints $LB \leq \mathbf{c}^t\mathbf{x} \leq UB$ to the subproblem structure X if the subproblem consists of a single block. Be aware that this cutting modifies the subproblem structure X, with all algorithmic consequences, that is, possible complications for a combinatorial algorithm. In our constrained shortest path example, two generated paths are feasible and provide upper bounds on the optimal integer solution z^{*}. The best one is path 13256 of cost 15 and duration 10. Table 1.3 shows the multipliers $\sigma_i, i = 1, \ldots, 6$ for the flow conservation constraints of the path structure X at the last iteration of the column generation process.

Table 1.3. Multipliers σ_i for the flow conservation constraints on nodes $i \in N$

node i	1	2	3	4	5	6
σ_i	29	8	13	5	0	-6

Therefore, given the optimal multiplier $\pi_1 = -2$ for the resource constraint, the reduced cost of an arc is given by $\bar{c}_{ij} = c_{ij} - \sigma_i + \sigma_j - t_{ij}\pi_1, \quad (i,j) \in A$. The reader can verify that $\bar{c}_{34} = 3 - 13 + 5 - (7)(-2) = 11$. This is the only reduced cost which exceeds the current optimality gap which equals to $15 - 7 = 8$. Arc $(3,4)$ can be permanently discarded from the network and paths 1346 and 13456 will never be generated.

Integrality Property. Solving the subproblem as an integer program usually helps in closing part of the integrality gap of the master problem (Geoffrion, 1974), except when the subproblem possesses the *integrality property*. This property means that solutions to the pricing problem are naturally integer when it is solved as a linear program. This is the case for our shortest path subproblem and this is why we obtained the value of the linear relaxation of the original problem as the value of the linear relaxation of the master problem.

When looking for an integer solution to the original problem, we need to impose new restrictions on (1.1)–(1.6). One way is to take advantage of a new X structure. However, if the new subproblem is still solved as a linear program, $z^\star_{MP}$ remains 7. Only solving the new X structure as an integer program may improve $z^\star_{MP}$.

Once we understand that we can modify the subproblem structure, we can devise other decomposition strategies. One is to define the X structure as

$$\sum_{(i,j)\in A} t_{ij}x_{ij} \leq 14, \quad x_{ij} \text{ binary}, \quad (i,j) \in A \tag{1.35}$$

so that the subproblem becomes a knapsack problem which does not possess the integrality property. Unfortunately, in this example, $z^\star_{MP}$ remains 7. However, improvements can be obtained by imposing more and more constraints to the subproblem. An example is to additionally enforce the selection of one arc to leave the source (1.2) and another one to enter the sink (1.4), and impose constraint $3 \leq \sum_{(i,j)\in A} x_{ij} \leq 5$ on the minimum and maximum number of selected arcs. Richer subproblems, as long as they can be solved efficiently and do not possess the integrality property, may help in closing the integrality gap.

It is also our decision how much branching and cutting information (ranging from none to all) we put into the subproblem. This choice depends on where the additional constraints are more easily accommodated in terms of algorithms and computational tractability. Branching decisions imposed on the subproblem can reduce its solution space and may turn out to facilitate a solution as integer program. As an illustration we describe adding the lower bound cut in the root node of our small example.

Imposing the Lower Bound Cut $\mathbf{c^t x \geq 7}$. Assume that we have solved the relaxed RMP in the root node and instead of branching, we impose the lower bound cut on the X structure, see Table 1.4. Note that this cut would not have helped in the RMP since $\bar{z} = \mathbf{c}^t\mathbf{x} = 7$ already holds. We start modifying the RMP by removing variables λ_{1246} and λ_{1256} as their cost is smaller than 7. In iteration BB0.6, for the re-optimized RMP $\lambda_{13256} = 1$ is optimal at cost 15; it corresponds to a feasible path of duration 10. UB is updated to 15 and the dual multipliers are $\pi_0 = 15$ and $\pi_1 = 0$. The X structure is modified by adding constraint $\mathbf{c}^t\mathbf{x} \geq 7$. Path 13246 is generated with reduced cost -2, cost 13, and duration 13. The new lower bound is $15 - 2 = 13$. On the downside of this significant improvement is the fact that we have destroyed the pure network structure of the subproblem which we have to solve as an integer program now. We may pass along with this circumstance if it pays back a better bound.

We re-optimize the RMP in iteration BB0.7 with the added variable λ_{13246}. This variable is optimal at value 1 with cost and duration equal to 13. Since this variable corresponds to a feasible path, it induces a better upper bound which is equal to the lower bound: optimality is proven. There is no need to solve the subproblem.

Table 1.4. Lower bound cut added to the subproblem at the end of the root node

Iteration	Master Solution	$\bar{z}$	π_0	π_1	$\bar{c}^\star$	p	c_p	t_p	UB	LB
BB0.6	$\lambda_{13256} = 1$	**15**	15	0	-2	13246	13	13	15	13
BB0.7	$\lambda_{13246} = 1$	**13**	13	0					13	13

Note that using the dynamically adapted lower bound cut right from the start has an impact on the solution process. For example, the first generated path 1246 would be eliminated in iteration BB0.3 since the lower bound reaches 6.6, and path 1256 is never generated. Additionally adding the upper bound cut has a similar effect.

Acceleration Strategies. Often acceleration techniques are key elements for the viability of the column generation approach. Without them, it would have been almost impossible to obtain quality solutions to various applications, in a reasonable amount of computation time. We sketch here only some strategies, see e.g., Desaulniers et al., 2001 for much more.

The most widely used strategy is to return to the RMP many negative reduced cost columns in each iteration. This generally decreases the number of column generation iterations, and is particularly easy when the subproblem is solved by dynamic programming. When the number of variables in the RMP becomes too large, non-basic columns with current reduced cost exceeding a given threshold may be removed. Accelerating the pricing algorithm itself usually yields most significant speed-ups. Instead of investing in a most negative reduced cost column, *any* variable with negative reduced cost suffices. Often, such a column can be obtained heuristically or from a pool of columns containing not yet used columns from previous calls to the subproblem. In the case of many subproblems, it is often beneficial to consider only few of them each time the pricing problem is called. This is the well-known partial pricing. Finally, in order to reduce the tailing off behavior of column generation, a heuristic rule can be devised to prematurely stop the linear relaxation solution process, for example, when the value of the objective function does not improve sufficiently in a given number of iterations. In this case, the approximate LP solution does not necessarily provide a lower bound but using the current dual multipliers, a lower bound can still be computed.

With a careful use of these ideas one may confine oneself with a non-optimal solution in favor of being able to solve much larger problems. This turns column generation into optimization based heuristics which may be used for comparison with other methods for a given class of problems.

3. A Dual Point of View

The dual program of the RMP is a relaxation of the dual of the master problem, since constraints are omitted. Viewing column generation as row generation in the dual, it is a special case of Kelley's cutting plane method from 1961. Recently, this dual perspective attracted considerable attention and we will see that it provides us with several key insights. Observe that the generation process as well as the stopping criteria are driven entirely by the dual multipliers.

3.1 Lagrangian Relaxation

A practically often used dual approach to solving (1.22) is *Lagrangian relaxation*, see Geoffrion, 1974. Penalizing the violation of $A\mathbf{x} \geq \mathbf{b}$ via Lagrangian multipliers $\boldsymbol{\pi} \geq \mathbf{0}$ in the objective function results in the *Lagrangian subproblem* relative to constraint set $A\mathbf{x} \geq \mathbf{b}$

$$L(\boldsymbol{\pi}) := \min_{\mathbf{x} \in X} \ \mathbf{c}^t\mathbf{x} - \boldsymbol{\pi}^t(A\mathbf{x} - \mathbf{b}). \tag{1.36}$$

Since $L(\boldsymbol{\pi}) \leq \min\{\mathbf{c}^t\mathbf{x} - \boldsymbol{\pi}^t(A\mathbf{x} - \mathbf{b}) \mid A\mathbf{x} \geq \mathbf{b}, \mathbf{x} \in X\} \leq z^\star$, $L(\boldsymbol{\pi})$ is a lower bound on $z^\star$. The best such bound on $z^\star$ is provided by solving the *Lagrangian dual problem*

$$\mathcal{L} := \max_{\boldsymbol{\pi} \geq \mathbf{0}} \ L(\boldsymbol{\pi}). \tag{1.37}$$

Note that (1.37) is a problem in the dual space while (1.36) is a problem in $\mathbf{x}$. The Lagrangian function $L(\boldsymbol{\pi}), \boldsymbol{\pi} \geq \mathbf{0}$ is the lower envelope of a family of functions linear in $\boldsymbol{\pi}$, and therefore is a concave function of $\boldsymbol{\pi}$. It is piecewise linear with breakpoints where the optimal solution of $L(\boldsymbol{\pi})$ is not unique. In particular, $L(\boldsymbol{\pi})$ is not differentiable, but only sub-differentiable. The most popular, since very easy to implement, choice to obtain optimal or near optimal multipliers are subgradient algorithms. However, let us describe an alternative computation method, see Nemhauser and Wolsey, 1988. We know that replacing X by $\mathrm{conv}(X)$ in (1.22) does not change $z^\star$ and this will enable us to write (1.37) as a linear program.

When $X = \emptyset$, which may happen during branch-and-bound, then $\mathcal{L} = \infty$. Otherwise, given some multipliers $\boldsymbol{\pi}$, the Lagrangian bound is

$$L(\boldsymbol{\pi}) = \begin{cases} -\infty & \text{if } (\mathbf{c}^t - \boldsymbol{\pi}^t A)\mathbf{x}_r < 0 \text{ for some } r \in R \\ \mathbf{c}^t\mathbf{x}_p - \boldsymbol{\pi}^t(A\mathbf{x}_p - \mathbf{b}) & \text{for some } p \in P \text{ otherwise.} \end{cases}$$

Since we assumed $z^\star$ to be finite, we avoid unboundedness by writing (1.37) as

$$\max_{\boldsymbol{\pi} \geq \mathbf{0}} \min_{p \in P} \mathbf{c}^t\mathbf{x}_p - \boldsymbol{\pi}^t(A\mathbf{x}_p - \mathbf{b}) \text{ such that } (\mathbf{c}^t - \boldsymbol{\pi}^t A)\mathbf{x}_r \geq 0, \forall r \in R,$$

or as a linear program with many constraints

$$\begin{array}{lrcll} \mathcal{L} = \max & \pi_0 & & & \\ \text{subject to} & \boldsymbol{\pi}^t(A\mathbf{x}_p - \mathbf{b}) + \pi_0 & \leq & \mathbf{c}^t\mathbf{x}_p, & p \in P \\ & \boldsymbol{\pi}^t A\mathbf{x}_r & \leq & \mathbf{c}^t\mathbf{x}_r, & r \in R \\ & \boldsymbol{\pi} & \geq & \mathbf{0}. & \end{array} \tag{1.38}$$

The dual of (1.38) reads as the linear relaxation of the master problem (1.24)–(1.29):

$$\begin{array}{lrcl} \mathcal{L} = \min & \sum_{p \in P} \mathbf{c}^t\mathbf{x}_p\lambda_p + \sum_{r \in R} \mathbf{c}^t\mathbf{x}_r\lambda_r & & \\ \text{subject to} & \sum_{p \in P} A\mathbf{x}_p\lambda_p + \sum_{r \in R} A\mathbf{x}_r\lambda_r & \geq & \mathbf{b}\sum_{p \in P}\lambda_p \\ & \sum_{p \in P}\lambda_p & = & 1 \\ & \boldsymbol{\lambda} & \geq & \mathbf{0}. \end{array} \tag{1.39}$$

Observe that for a given vector $\boldsymbol{\pi}$ of multipliers and a constant π_0,

$$L(\boldsymbol{\pi}) = (\boldsymbol{\pi}^t\mathbf{b} + \pi_0) + \min_{\mathbf{x} \in \mathrm{conv}(X)} (\mathbf{c}^t - \boldsymbol{\pi}^t A)\mathbf{x} - \pi_0 \;=\; \bar{z} + \bar{c}^\star,$$

that is, each time the RMP is solved during the Dantzig-Wolfe decomposition, the computed lower bound in (1.31) is the same as the Lagrangian bound, that is, for optimal $\mathbf{x}$ and $\boldsymbol{\pi}$ we have $z^\star = \mathbf{c}^t\mathbf{x} = L(\boldsymbol{\pi})$.

When we apply Dantzig-Wolfe decomposition to (1.22) we satisfy complementary slackness conditions, we have $\mathbf{x} \in \mathrm{conv}(X)$, and we satisfy $A\mathbf{x} \geq \mathbf{b}$. Therefore only the integrality of $\mathbf{x}$ remains to be checked. The situation is different for subgradient algorithms. Given optimal multipliers $\boldsymbol{\pi}$ for (1.37), we can solve (1.36) which ensures that the solution, denoted $\mathbf{x}_\pi$, is (integer) feasible for X and $\boldsymbol{\pi}^t(A\mathbf{x}_\pi - \mathbf{b}) = 0$. Still, we have to check whether the relaxed constraints are satisfied, that is, $A\mathbf{x}_\pi \geq \mathbf{b}$ to prove optimality. If this condition is violated, we have to recover optimality of a primal-dual pair $(\mathbf{x}, \boldsymbol{\pi})$ by branch-and-bound. For many applications, one is able to slightly modify infeasible solutions obtained from the Lagrangian subproblems with only a small degradation of the objective value. Of course these are only approximate solutions to the original problem. We only remark that there are more advanced (non-linear) alternatives to solve the Lagrangian dual like the *bundle method* (Hiriart-Urruty and Lemaréchal, 1993) based on quadratic programming, and the *analytic center cutting plane method* (Goffin and Vial, 1999), an interior point solution approach. However, the performance of these methods is still to be evaluated in the context of integer programming.

3.2 Dual Restriction / Primal Relaxation

Linear programming column generation remained "as is" for a long time. Recently, the dual point of view prepared the ground for technical advances.

Structural Dual Information. Consider a master problem and its dual and assume both are feasible and bounded. In some situations we may have additional knowledge about an optimal dual solution which we may express as additional valid inequalities $F^t\boldsymbol{\pi} \leq \mathbf{f}$ in the dual. To be more precise, we would like to add inequalities which are satisfied by at least one optimal dual solution. Such valid inequalities correspond to additional primal variables $\mathbf{y} \geq \mathbf{0}$ of cost $\mathbf{f}$ that are *not* present in the original master problem. From the primal perspective, we therefore obtain a relaxation. Devising such *dual-optimal inequalities* requires (but also exploits) a specific problem knowledge. This has been successfully applied to the one-dimensional cutting stock problem, see Valério de Carvalho, 2003 and Ben Amor et al., 2003.

Oscillation. It is an important observation that the dual variable values do not develop smoothly but they very much oscillate during column generation. In the first iterations, the RMP contains too few columns to provide any useful dual information, in turn resulting in non useful variables to be added. Initially, often the penalty cost of artificial variables guide the values of dual multipliers (Vanderbeck, 2005 calls this the *heading-in effect*). One observes that the variables of an optimal master problem solution are generated in the last iterations of the process when dual variables finally come close to their respective optimal values. Understandably, dual oscillation has been identified as a major efficiency issue. One way to control this behavior is to impose lower and upper bounds, that is, we constrain the vector of dual variables to lie "in a box" around its current value. As a result, the RMP is modified by adding slack and surplus variables in the corresponding constraints. After re-optimization of the new RMP, if the new dual optimum is attained on the boundary of the box, we have a direction towards which the box should be relocated. Otherwise, the optimum is attained in the box's interior, producing the global optimum. This is the principle of the `Boxstep` method (Marsten, 1975; Marsten et al., 1975).

Stabilization. *Stabilized column generation* (see du Merle et al., 1999; Ben Amor and Desrosiers, 2003) offers more flexibility for controlling the duals. Again, the dual solution $\boldsymbol{\pi}$ is restricted to stay in a box, however, the box may be left at a certain penalty cost. This penalty may be a piecewise linear function. The size of the box and the penalty are updated dynamically so as to make greatest use of the latest available information. With intent to reduce the dual variables' variation, select a small box containing the (in the beginning estimated) current dual solution, and solve the modified master problem. Componentwise, if the new dual solution lies in the box, reduce its width and increase the penalty. Otherwise, enlarge the box and decrease the penalty. This allows for fresh dual solutions when the estimate was bad. The update could be performed in each iteration, or alternatively, each time a dual solution of currently best quality is obtained.

3.3 Dual Aspects of our Shortest Path Example

Optimal Primal Solutions. Assume that we penalize the violation of resource constraint (1.5) via the objective function with the single multiplier $\pi_1 \leq 0$ which we determine using a subgradient method. Its optimal value is $\pi_1 = -2$, as we know from solving the primal master problem by column generation. The aim now is to find an optimal integer solution $\mathbf{x}$ to our original problem. From the Lagrangian subproblem with π_1 we get $L(-2) = 7$ and generate either the infeasible path 1256 of cost 5 and duration 15, or the feasible path 13256 of cost 15 and duration 10. The important issue now, left out in textbooks, is how to perform branch-and-bound in that context?

Assume that we generated path 1256. A possible strategy to start the branch-and-bound search tree is to introduce cut $x_{12} + x_{25} + x_{56} \leq 2$ in the original formulation (1.1)–(1.6), and then either incorporate it in X or relax it (and penalize its violation) in the objective function via a second multiplier. The first alternative prevents the generation of path 1256 for any value of π_1. However, we need to re-compute its optimal value according to the modified X structure, i.e., $\pi_1^\star = -1.5$. In this small example, a simple way to get this value is to solve the linear relaxation of the full master problem excluding the discarded path. Solving the new subproblem results in an improved lower bound $L(-1.5) = 9$, and the generated path 13256 of cost 15 and duration 10. This path is *feasible* but suboptimal. In fact, this solution $\mathbf{x}$ is integer, satisfies the path constraints but does not satisfy complementary slackness for the resource constraint. That is, $\pi_1(\sum_{(i,j)\in A} t_{ij}x_{ij} - 14) = -1.5(10 - 14) \neq 0$. The second cut $x_{13} + x_{32} + x_{25} + x_{56} \leq 3$ in the X structure results in $\pi_1^\star = -2$, an improved lower bound of $L(-2) = 11$, and the generated path 1246 of cost 3 and duration 18. This path is *infeasible*, and adding the third cut $x_{12} + x_{24} + x_{46} \leq 2$ in the subproblem X gives us the optimal solution, that is, $\pi_1 = 0$, $L(0) = 13$ with the generated path 13245 of cost 13 and duration 13.

Alternatively, we could have penalized the cuts via the objective function which would *not* have destroyed the subproblem structure. We encourage the reader to find the optimal solution this way, making use of any kind of branching and cutting decisions that can be defined on the $\mathbf{x}$ variables.

A Box Method. It is worthwhile to point out that a good deal of the operations research literature is about Lagrangian relaxation. We can steal ideas there about how to decompose problems and use them in column generation algorithms (see Guignard, 2004). In fact, the complementary availability of both, primal and dual ideas, brings us in a strong position which e.g., motivates the following. Given optimal multipliers $\boldsymbol{\pi}$ obtained by a subgradient algorithm, one can use very small boxes around these in order to rapidly derive an optimal primal solution $\mathbf{x}$ on which branching and cutting decisions are applied. The dual information is incorporated in the primal RMP in the form of initial columns together with the columns corresponding to the generated subgradients. This gives us the opportunity to initiate column generation with a solution which intuitively bears both, relevant primal and dual information.

Alternatively, we have applied a box method to solve the primal master problem by column generation, c.f. Table 1.5. We impose the box constraint $-2.1 \leq \pi_1 \leq -1.9$. At start, the RMP contains the artificial variable y_0 in the convexity constraint, and surplus (s_1 with cost coefficient 1.9) and slack (s_2 with cost coefficient 2.1) variables in the resource constraint.

Table 1.5. A box method in the root node with $-2.1 \leq \pi_1 \leq -1.9$

Iteration	Master Solution	$\bar{z}$	π_0	π_1	$\bar{c}^\star$	p	c_p	t_p	UB	LB
BoxBB0.1	$y_0 = 1, s_2 = 14$	73.4	100.0	-1.9	-66.5	1256	5	15	–	6.9
BoxBB0.2	$\lambda_{1256} = 1, s_1 = 1$	7.1	36.5	-2.1	-0.5	13256	15	10	7.1	6.6
BoxBB0.3	$\lambda_{13256} = 0.2, \lambda_{1256} = 0.8$	7.0	35.0	-2.0	0				7	7
Arc flows: $x_{12} = 0.8, x_{13} = x_{32} = 0.2, x_{25} = x_{56} = 1$										

In the first iteration, denoted BoxBB0.1, the artificial variable $y_0 = 1$ and the slack variable $s_2 = 14$ constitute a solution. The current dual multipliers are $\pi_0 = 100$ and $\pi_1 = -1.9$. Path 1256 is generated (cost 5 and duration 15) and the lower bound already reaches 6.9. In the second iteration, $\lambda_{1256} = 1$ and surplus $s_1 = 1$ define an optimal solution to the RMP. This solution provides an upper bound of 7.1 and dual multipliers are $\pi_0 = 36.5$ and $\pi_1 = -2.1$. Experimentation reveals that a smaller box around $\pi = -2$ results in a smaller optimality gap. The subproblem generates path 13256 (cost 15 and duration 10) and the lower bound decreases to 6.6. Solving the RMP in BoxBB0.3 gives us an optimal solution of the linear relaxation of the master problem. This can be verified in two ways: the previous lower bound values 6.9 and 6.6, rounded up, equal the actual upper bound $\bar{z} = 7$; and the reduced cost of the subproblem is zero. Hence, the solution process is completed in only three iterations! The box constraint has to be relaxed when branch-and-bound starts but this does not require any re-optimization iteration.

Geometric Interpretation. Let us draw the Lagrangian function $L(\pi)$, for $\pi \leq 0$, for our numerical example, where $\pi \equiv \pi_1$. Since the polytope X is bounded, there are no extreme rays and $L(\pi)$ can be written in terms of the nine possible extreme points (paths). That is, $L(\pi) = \min_{p \in P} c_p + (14 - t_p)\pi$, where c_p and t_p are the cost and duration of path p, respectively. Table 1.6 lists the lines (in general, hyperplanes) defined by $p \in P$, with an intercept of c_p and a slope of $14 - t_p$. We have plotted these lines in Figure 1.4.

Table 1.6. Hyperplanes (*lines*) defined by the extreme points of X, i.e., by the indicated paths

p	1246	1256	12456	13246	13256	132456	1346	13456	1356
line	$3-4\pi$	$5-\pi$	14	$13+\pi$	$15+4\pi$	$24+5\pi$	$16-3\pi$	$27+\pi$	$24+6\pi$

Observe that for π given, the line of smallest cost defines the value of function $L(\pi)$. The Lagrangian function $L(\pi)$ is therefore the lower envelope of all lines and its topmost point corresponds to the value $\mathcal{L}$ of the Lagrangian dual problem.

If one starts at $\pi = 0$, the largest possible value is $L(0) = 3$, on the line defined by path 1246. At that point the slope is negative (the line is defined by $3-4\pi$) so that the next multiplier should be found on the left to the current point. In Dantzig-Wolfe decomposition, we found $\pi(\equiv \pi_1) = -97/18 \approx -5.4$. This result depends on big M: the exact value of the multiplier is $(3 - M)/18$. For any large M, path 1356 is returned, and here, $L(-97/18) = -25/3$, a lesser lower bound on $L(\pi)$.

The next multiplier is located where the two previous lines intersect, that is, where $3-4\pi = 24+6\pi$ for $\pi = -2.1$. $L(-2.1) = 6.6$ for path 13256 with an improvement on the lower bound. In the next iteration, the optimal multiplier value is at the intersection of the lines defined by paths 1246 and 13256, that is, $3 - 4\pi = 15 + 4\pi$ for $\pi = -1.5$. For that value, the Lagrangian function reaches 6.5 for path 1256. The final and optimal Lagrangian multiplier is at the

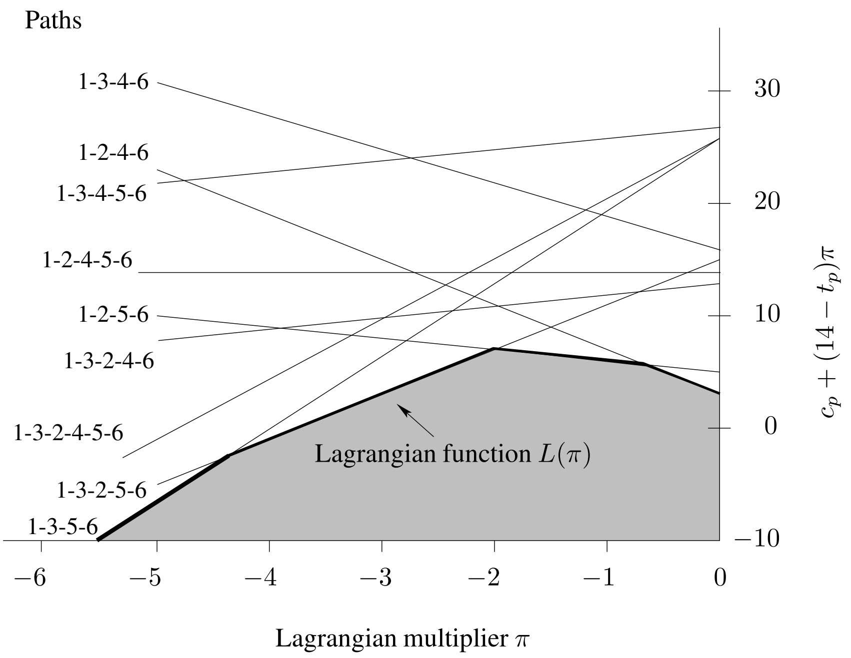

Figure 1.4. Lagrangian function $L(\pi)$

intersection of the lines defined by paths 13256 and 1256, that is, $15+4\pi = 5-\pi$ for $\pi = -2$ and therefore $L(-2) = 7$. We can now see why the lower bound is not strictly increasing: the point associated with the Lagrangian multiplier moves from left to right, and the value of the Lagrangian function is determined by the lowest line which is hit.

The Lagrangian point of view also teaches us why two of our methods are so successful: when we used the *box method* for solving the linear relaxation of the master problem by requiring that π has to lie in the small interval $[-2.1, -1.9]$ around the optimal value $\pi = -2$, only two paths are sufficient to describe the lower envelope of the Lagrangian function $L(\pi)$. This explains the very fast convergence of this stabilization approach.

Also, we previously added the cut $\mathbf{c}^t\mathbf{x} \geq 7$ in the subproblem, when we were looking for an integer solution to our resource constrained shortest path problem. In Figure 1.4 this corresponds to removing the two lines with an intercept smaller than 7, that is, for paths 1246 and 1256. The maximum value of function $L(\pi)$ is now attained for $\pi = 0$ and $L(0) = 13$.

4. On Finding a Good Formulation

Many vehicle routing and crew scheduling problems, but also many others, possess a multicommodity flow problem as an underlying basic structure (see Desaulniers et al., 1998). Interestingly, Ford and Fulkerson, 1958, suggested to solve this "problem of some importance in applications" via a "specialized computing scheme that takes advantage of the structure": the birth of column generation which then inspired Dantzig and Wolfe, 1960 to generalize the framework to a decomposition scheme for linear programs as presented in Section 1.2.1. Ford and Fulkerson had no idea "whether the method is practicable." In fact, at that time, it was not. Not only because of the lack of powerful computers but mainly because (only) linear programming was used to attack integer programs: "That integers should result as the solution of the example is, of course, fortuitous" (Gilmore and Gomory, 1961).

In this section we stress the importance (and the efforts) to find a "good" formulation which is amenable to column generation. Our example is *the* classical column generation application, see Ben Amor and Valério de Carvalho, 2005. Given a set of rolls of width L and integer demands n_i for items of length $\ell_i, i \in I$ the aim of the *cutting stock* problem is to find patterns to cut the rolls to fulfill the demand while minimizing the number of used rolls. An item may appear more than once in a cutting pattern and a cutting pattern may be used more than once in a feasible solution.

4.1 Gilmore and Gomory (1961, 1963)

Let R be the set of all feasible cutting patterns. Let coefficient a_{ir} denote how often item $i \in I$ is used in pattern $r \in R$. Feasibility of r is expressed by the knapsack constraint $\sum_{i\in I} a_{ir}\ell_i \leq L$. The classical formulation of Gilmore and Gomory (1961, 1963) makes use of non-negative integer variables: λ_r reflects how often pattern r is cut in the solution. We are first interested in solving the linear relaxation of that formulation by column generation. Consider the following primal and dual linear master problems P_{CS} and D_{CS}, respectively:

$$
\begin{array}{rll|rll}
(P_{CS}): & \min \sum_{r\in R} \lambda_r & & (D_{CS}): & \max \sum_{i\in I} n_i\pi_i & \\
\sum_{r\in R} a_{ir}\lambda_r & \geq n_i, & i \in I & \sum_{i\in I} a_{ir}\pi_i & \leq 1, & r \in R \\
\lambda_r & \geq 0, & r \in R & \pi_i & \geq 0, & i \in I\,.
\end{array}
$$

For $i \in I$, let π_i denote the associated dual multiplier, and let x_i count the frequency item i is selected in a pattern. Negative reduced cost patterns are generated by solving

$$
\min 1 - \sum_{i\in I} \pi_i\, x_i \equiv \max \sum_{i\in I} \pi_i x_i \text{ such that } \sum_{i\in I} x_i\ell_i \leq L,\ x_i \in \mathbb{Z}_+,\ i \in I.
$$

This pricing subproblem is a knapsack problem and the coefficients of the generated columns are given by the value of variables $x_i,\ i \in I$.

Gilmore and Gomory, 1961 showed that equality in the demand constraints can be replaced by greater than or equal. Column generation is accelerated by this transformation: dual variables $\pi_i, i \in I$ then assume only non-negative values and it is easily shown by contradiction that these *dual non-negativity constraints* are satisfied by all optimal solutions. Therefore they define a set of (simple) dual-optimal inequalities.

Although P_{CS} is known to provide a strong lower bound on the optimal number of rolls, its solution can be fractional and one has to resort to branch-and-bound. In the literature one finds several tailored branching strategies based on decisions made on the $\boldsymbol{\lambda}$ variables, see Barnhart et al., 1998; Vanderbeck and Wolsey, 1996. However, we have seen that branching rules with a potential for exploiting more structural information can be devised when some compact formulation is available.

4.2 Kantorovich (1939, 1960)

From a technical point of view, the proposal by Gilmore and Gomory is a master problem and a pricing subproblem. For precisely this situation, Villeneuve et al., 2003 show that an equivalent compact formulation exists under the assumption that the sum of the variables of the master problem be bounded by an integer κ and that we have the possibility to also bound the domain of the subproblem. The variables and the domain of the subproblem are duplicated κ times, and the resulting formulation has a block diagonal structure with κ identical subproblems. Formally, when we start from Gilmore and Gomory's formulation, this yields the following formulation of the cutting stock problem.

Given the dual multipliers $\pi_i, i \in I$, the pricing subproblem can alternatively be written as

$$\begin{array}{lrcll} \min & (x_0 - \sum_{i \in I} \pi_i\, x_i) & & & \\ \text{subject to} & \sum_{i \in I} \ell_i\, x_i & \leq & L\, x_0 & \\ & x_0 & \in & \{0,1\} & \\ & x_i & \in & \mathbb{Z}_+ & i \in I\ , \end{array} \tag{1.40}$$

where x_0 is a binary variable assuming value 1 if a roll is used and 0 otherwise. When x_0 is set to 1, (1.40) is equivalent to solving a knapsack problem while if $x_0 = 0$, then $x_i = 0$ for all $i \in I$ and this null solution corresponds to an empty pattern, i.e., a roll that is not cut.

The constructive procedure to recover a compact formulation leads to the definition of a specific subproblem for each roll. Let $K := \{1, \ldots, \kappa\}$ be a set of rolls of width L such that $\sum_{r \in R} \lambda_r \leq \kappa$ for some feasible solution $\boldsymbol{\lambda}$. Let

$\mathbf{x}^k = (x_0^k, (x_i^k)_{i\in I}), k \in K$, be duplicates of the $\mathbf{x}$ variables, that is, x_0^k is a binary variable assuming value 1 if roll k is used and 0 otherwise, and $x_i^k, i \in I$ is a non-negative integer variable counting how often item i is cut from roll k. The compact formulation reads as follows:

$$\begin{array}{rrcll}
\min & \sum_{k\in K} x_0^k & & & \\
\text{subject to} & \sum_{k\in K} x_i^k & \geq & n_i & i \in I \\
 & \sum_{i\in I} \ell_i \, x_i^k & \leq & L\, x_0^k & k \in K \\
 & x_0^k & \in & \{0,1\} & k \in K \\
 & x_i^k & \in & \mathbb{Z}_+ & k \in K,\ i \in I,
\end{array} \tag{1.41}$$

which was proposed already by Kantorovich in 1939 (a translation of the Russian original report is Kantorovich, 1960). This formulation is known for the weakness of its linear relaxation. The value of the objective function is equal to $\sum_{i\in I} \ell_i/L$. Nevertheless, a Dantzig-Wolfe decomposition with (1.40) as an integer program pricing subproblem (in fact, κ identical subproblems, which allows for further simplification), yields an extensive formulation the linear programming relaxation of which is equivalent to that of P_{CS}. However, the variables of the compact formulation (1.41) are in a sense interchangeable, since the paper rolls are indistinguishable. One speaks of *problem symmetry* which may entail considerable difficulties in branch-and-bound because of many similar and thus redundant subtrees in the search.

4.3 Valério de Carvalho (2002)

Fortunately, the existence of a compact formulation in the "reversed" Dantzig-Wolfe decomposition process by Villeneuve et al., 2003 does not mean uniqueness. There may exist alternative compact formulations that give rise to the same linear relaxation of an extensive formulation, and we exploit this freedom of choice.

Valério de Carvalho, 2002 suggests a very clever original network-based formulation for the cutting stock problem. Define the acyclic network $G = (N, A)$ where $N = \{0, 1, \ldots, L\}$ is the set of nodes and the set of arcs is given by $A = \{(u, v) \in N \times N \mid v - u = \ell_i, \forall i \in I\} \cup \{(u, v) \mid u \in N \backslash \{L\}\}$, see also Ben Amor and Valério de Carvalho, 2005. Arcs link every pair of consecutive nodes from 0 to L without covering any item. An item $i \in I$ is represented several times in the network by arcs of length $v - u = \ell_i$. A path from the source 0 to the sink L encodes a feasible cutting pattern.

The proposed formulation of the cutting stock problem, which is pseudo-polynomial in size, reads as

$$\min \quad z \tag{1.42}$$

$$\text{subject to} \quad \sum_{(u,u+\ell_i)\in A} x_{u,u+\ell_i} \geq n_i \quad i \in I \tag{1.43}$$

$$\sum_{(0,v)\in A} x_{0,v} = z \tag{1.44}$$

$$\sum_{(u,v)\in A} x_{uv} - \sum_{(v,u)\in A} x_{vu} = 0 \quad v \in \{1,\ldots,L-1\} \tag{1.45}$$

$$\sum_{(u,L)\in A} x_{uL} = z \tag{1.46}$$

$$x_{uv} \in \mathbb{Z}_+ \qquad (u,v) \in A\,. \tag{1.47}$$

Keeping constraints (1.43) in the master problem, the subproblem is $X = \{(\mathbf{x}, z) \text{ satisfying (1.44)–(1.47)}\}$. This set X represents flow conservation constraints with an unknown supply of z from the source and a matching demand at the sink. Given dual multipliers $\pi_i, i \in I$ associated to the constraint (1.43), the subproblem is

$$\min_{(1.44)-(1.47)} \quad z - \sum_{(u,u+\ell_i)\in A} \pi_i \, x_{u,u+\ell_i}\,. \tag{1.48}$$

Observe now that the solution $(\mathbf{x}, z) = (\mathbf{0}, 0)$ is the unique extreme point of X and that all other paths from the source to the sink are extreme rays. Such an extreme ray $r \in R$ is represented by a 0-1 flow which indicates whether an arc is used or not. An application of Dantzig-Wolfe decomposition to this formulation directly results in formulation P_{CS}, the linear relaxation of the extensive reformulation (this explains our choice of writing down P_{CS} in terms of extreme rays instead of extreme points). Formally, as the null vector is an extreme point, we should add one variable λ_0 associated to it in the master problem and the convexity constraint with only this variable. However, this empty pattern makes no difference in the optimal solution as its cost is 0.

The pricing subproblem (1.48) is a shortest path problem defined on a network of pseudo-polynomial size, the solution of which is also that of a knapsack problem. Still this subproblem suffers from some symmetries since the same cutting pattern can be generated using various paths. Note that this subproblem possesses the integrality property although the previously presented one (1.40) does not. Both subproblems construct the same columns and P_{CS} provides the same lower bound on the value of an optimal integer solution. The point is that the integrality property of a pricing subproblem, or the absence of this property, has to be evaluated relative to its own compact formulation. In the present case, the linear relaxation of Kantorovich's formulation provides a lower bound that is weaker than that of P_{CS}, although it can be improved by solving the integer knapsack pricing subproblem (1.40). On the other hand, the linear relaxation of Valério de Carvalho's formulation already provides the same lower bound as P_{CS}. Using this original formulation, one can design branching and cutting decisions on the arc flow variables of network G to get an optimal integer solution.

Let us mention that there are many important applications which have a natural formulation as set covering or set partitioning problems, without any decomposition. In such models it is usually the master problem itself which has to be solved in integers (Barnhart et al., 1998). Even though there is no explicit original formulation used, customized branching rules can often be interpreted as branching on variables of such a formulation.

5. Further Reading

Even though column generation originates from linear programming, its strengths unfold in solving integer programming problems. The simultaneous use of two concurrent formulations, compact and extensive, allows for a better understanding of the problem at hand and stimulates our inventiveness in what concerns for example branching rules.

We have said only little about implementation issues, but there would be plenty to discuss. Every ingredient of the process deserves its own attention, see e.g., Desaulniers et al., 2001, who collect a wealth of acceleration ideas and share their experience. Clearly, an implementation benefits from customization to a particular application. Still, it is our impression that an off-the-shelf column generation software to solve large scale integer programs is in reach reasonably soon; the necessary building blocks are already available. A crucial part is to automatically detect how to "best" decompose a given original formulation, see Vanderbeck, 2005. This means in particular exploiting the absence of the subproblem's integrality property, if applicable, since this may reduce the integrality gap without negative consequences for the linear master program. Let us also remark that instead of a convexification of the subproblem's domain X (when bounded), one can explicitly represent all integer points in X via a *discretization* approach formulated by Vanderbeck, 2000. The decomposition then leads to a master problem which itself has to be solved in integer variables.

In what regards new and important technical developments, in addition to the stabilization of dual variables already mentioned, one can find a dynamic row aggregation technique for set partitioning master problems in Elhallaoui et al., 2003. This allows for a considerable reduction in size of the restricted master problem in each iteration. An interesting approach is also proposed by Valério de Carvalho, 1999 where variables and rows of the original formulation are dynamically generated from the solutions of the subproblem. This technique exploits the fact that the subproblem possesses the integrality property. For a presentation of this idea in the context of a multicommodity network flow problem we refer to Mamer and McBride, 2000.

This primer is based on our recent survey (Lübbecke and Desrosiers, 2002), and a much more detailed presentation and over one hundred references can be found there. For those interested in the many column generation applications in practice, the survey articles in this book will serve the reader as entry points to the large body of literature. Last, but not least, we recommend the book by Lasdon, 1970, also in its recent second edition, as an indispensable source for alternative methods of decomposition.

Acknowledgments. We would like to thank Steffen Rebennack for cross-checking our numerical example, and Marcus Poggi de Aragão, Eduardo Uchoa, Geir Hasle, and José Manuel Valério de Carvalho, and the two referees for giving us useful feedback on an earlier draft of this chapter. This research was supported in part by an NSERC grant for the first author.

References

Ahuja, R.K., Magnanti, T.L., and Orlin, J.B. (1993). *Network Flows: Theory, Algorithms and Applications*. Prentice-Hall, Inc., Englewood Cliffs, New Jersey 07632.

Barnhart, C., Johnson, E.L., Nemhauser, G.L., Savelsbergh, M.W.P., and Vance, P.H. (1998). Branch-and-price: Column generation for solving huge integer programs. *Oper. Res.*, 46(3):316–329.

Ben Amor, H. and Desrosiers, J. (2003) A Proximal Trust Region Algorithm for Column Generation Stabilization. Les Cahiers du GERAD G-2003-43 (2003). To appear in *Comp. & OR*.

Ben Amor, H., Desrosiers, J., and Valério de Carvalho, J.M. (2003). Dual-optimal inequalities for stabilized column generation. Les Cahiers du GERAD G-2003-20, HEC Montréal. Under revision for *Oper. Res.*

Ben Amor, H. and Valério de Carvalho, J.M. (2005). Cutting stock problems. In Desaulniers, G., Desrosiers, J., and Solomon, M.M., editors, *Column Generation*. Kluwer Academic Publishers, Boston, MA.

Dantzig, G.B. and Wolfe, P. (1960). Decomposition principle for linear programs. *Oper. Res.*, 8:101–111.

Desaulniers, G., Desrosiers, J., Ioachim, I., Solomon, M.M., Soumis, F., and Villeneuve, D. (1998). A unified framework for deterministic time constrained vehicle routing and crew scheduling problems. In Crainic, T.G. and Laporte, G., editors, *Fleet Management and Logistics*, pages 57–93. Kluwer, Norwell, MA.

Desaulniers, G., Desrosiers, J., and Solomon, M.M. (2001). Accelerating strategies in column generation methods for vehicle routing and crew scheduling problems. In Ribeiro, C.C. and Hansen, P., editors, *Essays and Surveys in Metaheuristics*, pages 309–324, Boston. Kluwer.

du Merle, O., Villeneuve, D., Desrosiers, J., and Hansen, P. (1999). Stabilized column generation. *Discrete Math.*, 194:229–237.

Elhallaoui, I., Villeneuve, D., Soumis, F., and Desaulniers, G. (2003). Dynamic aggregation of set partitioning constraints in column generation. Les Cahiers du GERAD G-2003-45, HEC Montréal. Under revision for *Oper. Res.*

Ford, L.R. and Fulkerson, D.R. (1958). A suggested computation for maximal multicommodity network flows. *Management Sci.*, 5:97–101.

Geoffrion, A.M. (1974). Lagrangean relaxation for integer programming. *Math. Programming Stud.*, 2:82–114.

Gilmore, P.C. and Gomory, R.E. (1961). A linear programming approach to the cutting-stock problem. *Oper. Res.*, 9:849–859.

Gilmore, P.C. and Gomory, R.E. (1963). A linear programming approach to the cutting stock problem—Part II. *Oper. Res.*, 11:863–888.

Goffin, J.-L. and Vial, J.-Ph. (1999). Convex nondifferentiable optimization: A survey focussed on the analytic center cutting plane method. Technical Report 99.02, Logilab, Université de Genève. To appear in *Optim. Methods Softw.*

Guignard, M. (2004). Lagrangean relaxation. In Resende, M. and Pardalos, P., editors, *Handbook of Applied Optimization*. Oxford University Press.

Hiriart-Urruty, J.-B. and Lemaréchal, C. (1993). *Convex analysis and minimization algorithms, part 2: Advanced theory and bundle methods*, volume 306 of *Grundlehren der mathematischen Wissenschaften*. Springer, Berlin.

Kantorovich, L. (1960). Mathematical methods of organising and planning production (translated from a report in russian, dated 1939). *Management Science*, 6:366–422.

Kelley Jr., J.E. (1961). The cutting-plane method for solving convex programs. *J. Soc. Ind. Appl. Math.*, 8(4):703–712.

Lasdon, L.S. (1970). *Optimization Theory for Large Systems*. Macmillan, London.

Lübbecke, M.E. and Desrosiers, J. (2002). Selected topics in column generation. Les Cahiers du GERAD G-2002-64, HEC Montréal. Under revision for *Oper. Res.*

Nemhauser, G.L. and Wolsey, L.A. (1988), *Integer and Combinatorial Optimization.* John Wiley & Sons, Chichester.

Mamer, J.W. and McBride, R.D. (2000). A decomposition-based pricing procedure for large-scale linear programs – An application to the linear multicommodity flow problem. *Management Sci.*, 46(5):693–709.

Marsten, R.E. (1975). The use of the boxstep method in discrete optimization. *Math. Programming Stud.*, 3:127–144.

Marsten, R.E., Hogan, W.W., and Blankenship, J.W. (1975). The BOXSTEP method for large-scale optimization. *Oper. Res.*, 23:389–405.

Poggi de Aragão, M. and Uchoa, E. (2003). Integer program reformulation for robust branch-and-cut-and-price algorithms. In *Proceedings of the Conference Mathematical Program in Rio: A Conference in Honour of Nelson Maculan*, pages 56–61.

Schrijver, A. (1986). *Theory of Linear and Integer Programming*. John Wiley & Sons, Chichester.

Valério de Carvalho, J.M. (1999). Exact solution of bin-packing problems using column generaion and branch-and-bound. *Annals of Operations Research*, 86:629–659.

Valério de Carvalho, J.M. (2002). LP models for bin-packing and cutting stock problems. *European J. Oper. Res.*, 141(2):253–273.

Valério de Carvalho, J.M. (2003). Using extra dual cuts to accelerate convergence in column generation. To appear in *INFORMS J. Comput.*

Vanderbeck, F. (2000). On Dantzig-Wolfe decomposition in integer programming and ways to perform branching in a branch-and-price algorithm. *Oper. Res.*, 48(1):111–128.

Vanderbeck, F. (2005). Implementing mixed integer column generation. In Desaulniers, G., Desrosiers, J., and Solomon, M.M., editors, *Column Generation*. Kluwer Academic Publishers, Boston, MA.

Vanderbeck, F. and Wolsey, L.A. (1996). An exact algorithm for IP column generation. *Oper. Res. Lett.*, 19:151–159.

Villeneuve, D., Desrosiers, J., Lübbecke, M.E., and Soumis, F. (2003). On compact formulations for integer programs solved by column generation. Les Cahiers du GERAD G-2003-06, HEC Montréal. To appear in *Ann. Oper. Res.*

Walker, W.E. (1969). A method for obtaining the optimal dual solution to a linear program using the Dantzig-Wolfe decomposition. *Oper. Res.*, 17:368–370.