RESEARCH REPORT 0017

SOLVING THE LINEAR PROGRAMMING RELAXATION OF CUTTING AND PACKING PROBLEMS: A HYBRID SIMPLEX METHOD/SUBGRADIENT OPTIMIZATION PROCEDURE

by
Z. DEGRAEVE
M. PEETERS

D/2000/2376/17

Solving the Linear Programming Relaxation of Cutting and Packing Problems : A Hybrid Simplex Method/Subgradient Optimization Procedure

Zeger Degraeve
Katholieke Universiteit Leuven, Department of Applied Economics, Leuven, Belgium
London Business School, Regent's Park, London NW1 4SA, United Kingdom

Marc Peeters
Katholieke Universiteit Leuven, Department of Applied Economics, Leuven, Belgium

Abstract
In this paper we present a new method for solving the linear programming relaxation of the Cutting Stock Problem. The method is based on the relationship between column generation and Lagrange relaxation. We have called our method the Hybrid Simplex Method/Subgradient Optimization Procedure. We test our procedure on generated data sets and compare it with the classical column generation approach.

Programming : Linear Programming, Column Generation, Lagrange Relaxation;

Katholieke Universiteit Leuven
Department of Applied Economics
Naamsestraat 69, B-3000 Leuven
zdegraeve@london.edu
marc.peeters@econ.kuleuven.ac.be

May, 2000

Introduction.

The Cutting Stock Problem (CSP), in its simplest form, is as follows. From one raw material type of a given length, b, available in unlimited supply, a set of n orders of finished product types requiring d_i units of length l_i ($i = 1, 2, \dots , n$) has to be cut such that the total number of the raw material used is minimized. In this case, copy j of the raw material type corresponds to a "cutting pattern", that is, a way of partitioning the copy of the raw material type into various units of the different finished product types.

The Bin Packing Problem (BPP) is a special case of the cutting stock problem and thus can be solved likewise. Each item i of a set of n items with size or weight l_i, must be assigned to exactly one bin of capacity b. In terms of the notation defined above this implies that $d_i = 1$. The total number of bins used must be minimized. According to Dyckhoff's typology (1990) the CSP is a 1/V/I/R problem. This means that it is a one-dimensional problem (1), where all items must be assigned to a selection of objects (V), the large objects are identical (I) and there are relatively few different items (R). The BPP is denoted by 1/V/I/M. The difference with the CSP is that there are many different items (M).

As recently as 1996, state-of-the-art heuristics were published for the cutting stock problem (Wäscher and Gau 1996). During that time, various authors (Degraeve 1992, Vance 1998, Degraeve and Schrage 1999 and Vanderbeck 1999) were developing an exact solution approach by branch and price. This technique was demonstrated to be efficient for solving binary cutting stock problems (Vance et al. 1994) and the generalized assignment problem (Savelsbergh 1997).

In this paper we will focus on solving the linear programming (LP) relaxation. We will present a Hybrid Simplex Method/Subgradient Optimization Procedure (HP). The outline of this paper is as follows. In a first section we will give two formulations of the CSP. The second section presents the HP. In the third section we present the data sets that we will use to test our procedure. In the fourth section we give computational results. Finally we conclude in section five.

1. Formulations

One possible formulation of the Cutting Stock Problem is the generalized assignment formulation. We use the following notation :

n	: number of finished product types (demand lengths), index i,
m	: some upper bound on the minimum number of copies of the raw material needed,
l_i	: length of finished product type i,
d_i	: demand, i.e., number of units required, of finished product type i,
b	: length of the raw material type (raw width),
y_j	= 1, if copy j of the raw material is used, 0, otherwise,
x_{ij}	= the number of times finished product type i is cut from the raw material copy j.

The generalized assignment formulation for CSP is then as follows :

(i) minimize the total number of raw materials used ;

$$Min\sum_{j=1}^{m} y_j \qquad (1.1)$$

subject to

(ii) satisfy demand ;

$$\sum_{j=1}^{m} x_{ij} \geq d_i \qquad i = 1,\ldots,n \qquad (1.2)$$

(iii) cut at most the raw material width ;

$$\sum_{i=1}^{n} l_i x_{ij} \leq b y_j \qquad j = 1,\ldots,m \qquad (1.3)$$

(iv) integrality ;

$$x_{ij} \in \{0,1,2,\ldots\} \qquad i = 1,\ldots,n,\ j = 1,\ldots,m \qquad (1.4)$$

$$y_j \in \{0,1\} \qquad j = 1,\ldots,m$$

Gilmore and Gomory (1961) introduced a different formulation for the CSP. Let P be the index set of all feasible cutting patterns $(a_{1j}, a_{2j}, \ldots, a_{nj})$, where a_{ij} represents the number of times item i is cut out of pattern j i.e. $P = \left\{ j \middle| (a_{1j},\ldots,a_{nj}) : \sum_{i=1}^{n} l_i a_{ij} \leq b \right\}$, and let z_j be number of times to use pattern j in the solution, then we can state their formulation as follows :

(i) minimize total number of patterns used ;

$$Min \quad \sum_{j \in P} z_j \tag{1.5}$$

subject to

(ii) satisfy demand for each finished product type ;

$$\sum_{j \in P} a_{ij} z_j \geq d_i \qquad i = 1,2,\dots,n \tag{1.6}$$

(iii) integrality ;

$$z_j \in \{0,1,2,\dots\} \qquad j = 1,2,\dots,p \tag{1.7}$$

Enumerating all feasible cutting combinations is practically impossible when n becomes high. Therefore column generation is used to solve this problem. Let π_i be the dual price of demand constraint i. If a subset of all columns P is presented in a restricted master problem, then, a new column that can improve the current value of the restricted master $\bar{v}_{RMLP}$, can be found by the solving the following general integer knapsack problem:

$$Max \quad \sum_{i=1}^{n} \pi_i a_{ij} \tag{1.8}$$

subject to

$$\sum_{i=1}^{n} l_i a_{ij} \leq b \tag{1.9}$$

$$a_{ij} \in \{0,1,2,\dots\} \qquad i = 1,2,\dots,n \tag{1.10}$$

If $\sum_{i=1}^{n} \pi_i a_{ij}^* > 1$, the new column $(a_{1j}^*, a_{2j}^*, \dots, a_{nj}^*)$, which is the optimal solution vector to (1.8)-(1.10), prices out attractively, i.e. its reduced cost is negative, it is added to the master and column generation continues, otherwise, column generation can stop and we have an LP optimum to the master, denoted as $\bar{v}_{RMLP}^*$. Traditionally, the restricted master is resolved each time after adding a new column to obtain new dual prices. However, solving the restricted master problem becomes very time consuming particularly when the number of items is large. In this paper we present a different method to compute the LP relaxation of (1.5) - (1.7), called the Hybrid Simplex Method/Subgradient Optimization Procedure (HP).

2. Hybrid simplex method/Subgradient optimization procedure

In this section, we outline an efficient method to find the LP relaxation of the master program using the Lagrange dual. Instead of uniquely relying on the simplex method to compute the dual prices, π_i, to generate new columns, we have implemented a combination of the simplex method and subgradient optimization. This HP essentially consists of a nested double loop. In the outer loop, we compute first the dual prices using the simplex method, then, in the inner loop, subgradient optimization is used to update the dual prices for a specific number of iterations. Each time new dual prices are found they are used to price out a new column.

We use the following additional notation:

π	= vector of the dual prices of the demand constraints,
$w(\pi)$	= objective value of the Lagrange relaxation for a given vector of dual prices π,
w_{LD}	= objective value of the Lagrange dual,
l	: index of the outer loop indicating the number of times the dual prices are computed with the simplex method in the hybrid simplex method/subgradient optimization procedure,
k	: index of the inner loop indicating the number of iterations of the subgradient optimization in the hybrid simplex method/subgradient optimization procedure,
$\overline{w}_{LD}^{l}$	= approximation to the Lagrange dual at iteration l,
$\pi^{l,k}$	= vector of the dual prices at the iteration l, k.

The Lagrange relaxation for the CSP and BPP obtained by dualizing the demand constraints (1.3) of generalized assignment formulation (1.1)-(1.4) with dual prices π_i, is then as follows :

$$w(\pi) = Min \sum_{j=1}^{m} \left(y_j - \sum_{i=1}^{n} \pi_i x_{ij} \right) + \sum_{i=1}^{n} \pi_i d_i \qquad (2.1)$$

subject to

$$\sum_{i=1}^{n} l_i x_{ij} \le b y_j \qquad j = 1,\ldots,m \qquad (2.2)$$

$$x_{ij} \in \{0,1,2,\ldots\} \qquad i = 1,\ldots,n,\ j = 1,\ldots,m \qquad (2.3)$$

$$y_j \in \{0,1\} \qquad j = 1,\ldots,m \tag{2.4}$$

Solving this problem is equivalent to solving m times the same knapsack problem, so we can drop the index j and rewrite the above problem as follows :

$$w(\pi) = m * Min\left(y - \sum_{i=1}^{n} \pi_i x_i \right) + \sum_{i=1}^{n} \pi_i d_i \tag{2.5}$$

subject to

$$\sum_{i=1}^{n} l_i x_i \le by \tag{2.6}$$

$$x_i \in \{0,1,2,\ldots\} \qquad i = 1,\ldots,n \tag{2.7}$$

$$y \in \{0,1\} \tag{2.8}$$

The solution of problem (2.5) - (2.8) is as follows. If the solution of the knapsack problem :

$$Max\left\{ \sum_{i=1}^{n} \pi_i x_i : \sum_{i=1}^{n} l_i x_i \le b,\, x_i \in \{0,1,2,\ldots\}, i = 1,\ldots,n \right\} \tag{2.9}$$

is greater than one, then y is one, else y and x_i, $i = 1,\ldots,n$, are zero. To find the Lagrange dual w_{LD}, we must maximize (2.5) – (2.8) over the dual prices, or :

$$w_{LD} = \underset{\pi}{Max}\{w(\pi) : \pi \ge 0\} \tag{2.10}$$

It is well known that the optimal value of the Lagrange dual is equal to the optimal LP relaxation value of formulation (1.5)-(1.7), i.e. $w_{LD} = \overline{v}^*_{RMLP}$. Indeed LP based column generation can be considered as a way to find the Lagrange dual (Wolsey 1998 or Martin 1999). We will exploit this fact in our new procedure for finding the LP lower bound on the Gilmore and Gomory formulation (1.5)-(1.7).

At the outset, the master problem (1.5)-(1.7) is initialized with n initial columns. Each initial column contains, for each finished product type, its maximum number given demand and knapsack constraint, i.e. $a_{ii} = \min\left(d_i, \left\lfloor b/l_i \right\rfloor \right)$, i = 1, 2, ..., n, and this is complemented by other finished product types to create a non-dominated pattern. We solve the restricted master with the simplex method and collect the dual prices. These dual prices are used as the initial values to start the subgradient optimization procedure. In terms of the notation introduced above, they constitute the vector $\pi^{1,0}$.

We now use subgradient optimization to find the Lagrange dual. Let t_k denote the stepsize. The updating formula for the dual price of demand constraint i at iteration k is then as follows :

$$\pi_i^{l,k+1} = \pi_i^{l,k} + t_k(d_i - mx_i) \quad i = 1,\ldots,n \tag{2.11}$$

$$t_k = \frac{(m - w(\pi^{l,k}))}{\sum_{i=1}^{n}(d_i - mx_i)^2} \tag{2.12}$$

We set the upper bound m equal to the objective value of the master problem solved with the simplex method at the l^{th} iteration rounded up, $m = \lceil \overline{v}_{RMLP} \rceil$.

As a result, the dual prices $\pi^{l,k}$ that we obtain with (2.11) are in fact an approximation to the dual prices of the master problem at the LP optimum. As we have to solve (2.5) – (2.8) at each iteration k of the subgradient optimization procedure, a feasible cutting pattern j is generated $(a_{1j}, a_{2j}, \ldots, a_{nj})$, where $a_{ij} = x_i^*$ for i=1,2,…,n and $(x_1^*, x_2^*, \ldots, x_n^*)$ is the optimal solution vector of the knapsack problem (2.9). We add these columns to the master program if they price out given the dual prices of the last solved master, i.e.

$$1 - \sum_{i=1}^{n} \pi_i^{l,0} a_{ij} < 0 \tag{2.13}$$

and if they are different from the columns added during the current subgradient optimization step. After a few iterations or if the reduced cost of the last found cutting pattern is non negative, the master with the new columns is solved again with the simplex method. This results in a new upper bound on the optimal LP relaxation value, $m = \lceil \overline{v}_{RMLP} \rceil$ and in new dual prices, $\pi^{l+1,0}$. These are used again as input for the subgradient optimization process.

Observe that in the first iteration of each subgradient optimization phase we solve a knapsack subproblem that is completely identical to the subproblem we would have solved in the LP based column generation approach. If the reduced cost of this column is negative, we can be sure that the column is different from all the others currently in the master. If the reduced cost of the column is non-negative, the LP optimum is found. If the approximation to the Lagrange dual at iteration l rounded up

$\lceil \overline{w}_{LD}^{l} \rceil$ is equal to the value of the master LP rounded up $\lceil \overline{v}_{RMLP} \rceil$, we can stop generating columns as well.

An overview of the procedure is depicted in Figure 1 (Page 16). In the following text, a number between square brackets refers to the number of the corresponding box in the Figure. First, we initialize the master problem (1.5)-(1.7) with the initial columns as described above [0]. We solve it with the simplex method and increase the index *l*, which indicates the number of times the master is solved [1]. Then in [2] we check whether the approximation of the Lagrange dual rounded up $\lceil \overline{w}_{LD}^{l} \rceil$ is smaller then the current master LP value rounded up $\lceil \overline{v}_{RMLP} \rceil$. If not, we have found the LP lower bound [12], else we collect the dual prices of the demand constraints in the master $\pi^{l,0}$, solve subproblem (2.9) and set the number of subgradient iterations *k* to zero [3]. We compute the reduced cost $1-\sum_{i=1}^{n}\pi_i^{l,0}\, x_i^*$ [4]. In case the reduced cost is non negative the LP lower bound is found [12], if it is negative, we compute the new Lagrange bound $w(\pi^{l,k})$ in [5]. In [6], we check again whether the new Lagrange bound rounded up is smaller then the current master LP value rounded up. If it is not, we can stop. Else, we check if we do not exceed a preset limit on the number of subgradient iterations in [7]. If not, we perform a new iteration of the subgradient optimization procedure in [8] to obtain new dual prices $\pi^{l,k+1}$ using the update formula's (2.11)-(2.12) and increase the number of iterations *k* by one. If the limit is exceeded, we add the new columns generated in the subgradient optimization process to the master [11] and resolve the master [1]. In [9] we solve the subproblem (2.9) again and we compute the reduced cost in [10]. If the reduced cost of the cutting proposal found in [9] is negative, we compute the Lagrange bound again [5]. Else, we add the new columns [11] and solve the master again [1].

The main advantage of our hybrid simplex method/subgradient optimization procedure is that we do not need to solve the master problem as an LP each time to get new dual prices necessary for pricing out a new column. Solving the master problem is computationally much more expensive than performing an iteration of the subgradient optimization procedure. At each subgradient iteration, we get a new column and these columns are "good" because the dual prices found with subgradient optimization converge towards the dual prices of the master at the LP optimum of the problem. Indeed, at the master's LP optimum, they are approximately equal. Therefore, we can expect that columns needed to find the optimal LP solution will be generated during the subgradient optimization phase. This will be confirmed by our computational results, which show that only a very few times we need to solve the master with the simplex method. A second advantage is that we can stop the column generation process short of proving LP optimality of the master. The solution of a restricted master LP is an upper bound on the optimal LP relaxation value, while the Lagrange bound is a lower on this value. Once the Lagrange bound rounded up and the solution of the master LP rounded up are equal, we have found a lower bound to CS. As such, we avoid the "tailing-off" effect typical for traditional column generation. This effect implies that a large percentage of the total number of columns generated is needed to prove LP optimality (Vanderbeck and Wolsey 1996). The main disadvantage of subgradient optimization is that it does not provide a primal solution. Consequently, at the end, we have to solve the master as an LP with the simplex method to obtain variable value. A second disadvantage is that the dual prices are only approximately equal to the optimal dual prices of the master LP. As a consequence the lower bound can be somewhat less tight.

The hybrid simplex method/subgradient optimization procedure combines the speed of subgradient optimization with the exactness of the simplex method. This procedure can be extremely effective for other problems solved with column generation and branch and price as well, e.g. for solving the Generalized Assignment Problem (Sȧvelsberg 1997).

3 Description of the Data

For our computational tests, we use 560 randomly generated data sets for the cutting stock problem. The generated data were derived using a portable implementation (Degraeve and Schrage 1997) of a problem generator by Gau and Wäscher (1995). The number of finished product types n determines the size of an instance of the problem, it has been fixed to n = 10, 20, 30, 40, 50, 75 and 100. The raw material length has been set to b = 10000. We have modeled the demand lengths l_i as uniform distributed integer random variables between 1 and a fraction c of b, i.e.

$1 \le l_i \le c*b$. The values of the fraction c used are c = 0.25, 0.50, 0.75 and 1. The demand of finished product type i, d_i, is a uniformly distributed integer random variables derived from a constant total demand T with T = 50*n. Combining the different values for n, c and T results in 28 (= 7 * 4 * 3) classes of problem instances. We have generated 20 problem instances in each class so that a total of 560 data sets were used for our computations.

We test our algorithm on Bin Packing instances as well. We use the classical bin packing data set of Martello and Toth (1990). They use three different bin capacities b of 100, 120 and 150 and three different intervals from which the item length or weight l_i is randomly drawn from a uniform distribution, namely (1, 100), (20, 100) and (50, 100). The number of item generated in these intervals will be 500 and 1000. We will generate 20 problem instances for each class, using an adapted version of the portable problem generator (Degraeve and Schrage 1997). We consider a total of 360 (= 2 * 3 * 3 * 20) bin packing data sets. In case identical item lengths are generated during the construction of the bin packing instances, only the item length is considered but the demand is incremented. We are in fact changing a BPP in an equivalent CSP instance.

Finally, we also use the "triplet" instances from the OR-library (Beasley 1990). Those instances are so constructed that the optimum is always equal to $n/3$ and in addition $\sum_{i=1}^{n} l_i = n/3 * b$ with n indicating the number of items, l_i the length or weight of item i and b the bin capacity. The length of each item is carefully chosen from the interval (250, 499), the bin capacity b = 1000. Four classes of 20 problem instances were constructed with different number of items n = 60, 120, 249, 501 which we will denote by BPL5, BPL6, BPL7 and BPL8 respectively.

We have programmed our procedure in Fortran 77 using the WATCOM Fortran compiler version 10.6 and linked with the industrial LINDO optimization library version 5.3 (Schrage 1995). The experiments were run on a Dell Pentium Pro 200Mhz PC (Dell Dimension XPS Pro 200n) using the Windows95 operating system, all computation times are given in seconds.

4 Computational Results

In the first table (Table 1.A), you find the computational results for the cutting stock instances. In the columns labeled 'TM', you find the CPU seconds to solve the LP relaxation, using the traditional method, while in the columns labeled 'HP' you find the results for our new procedure. The procedure TM initializes the restricted master with the same columns as HP. HP is consistently better than TM, especially as the number of different item types becomes large. Table 1.B gives the number of restricted master problems that must be solved by procedure 'TM' and 'HP'. As expected, the number of masters for the traditional method is a lot higher than for the hybrid simplex method/subgradient optimization procedure.

Table 1.A : Comparison HP and TM, CSP, cpu-seconds.

	(1,2500)		(1,5000)		(1,7500)		(1,10000)	
n	TM	HP	TM	HP	TM	HP	TM	HP
10	0.02	0.01	0.01	0.01	0.00	0.00	0.01	0.00
20	0.09	0.04	0.12	0.07	0.02	0.03	0.01	0.02
30	0.18	0.09	0.38	0.19	0.08	0.06	0.03	0.02
40	0.03	0.15	0.87	0.35	0.26	0.20	0.07	0.06
50	0.44	0.21	1.47	0.52	0.67	0.46	0.14	0.10
75	1.14	0.47	4.82	1.12	4.26	1.14	0.53	0.27
100	3.19	0.84	15.96	2.05	14.78	3.99	1.65	0.73

Table 1.B: Comparison HP and TM, CSP, number of masters.

	(1,2500)		(1,5000)		(1,7500)		(1,10000)	
n	TM	HP	TM	HP	TM	HP	TM	HP
10	17.7	3.3	10.8	3.1	4.2	2.2	2.7	2.0
20	29.0	3.3	39.1	4.8	15.6	3.3	5.6	2.3
30	39.4	3.6	66.9	4.6	19.3	3.0	8.9	2.1
40	50.1	4.0	100.5	4.8	37.6	4.7	16.1	2.6
50	58.1	4.1	116.8	6.1	52.2	4.6	21.5	2.8
75	81.5	3.5	210.6	5.6	103.2	4.9	35.2	2.9
100	113.6	4.8	230.2	7.7	159.5	6.5	56.1	3.5

Table 2 presents the results for the Bin Packing instances. We remark that before solving the BPP instances, we call a preprocessing routine, which eliminates some items, using a dominance rule of Martello and Toth (1990). For more details we refer to Degraeve and Peeters (1998). We can conclude that our new method is always faster. For the three weight intervals, the average reduction is respectively 83%, 78%

and 67%. Table 2.B shows that the number of restricted master problems that must be solved is considerably lower for HP than for the traditional method.

Table 2.A: Comparison HP and TM, BPP, cpu seconds.

		(1,100)		(20,100)		(50,100)	
n	*b*	TM	HP	TM	HP	TM	HP
	100	0.19	0.04	0.09	0.05	0.03	0.01
500	120	0.70	0.08	0.09	0.01	0.04	0.02
	150	0.40	0.14	0.24	0.02	0.05	0.01
	100	0.20	0.08	0.07	0.06	0.03	0.01
1000	120	1.49	0.18	0.08	0.01	0.03	0.02
	150	1.32	0.22	0.31	0.04	0.05	0.01

Table 2.B: Comparison HP and TM, BPP, number of masters.

		(1,100)		(20,100)		(50,100)	
n	*b*	TM	HP	TM	HP	TM	HP
	100	22.7	2.4	21.3	2.9	1.0	1.0
500	120	44.0	3.0	23.6	2.1	4.4	1.9
	150	30.3	2.2	59.0	6.1	18.5	2.0
	100	24.3	2.9	16.7	2.9	1.0	1.0
1000	120	62.0	3.5	22.5	2.3	2.9	1.8
	150	56.0	4.3	72.9	4.3	17.5	2.0

Finally, Table 3 shows the results for the triplet instances. The column labeled 'CPU' presents the CPU seconds needed to find the lower bound, while the columns labeled 'MST' presents the number of master problems that must be solved. Again, The HP is consistently better. For the most difficult problems, BPL8, HP reduces the CPU time to less then than 1/3.

Table 3: Triplets

	CPU		MST	
	TM	HP	TM	HP
BPL5	1.71	1.01	201.6	8.8
BPL6	8.15	5.77	467.2	24.0
BPL7	42.49	19.68	829.2	46.9
BPL8	140.57	32.57	1556.2	161.7

5 Conclusions and Ideas for Future Research

We have described and tested an improved procedure to solve the LP relaxation of the CSP and BPP, called the Hybrid Simplex Method/Subgradient Optimization Method. In our future work we will integrate this method in branch and price algorithms to solve the one-dimensional cutting stock and related problems.

References

BEASLEY, J.E. 1990. OR-library : Distributing Test Problems by Electronic Mail. *Journal of the Operational Research Society* **41**, 1069-1072.

DEGRAEVE, Z. 1992. Scheduling Joint Product Operations with Proposal Generation Methods, Ph.D. Dissertation, The University of Chicago, Graduate School of Business, Chicago, IL.

DEGRAEVE, Z. AND PEETERS, M. 1998. Benchmark Results for the Cutting Stock and Bin Packing Problem. Onderzoeksrapport nr 9820, K.U.Leuven, Departement of Applied Economics.

DEGRAEVE, Z. AND L. SCHRAGE. 1999. Optimal Integer Solutions to Industrial Cutting Stock Problems. *INFORMS Journal on Computing* **11**, 406-419.

DEGRAEVE, Z. AND L. SCHRAGE. 1997. Should We Use a Portable Generator in an Emergency? Research Report, Department of Applied Economic Sciences, Katholieke Universiteit Leuven, Belgium.

DYCKHOFF, H. 1990. A Typology of Cutting and Packing Problems. *European Journal of Operational Research* **44**, 145-159.

GAU T. AND G. WÄSCHER. 1995. CUTGEN1 : A Problem Generator for the Standard One-dimensional Cutting Stock Problem. *European Journal of Operational Research* **84**, 572-579.

GILMORE, P.C. AND R.E. GOMORY. 1961. A Linear Programming Approach to the Cutting Stock Problem. *Operations Research* **9**, 849-859.

MARTELLO, S. AND P. TOTH. 1990. *Knapsack Problems : Algorithms and Computer Implementations.* Wiley Interscience Series in Discrete Mathematics and Optimization, John Wiley & Sons, Chichester.

MARTIN, R.K. 1999. *Large Scale Linear and Integer Optimization.* Kluwer Academic Publishers, Boston.

SAVELSBERGH, M. 1997. A Branch-and-Price Algorithm for the Generalized Assignment Problem. *Operations Research* **45**, 831-841.

SCHRAGE, L. 1995. LINDO : Optimization Software for Linear Programming. Lindo Systems Inc., Chicago, IL.

VANCE, P.H., BARNHART, C., JOHNSON, E.L. AND G.L. NEMHAUSER. 1994. Solving Binary Cutting Stock Problems by Column Generation and Branch and Bound. *Computational Optimization and Applications*, **3**, 111-130.

VANCE, P.H. 1998. Branch-and-Price Algorithms for the One-Dimensional Cutting Stock Problem. *Computational Optimization and Applications*. **9**(3), 212-228.

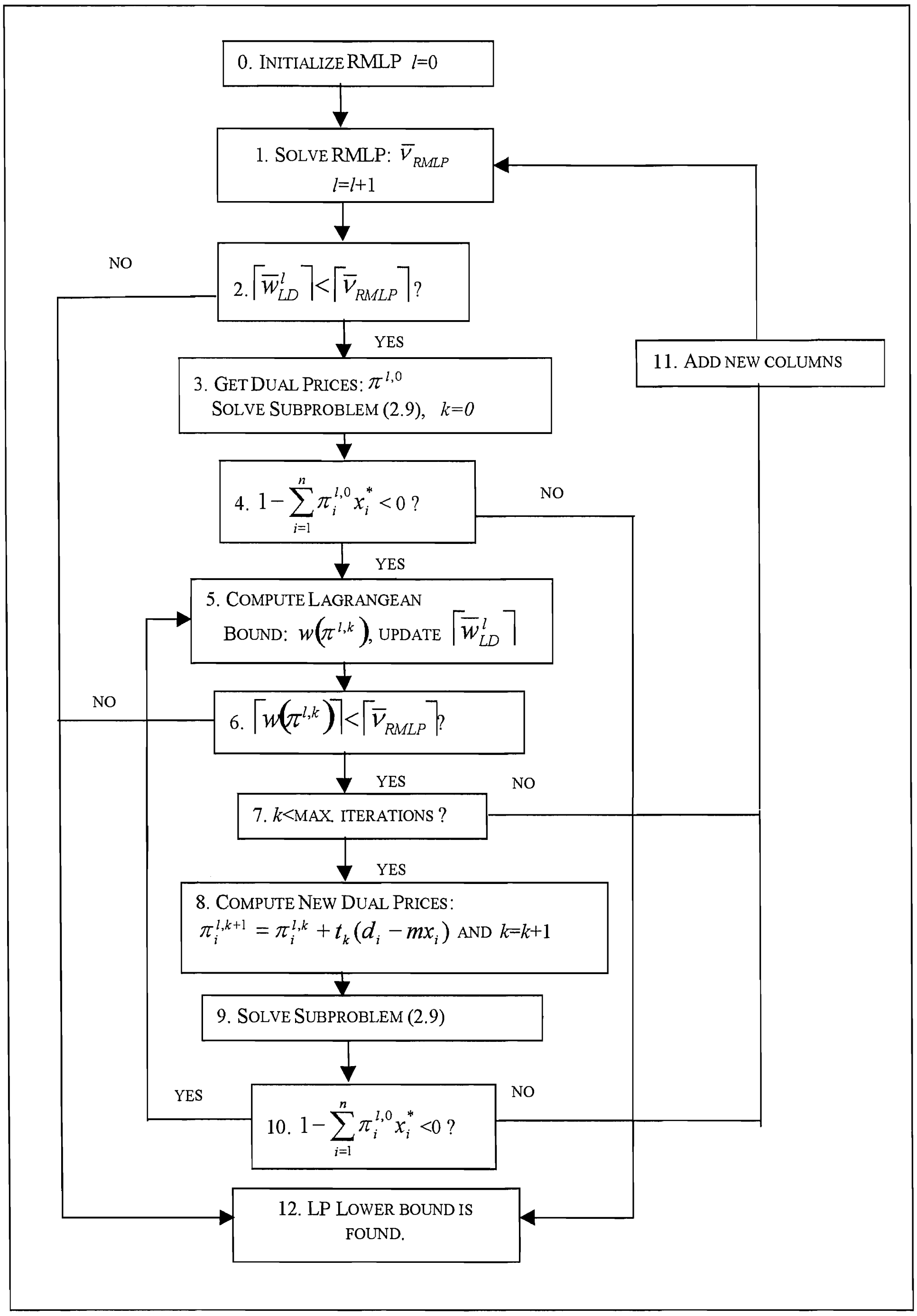
VANDERBECK, F. 1999. Computational Study of a Column Generation algorithm for Bin Packing and Cutting Stock problems, *Mathematical Programming* **86**, Ser. A, 565-594.

VANDERBECK, F. AND L.A. WOLSEY 1996. An exact algorithm for IP column generation. *OR letters* **19**,151-159.

WÄSCHER, G. AND T. GAU. 1996. Heuristics for the Integer One-dimensional Cutting stock Problem : A Computational Study. *OR Spektrum* **18**, 131-144.

WOLSEY, L.A. 1998. *Integer Programming*. Wiley Interscience Series in Discrete Mathematics and Optimization, John Wiley & Sons, Chichester.

Figure 1: Flow Chart: Hybrid simplex method/subgradient optimization Procedure.



0. INITIALIZE RMLP l=0
1. SOLVE RMLP: $\bar{v}_{RMLP}$ $l=l+1$
2. $\lceil \bar{w}_{LD}^{l} \rceil < \lceil \bar{v}_{RMLP} \rceil$?
NO
YES
3. GET DUAL PRICES: $\pi^{l,0}$ SOLVE SUBPROBLEM (2.9), $k=0$
4. $1-\sum_{i=1}^{n} \pi_i^{l,0} x_i^* < 0$?
NO
YES
5. COMPUTE LAGRANGEAN BOUND: $w(\pi^{l,k})$, UPDATE $\lceil \bar{w}_{LD}^{l} \rceil$
6. $\lceil w(\pi^{l,k}) \rceil < \lceil \bar{v}_{RMLP} \rceil$?
NO
YES
NO
7. k<MAX. ITERATIONS ?
YES
8. COMPUTE NEW DUAL PRICES: $\pi_i^{l,k+1} = \pi_i^{l,k} + t_k(d_i - mx_i)$ AND $k=k+1$
9. SOLVE SUBPROBLEM (2.9)
YES
NO
10. $1-\sum_{i=1}^{n} \pi_i^{l,0} x_i^* <0$?
11. ADD NEW COLUMNS
12. LP LOWER BOUND IS FOUND.