RAIRO. RECHERCHE OPÉRATIONNELLE

Y. CRAMA
J. VAN DE KLUNDERT

Approximation algorithms for integer covering problems via greedy column generation

RAIRO. Recherche opérationnelle, tome 28, n° 3 (1994), p. 283-302

<http://www.numdam.org/item?id=RO_1994__28_3_283_0>

Recherche opérationnelle/Operations Research
(vol. 28, n° 3, 1994, p. 283 à 302)

APPROXIMATION ALGORITHMS FOR INTEGER COVERING PROBLEMS VIA GREEDY COLUMN GENERATION (*)

by Y. CRAMA ([1, 2]) and J. VAN DE KLUNDERT ([2])

Communicated by Catherine ROUCAIROL

Abstract. – *Many combinatorial problems can be formulated as covering problems. In some cases, e.g. the cutting stock problem, this formulation is not polynomial in the input size of the original problem. In order to solve the problem approximately one can apply the Greedy algorithm to the covering formulation. In this case, the column generation subproblem is to determine which column the Greedy algorithm chooses. This problem might be NP-hard in itself. We propose a modification of the Greedy algorithm in which the column generation subproblem is solved approximately, within a factor α. We derive upper bounds for the worst case ratio of this algorithm and related polynomial approximation algorithms.*

Keywords: integer programming; covering problems; greedy heuristics; approximation algorithms; column generation.

Résumé. – *Beaucoup de problèmes d'optimisation combinatoire peuvent être formulés comme des problèmes de recouvrement. Dans certains cas, par exemple pour le problème de découpe de stocks, cette formulation est de dimension exponentielle en la dimension du problème de départ. Pour résoudre le problème de façon approximative, on peut appliquer une heuristique gourmande au modèle de recouvrement. Dans ce cadre, le problème du choix de la meilleure colonne, à chaque itération de l'heuristique gourmande, peut être vu comme un problème de génération de colonne. Ce sous-problème est parfois lui-même NP-difficile. On propose ici une modification de l'heuristique gourmande, dans laquelle le sous-problème de génération de colonne est résolu par une heuristique polynomiale à performance garantie. On analyse de façon précise la performance de l'algorithme résultant. Une analyse similaire est réalisée pour différentes variantes de cette approche.*

Mots clés : programmation entière ; problèmes de recouvrement ; heuristiques gourmandes ; algorithmes approximatifs ; génération de colonnes.

1. INTRODUCTION

Many combinatorial optimization problems can be formulated as covering problems of the form:

(*) Received September 1993.

([1]) Département de Gestion, Université de Liège, B-4000 Liège, Belgium.
([2]) Department of Quantitative Economics, Faculty of Economics, University of Limburg, Maastricht, The Netherlands.

Recherche opérationnelle/Operations Research, 0399-0559/94/03/$ 4.00

$$\begin{array}{ll} \text{minimize} & cx \\ \text{s.t.} & Ax \geqq b \\ & x \in \mathbf{Z}_+^n \end{array} \qquad (P)$$

where $c \in \mathbb{R}_+^n$, $A \in \mathbf{Z}_+^{m \times n}$ and $b \in \mathbf{Z}_+^m$. (Notice that all data are nonnegative.)

In some cases, this covering formulation is not polynomial in the input size of the original problem. Typically, for instance, m may be a parameter of the original problem, whereas n is exponential in m. This can be illustrated by the covering formulation of the cutting stock problem due to Gilmore and Gomory [5] (*see* also Chvátal [2]; for a general discussion of cutting stock problems see Dyckhoff [3], Haessler and Sweeney [7]). In this problem there are rolls of material of various length, called raws, from which pieces of specified length, called finals, have to be cut. The demand for each final is given. The cost of cutting finals from a raw may depend on the size of the raw, and/or on the cutting pattern. A covering formulation can be obtained as follows. Each row corresponds to some final. Each column corresponds to a feasible cutting pattern, *i.e.* to some way of cutting certain finals from some specific raw. The cost of pattern j is denoted c_j, the number of finals of type i produced by pattern j is denoted a_{ij}, and the demand for final i is denoted b_i. It is not hard to see that the number of columns might be exponential in the problem size.

Gilmore and Gomory [5] proposed a column generation technique to solve the linear relaxation of such large-scale models. In their approach, the columns of (P) are not explicitly listed. Rather, the pricing step of the simplex algorithm is implemented as follows: given current values $u_1, \ldots, u_m$ of the dual variables, the column generation subproblem $\max \{ \Sigma_i u_i a_{ij} - c_j \mid j = 1, \ldots, n \}$ is solved in order to detect columns with positive reduced cost. For the method to be applicable, the *column generation problem* should be (relatively) easy to solve, and should not require a complete enumeration of all columns of (P). For instance, Minoux [14] observed that the approach allows to solve the linear relaxation of (P) in polynomial time whenever the column generation subproblem itself is polynomially solvable. When (P) is a cutting stock problem, the column generation subproblem turns out to be a knapsack problem, which is *NP*-hard, but can be solved reasonably fast in practice (*see* Gilmore and Gomory

[5] or Chvátal [2] for details). Similar approaches have been successfully applied to many other problems.

In this paper, we turn to a different, but related question. Rather than solving the linear relaxation of (*P*) [which only yields a lower-bound on the optimal value of (*P*)], we are interested here in computing a good heuristic solution of (*P*). More precisely, we are going to discuss a class of approximation algorithms for problem (*P*) and to derive bounds on their worst-case ratio (the worst-case ratio of an approximation algorithm is the supremum, taken over all problem instances, of the ratio between the value of the solution delivered by the approximation algorithm and the optimal value of the problem instance; *see* e.g. Nemhauser and Wolsey [15]). As customary when discussing heuristics, we will be especially interested in polynomial-time algorithms.

Our algorithms build on the algorithm Greedy studied by Johnson [9], Lovász [12] (in the case where all entries of b and c are 1), Chvátal [1] (in the case where all entries of b are 1) and Dobson [4] (for general b and c). Dobson [4] showed that Greedy has worst-case ratio $H(A_{\max})$ where $A_{\max} = \max_{1 \leq j \leq n} \sum_{i=1}^{m} a_{ij}$, and $H(d) = \sum_{i=1}^{d} 1/i$. It is well known that $\ln d < H(d) < \ln d + 1$. (Recently Lund and Yannakakis [13] proved that there cannot exist an approximation algorithm for the Set Covering Problem having worst-case ratio $c \times \log m$ with $0 < c < 1/4$, unless $NP = \text{DTIME}(n^{\text{polylog } n})$.) The previous results assume that the formulation (*P*) is explicitly given as input. Our goal will be to circumvent the difficulties posed when this is not the case, *i.e.* when n is very large.

For instance, Greedy requires to select (iteratively) a column j of A which minimizes the ratio $c_j / \sum_i a_{ij}$. Of course, this step poses no particular difficulty if all columns of A are explicitly available. When this is not the case, it is also clear that the column to be selected can be found by solving a generation subproblem similar to the one discussed above. If this subproblem is polynomially solvable, then the complexity of Greedy is only affected by a polynomial factor. However, when the subproblem itself is *NP*-hard, Greedy may no longer be a polynomial-time procedure (for instance, in the framework of Gilmore and Gomory's cutting stock problem, the subproblem is a knapsack problem, as before). In this paper we concentrate on the situation where an approximation algorithm with worst-case ratio α is available to solve the column generation subproblem. (Notice that α may not

be a constant, and that the approximation algorithm need not be polynomial – though this is obviously the most attractive case.) In other words, we assume that we can find (in polynomial time) a column k of (P) such that

$$c_k / \sum_i a_{ik} \leqq \alpha \times \min_{1 \leqq j \leqq n} \{ c_j / \sum_i a_{ij} \}$$

where $\alpha \geqq 1$ (this extends to problem (P) the Master-Slave approach described by Simon [18] for "classical" set covering problems). Under this assumption, we prove in Section 2 that the worst-case ratio of a modified version of Greedy (called αGreedy) is α times the worst-case ratio of Greedy (Theorem 1). In the special case where all entries of b and c are 1, Simon [17] established the worst-case ratio $\alpha \ln m$ for αGreedy. Theorem 1 refines this result and extends it to general values of b and c. As a matter of fact, under Simon's assumptions, we can even obtain a slightly sharper result, which we state as Theorem 2.

When applied to (P), αGreedy may require the selection of $O(n+m)$ columns. Therefore, the complexity of αGreedy is $O((n+m)T)$, where T is (up to some algebraic manipulations on A, b and c) the running time of the approximation algorithm used to solve the column generation subproblem. Notice that this running time is not satisfactory when n is large (*i.e.*, exponential in m), *even* if we assume that T is polynomial in m. To remedy this difficulty, we propose in Section 3 a modification of αGreedy that reduces its complexity to $O(mT)$ while degrading its worst-case ratio by a factor of at most 2 (*see* Theorem 3). In particular, the resulting HyperαGreedy algorithm runs in polynomial time if T is polynomial in m. Some illustrations are presented in Section 4.

2. αGREEDY ALGORITHMS AND THEIR WORST-CASE RATIO

In this section we first describe the greedy algorithm for problem (P) (Dobson [4]). Thereafter we present the αGreedy algorithm and we derive its worst-case ratio. We give a worst-case example, showing that the bound is tight. Finally, we derive a sharper result for the case where all entries of b and c are 1.

The idea behind the greedy algorithm is, in each iteration, to increase the variable corresponding to the relatively cheapest column. Hence, the greedy

algorithm picks a column k such that

$$k = \arg \min_j \left(c_j \Big/ \sum_{i=1}^{m} a_{ij} \right),$$

and increases x_k by 1 unit. In order for the greedy algorithm not to be misled by extremely high values of the constraint coefficients, we assume as in Dobson [4] that $a_{ij} \leqq b_i$ for all i in $\{1, \ldots, m\}$, j in $\{1, \ldots, n\}$ (otherwise the data should be updated accordingly.) We now formulate the greedy algorithm in a slightly different version from the one in Dobson [4].

Here and in the remainder of this paper, we implicitly assume that any row or column which becomes zero during the execution of the algorithm is left out of consideration in all subsequent iterations.

ALGORITHM 1: Greedy

x:=0
z:=0
while $b \neq 0$ *do*
begin
  $k := \arg \min_{1 \leqq j \leqq n} \left(c_j \Big/ \sum_{i=1}^{m} a_{ij} \right)$
  while for all i, $b_i \geqq a_{ik}$ *do*
  begin
    $x_k := x_k + 1$
    $b_i := b_i - a_{ik}$ *for all* i
    $z := z + c_k$
  end
  $a_{ij} := \min(a_{ij}, b_i)$ *for all* i, j
end.

This presentation of the algorithm facilitates its worst case analysis. However, it is easy to see that the inner-while loop could also be replaced by a statement of the form

$$x_k := x_k + \min_{1 \leqq i \leqq m} \lfloor b_i / a_{ik} \rfloor$$

with corresponding updates of b and z. This would result in an algorithm with $O(n + m)$ iterations.

We now give the description of the αGreedy algorithm. The idea here is that instead of iteratively picking the relatively cheapest column, we pick

some column k such that

$$c_k \Big/ \sum_{i=1}^{m} a_{ik} \leqq \alpha \times \min_{1 \leqq j \leqq n} c_j \Big/ \sum_{i=1}^{m} a_{ij}, \tag{1}$$

where $\alpha \geqq 1$, and α is fixed throughout the algorithm. Formally the αGreedy algorithm is:

ALGORITHM 2: αGreedy
x:=0
z:=0
while $b \neq 0$ *do*
begin
select k such that
$c_k \Big/ \sum_{i=1}^{m} a_{ik} \leqq \alpha \times \min_{1 \leqq j \leqq n} \left(c_j \Big/ \sum_{i=1}^{m} a_{ij} \right)$
while for all i, $b_i \geqq a_{ik}$ *do*
begin
$x_k := x_k + 1$
$b_i := b_i - a_{ik}$ *for all* i
$z := z + c_k$
end
$a_{ij} := \min(a_{ij}, b_i)$ *for all* i, j
end.

Clearly αGreedy can be modified in the same way as Greedy, so as to reduce its number of iterations to $O(n + m)$. In particular, αGreedy can be implemented to run in polynomial time if there is a polynomial algorithm to select a column k satisfying (1).

As mentioned in the introduction, Dobson [4] proved that Greedy has worst-case ratio $H(A_{\max})$. We claim the following worst-case ratio for αGreedy:

THEOREM 1: *If* x^* *is an optimal solution of problem (P) and* x' *is the solution computed by* α*Greedy, then*

$$cx'/cx^* \leqq \alpha \times H(A_{\max})$$

and this bound is tight.

Proof of Theorem 1: Since the proof is analogous to the proof in Dobson [4] for the worst-case ratio of Greedy, we introduce the same notations and we skip most of the parts that go through with little or no modification. We

consider one execution of the body of the inner while statement to be one iteration of αGreedy and we denote by t the total number of iterations. During the execution of the algorithm, A and b change frequently. The notations $A^r = (a_{ij}^r)$ and $b^r = (b_i^r)$ refer to the data at the beginning of iteration r, $r = 1, \ldots, t+1$, with $A^{t+1} = 0$ and $b^{t+1} = 0$, by definition. We also let $w_j^r = \sum_{i=1}^{m} a_{ij}^r$ for all j in $\{1, \ldots, n\}$. Notice that the following implication holds, in view of the description of αGreedy: for $i = 1, \ldots, m$, $j = 1, \ldots, n$ and $r = 1, \ldots, t$

$$\text{if} \quad a_{ij}^r < a_{ij}^1 \qquad \text{then} \quad a_{ij}^r = b_i^r \tag{2}$$

We also use a set of step functions, one for each constraint of (P). Let k_r be the column picked in iteration r of αGreedy. Then we define $p_i(s)$ to be the function with domain $[0, b_i)$ and such that

$$p_i(s) = c_{k_r}/w_{k_r}^r \qquad \text{if} \quad s \in [b_i^{r+1}, b_i^r) \quad \text{for } r = 1, \ldots, t.$$

Finally we introduce a step function p_{ij} for each a_{ij}, where

$$p_{ij}(s) = \begin{cases} c_j/w_j^r & \text{if} \quad s \in [a_{ij}^{r+1}, a_{ij}^r) \quad \text{for } r = 1, \ldots, t \\ p_i(s) & \text{if} \quad s \in [a_{ij}^1, b_i). \end{cases}$$

The proof will be based on the following three lemmas.

LEMMA 1: *If x^l is the solution computed by the αGreedy algorithm, then*

$$cx^l = \sum_{i=1}^{m} \int_0^{b_i} p_i(s)\, ds.$$

Proof: See Dobson [4]; the reasoning goes through without modification.

LEMMA 2: *For all $i = 1, \ldots, m$ and $j = 1, \ldots, n$,*

$$\frac{a_{ij}}{b_i} \int_0^{b_i} p_i(s)\, ds \leqq \alpha \int_0^{a_{ij}} p_{ij}(s)\, ds.$$

Proof: Fix $j \in \{1, \ldots, n\}$ and $i \in \{1, \ldots, m\}$. The inequality holds if $a_{ij} = 0$. So assume $a_{ij} \neq 0$. Define $Min = \min_{0 \leqq s < a_{ij}} p_{ij}(s)$ and

$Max = \max_{a_{ij} \leqq s < b_i} p_{ij}(s)$. Let r be such that

$$Min = \frac{c_j}{w_j^r} \equiv p_{ij}(a_{ij}^{r+1}),$$

where $a_{ij}^{r+1} < a_{ij}$, and let u and $k = k_u$ be such that

$$Max = \frac{c_k}{w_k^u} \equiv p_{ij}(b_i^{u+1}),$$

where $b_i^{u+1} \geqq a_{ij}$.

Notice that from (1), it follows that $a_i^{r+1} = b_i^{r+1}$, and hence $r > u$. We now claim that the following inequalities hold:

$$\alpha \times Min = \alpha \frac{c_j}{w_j^r} \geqq \alpha \frac{c_j}{w_j^u} \geqq \frac{c_k}{w_k^u} = Max. \tag{3}$$

Indeed, by definition,

$$w_j^r = \sum_{i=1}^{m} a_{ij}^r \qquad \text{and} \qquad w_j^u = \sum_{i=1}^{m} a_{ij}^u.$$

Since $u < r$ it must be the case that $a_{ij}^r \leqq a_{ij}^u$ for all $i \in \{1, \ldots, m\}$. This proves the first inequality in (3). The second inequality is immediate by definition of αGreedy. Now Lemma 2 can be proved as follows:

$$\begin{aligned}
\frac{1}{b_i}\int_0^{b_i} p_i(s)\,ds &= \frac{1}{b_i}\left[\int_0^{a_{ij}} p_i(s)\,ds + \int_{a_{ij}}^{b_i} p_i(s)\,ds\right] \\
&\leqq \frac{1}{b_i}\left[\int_0^{a_{ij}} \alpha\, p_{ij}(s)\,ds + \int_{a_{ij}}^{b_i} p_{ij}(s)\,ds\right] \quad \text{(by definition of } \alpha\text{Greedy)} \\
&\leqq \frac{1}{b_i}\left[\int_0^{a_{ij}} \alpha\, p_{ij}(s)\,ds + (b_i - a_{ij})\,Max\right] \\
&\leqq \frac{1}{b_i}\left[\int_0^{a_{ij}} \alpha\, p_{ij}(s)\,ds + (b_i - a_{ij})\,\alpha \times Min\right] \quad \text{(by (3))} \\
&\leqq \frac{\alpha}{b_i}\left[\int_0^{a_{ij}} p_{ij}(s)\,ds + \frac{b_i - a_{ij}}{a_{ij}} \int_0^{a_{ij}} p_{ij}(s)\,ds\right] \\
&= \frac{\alpha}{a_{ij}}\int_0^{a_{ij}} p_{ij}(s)\,ds. \quad \blacksquare
\end{aligned}$$

LEMMA 3: *For all* $j = 1, \ldots, n$,

$$\sum_{i=1}^{m} \frac{a_{ij}}{b_i} \int_0^{b_i} p_i(s)\, ds \leqq \alpha\, c_j\, h_j$$

where

$$h_j = \sum_{r=1}^{v_j} \frac{w_j^r - w_j^{r+1}}{w_j^r} \qquad \textit{and} \qquad v_j = \min\{\, r \mid w_j^{r+1} = 0 \,\}.$$

Proof: One proves, as in Dobson [4], that

$$\sum_{i=1}^{m} \int_0^{a_{ij}} p_{ij}(s)\, ds = c_j\, h_j$$

Together with Lemma 2, this easily implies the statement. ■

Now we are in a position to prove Theorem 1. Let x be any feasible solution, then the following holds for $i = 1, \ldots, m$:

$\sum_{j=1}^{n} \frac{a_{ij}\, x_j}{b_i} \geqq 1$ and hence:

$$\begin{aligned} cx' &= \sum_{i=1}^{m} \int_0^{b_i} p_i(s)\, ds && \text{(by Lemma 1)} \\ &\leqq \sum_{i=1}^{m} \sum_{j=1}^{n} \frac{a_{ij}\, x_j}{b_i} \int_0^{b_i} p_i(s)\, ds \\ &= \sum_{j=1}^{n} \Big(\sum_{i=1}^{m} \frac{a_{ij}}{b_i} \int_0^{b_i} p_i(s)\, ds \Big)\, x_j \\ &\leqq \alpha \sum_{j=1}^{n} (c_j\, h_j)\, x_j && \text{(by Lemma 3)} \\ &\leqq \alpha \max_{1 \leqq j \leqq n} (h_j) \sum_{j=1}^{n} c_j\, x_j. \end{aligned}$$

Hence we have

$$\frac{cx'}{cx} \leqq \alpha \max_{1 \leqq j \leqq n} h_j,$$

where x is any feasible solution. Dobson [4] shows that

$$\max_{1 \leqq j \leqq n} h_j \leqq H(A_{\max}).$$

This inequality yields the desired upper bound.

The following problem instance (adapted from Chvátal [1]) shows that this bound is tight for any $\alpha \geqq 1$, and any (integer) value of $A_{\max}$.

$$
\begin{array}{lccccccccc}
\text{Min} & \frac{\alpha}{d} x_1 & + & \frac{\alpha}{(d-1)} x_2 & + \ldots + & \frac{\alpha}{2} x_{d-1} & + \alpha x_d & + x_{d+1} & & \\
\text{s.t.} & x_1 & & & & & & + x_{d+1} & \geqq & 1 \\
 & & & x_2 & & & & + x_{d+1} & \geqq & 1 \\
 & & & & \ddots & & & \vdots & & \\
 & & & & & x_{d-1} & & + x_{d+1} & \geqq & 1 \\
 & & & & & & x_d & + x_{d+1} & \geqq & 1.
\end{array}
$$

The optimal solution of this instance is given by $x_i^* = 0\,(1 \leqq i \leqq d)$, $x_{d+1} = 1$, and has value 1. But αGreedy could set $x_i = 1$ in iteration $i\,(1 \leqq i \leqq d)$, thus resulting in a solution with value $\alpha\, H\,(d)$. ■

Notice that the worst-case instance is in fact a Weighted Set Covering Problem. Since in the Weighted Set Covering Problem the constraint matrix is a 0-1 matrix, the running time of an αGreedy algorithm is $O\,(m)$ instead of the $O\,(n + m)$ in the case of integral constraints.

When all entries of b and c are 1 (Unweighted Set Covering Problem), we can obtain a sharper result than Theorem 1. (Theorem 2 strengthens a result of Simon [18] - *see* also Johnson [10], Simon [17] - who establishes the asymptotic worst-case ratio $\alpha \ln m$ for αGreedy.)

THEOREM 2: *If x^* is an optimal solution of the Unweighted Set Covering Problem and x' is the solution given by αGreedy then*

$$cx'/cx^* \leqq \alpha\, H\,(\lceil d/\alpha \rceil) + \frac{d}{\lceil d/\alpha \rceil} - \alpha$$

and this bound is tight.

Proof: As in Lovász [12], one proves in case of αGreedy that:

$$
\begin{aligned}
cx'/cx^* &\leqq \sum_{i=1}^{\lceil d/\alpha\rceil-1} \frac{\lfloor i\,\alpha\rfloor}{i\,(i+1)} + \sum_{i=\lceil d/\alpha\rceil}^{d-1} \frac{d}{i\,(i+1)} + 1 \\
&\leqq \sum_{i=1}^{\lceil d/\alpha\rceil-1} \frac{i\,\alpha}{i\,(i+1)} + d \times \sum_{i=\lceil d/\alpha\rceil}^{d-1} \frac{1}{i\,(i+1)} + 1 \\
&= \alpha\, H\,(\lceil d/\alpha\rceil) - \alpha + d \times \sum_{i=\lceil d/\alpha\rceil}^{d-1} \left(\frac{1}{i} - \frac{1}{i+1}\right) + 1 \\
&= \alpha\, H\,(\lceil d/\alpha\rceil) + \frac{d}{\lceil d/\alpha\rceil} - \alpha
\end{aligned}
$$

This bound can only be tight for integral values of α, as is easily checked. Hence we show now how to construct a worst-case instance for arbitrary integral values for α and d. The worst case instance consists of $\lceil d/\alpha\rceil + 1$ groups of columns. The columns of the last group will constitute the optimal solution. The columns of the other groups will together constitute the solution found by αGreedy. For the construction of the columns we refer to the example. We give here the number of columns in each group, showing that the bound is tight.

In total there are $d \times (\lceil d/\alpha\rceil)!$ rows. Every column in the last group covers d rows and there are $(\lceil d/\alpha\rceil)!$ such columns. Every column of group $i\,(i = 1 \ldots \lceil d/\alpha\rceil)$ covers i rows. In the i-th group $(i = 1 \ldots \lceil d/\alpha\rceil - 1)$ there are $\alpha/i \times (\lceil d/\alpha\rceil)!$ columns. In group $\lceil d/\alpha\rceil$ there are $(d - \alpha\,(\lceil d/\alpha\rceil - 1)) \times (\lceil d/\alpha\rceil)!)/\lceil d/\alpha\rceil)$ columns.

Any two columns in the last group are disjoint, as are any two columns not in the last group. Thus, together, the columns in groups $1 \ldots \lceil d/\alpha\rceil$ cover all $d \times (\lceil d/\alpha\rceil)!$ rows.

Together, all columns in group i intersect an arbitrary column of the last group in exactly α rows, for $(i = 1 \ldots \lceil d/\alpha\rceil - 1)$. Similarly, all columns in group $\lceil d/\alpha\rceil$ intersect an arbitrary column of the last group in $(d - \alpha\,(\lceil d/\alpha\rceil - 1))$ rows, *i.e.* at most α rows. It follows that αGreedy may first pick all columns of group $\lceil d/\alpha\rceil$, then all columns of group $\lceil d/\alpha\rceil - 1$, and so on until group 1. Hence, the solution found by αGreedy consists of

$$
\begin{aligned}
&\lceil d/\alpha\rceil ! \times (\alpha\, H\,(\lceil d/\alpha\rceil - 1) + (d - \alpha\,(\lceil d/\alpha\rceil - 1))/\lceil d/\alpha\rceil) \\
&\quad = \lceil d/\alpha\rceil ! \times \left(\alpha\, H\,(\lceil d/\alpha\rceil) + \frac{d}{\lceil d/\alpha\rceil} - \alpha\right)
\end{aligned}
$$

columns, whereas the optimal solution contains $(\lceil d/\alpha \rceil)!$ columns.

An example with $\alpha = 4$, $d = 10$ is shown in Figure 1. The example actually shows only half of the rows and columns. The first group is on the right etc. ■

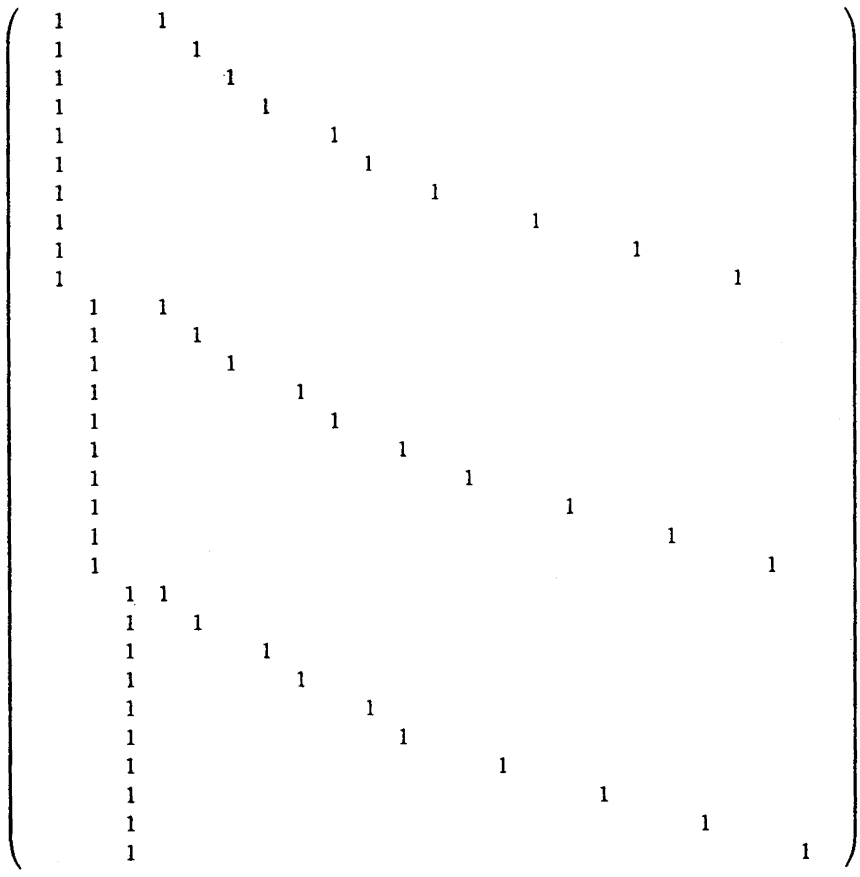

Figure 1

3. HYPERαGREEDY ALGORITHMS AND THEIR WORST-CASE RATIO

For reasons explained in the introduction we are not satisfied with an αGreedy algorithm performing $O(m+n)$ iterations. To get rid of the $O(n)$ term, we now propose a modification of the αGreedy algorithm into a HyperαGreedy algorithm performing $O(m)$ iterations. In this section we describe the HyperαGreedy algorithm, and derive a bound on its worst-case ratio.

ALGORITHM 3: HyperαGreedy

x:=0
z:=0
while $b \neq 0$ *do*
begin
  select k *such that*
  $$c_k \Big/ \sum_{i=1}^{m} a_{ik} \leqq \alpha \times \min_{1 \leqq j \leqq n} \left(c_j \Big/ \sum_{i=1}^{m} a_{ij} \right)$$
  while for all i, $b_i > 0$ *do*
  begin
    $x_k := x_k + 1$
    $b_i := \max(0, b_i - a_{ik})$ *for all* i
    $z := z + c_k$
  end
  $a_{ij} := \min(a_{ij}, b_i)$ *for all* i, j
end.

This presentation parallels the presentation of αGreedy in Section 2. The difference between the two algorithms is that, in αGreedy, x_k stops increasing as soon as $b_i < a_{ik}$ for some i. Under the same conditions, x_k is increased by one more unit in HyperαGreedy. It is easily seen that the inner-while statement can be replaced by a statement of the form

$$x_k := x_k + \min_{1 \leqq i \leqq m} \lceil b_i / a_{ik} \rceil$$

with corresponding updates of b and z. (Contrast this with the statement $x_k := x_k + \min_{1 \leqq i \leqq m} \lfloor b_i / a_{ik} \rfloor$ in the case of αGreedy.) This results in an algorithm performing $O(m)$ iterations, since HyperαGreedy covers then at least one b_i in each iteration of the outer while statement (*i.e.* satisfies at least one constraint).

We claim the following worst-case ratio for HyperαGreedy:

THEOREM 3: *If* x^* *is an optimal solution of problem (P) and* x^l *is the solution computed by Hyper*α*Greedy, then*

$$cx^l / cx^* < 2\,\alpha\, H(A_{\max}).$$

Proof of Theorem 3: First, we introduce some notations. We view one execution of the outer while statement as a step of HyperαGreedy. Let l be the number of such steps, and notice that l is in $\{1, \ldots, m\}$. The data at the beginning of step r are denoted by A^r, b^r. Let k_r, $r \in \{1, \ldots, l\}$, be

the column picked by HyperαGreedy in step r. Let i_r be any row such that

$$i_r = \arg \min_{1 \leqq i \leqq m} \left\lceil \frac{b_i^r}{a_{ik_r}^r} \right\rceil$$

and let

$$h_r = \left\lceil \frac{b_{i_r}^r}{a_{i_r k_r}^r} \right\rceil$$

Thus, h_r represents the increase of x_{k_r} in step r. Let

$$p_r = \left\lfloor \frac{b_{i_r}^r}{a_{i_r k_r}^r} \right\rfloor$$

Notice that $p_r \geqq 1$ for all r in $\{1, \ldots, l\}$. We will use p^r in order to define a problem (P^1) that plays an important role in the proof. We define A_j^r to be the j-th column of A^r and β^r as follows:

$$\beta^r = \sum_{t=r}^{l} p_t A_{k_t}^t \qquad (r = 1, \ldots, l).$$

The idea behind this definition is that β^r is precisely the vector that would have been covered by αGreedy in steps $r, \ldots, l$, had αGreedy followed the same steps as HyperαGreedy. Notice that, in these steps, HyperαGreedy covers b^r. This motivates the following lemma:

LEMMA 4: *For all r in $\{1, \ldots, l\}$ and i in $\{1, \ldots, m\}$, $\beta_i^r \leqq b_i^r$.*

Proof: Fix r and i and assume that row i is covered by HyperαGreedy in step s, *i.e.* $i = i_s$. Observe that for $t > s$, $a_{ij}^t = 0$ for all j in $\{1, \ldots, n\}$. Now consider first the case where $r \leqq s$. Then, since $b_i^r - b_i^s$ is precisely the quantity "covered" in steps $r, r+1, \ldots, s-1$, we have:

$$b_i^r = \sum_{t=r}^{s-1} h_t a_{ik_t}^t + b_i^s = \sum_{t=r}^{s-1} h_t a_{ik_t}^t + \frac{b_i^s}{a_{ik_s}^s} a_{ik_s}^s.$$

Since $h_t \geqq p_t$ and $(b_i^s / a_{ik_s}^s) \geqq p_s$, we obtain for $r \leqq s$,

$$b_i^r \geqq \sum_{t=r}^{s} p_t a_{ik_t}^t = \sum_{t=r}^{l} p_t a_{ik_t}^t = \beta_i^r.$$

On the other hand, when $r > s$, $b_i^r = \beta_i^r = 0$. This proves lemma 4. ■

Define now a new problem (P^1) as follows:

$$\begin{array}{ll} \text{minimize} & cy \\ \text{s.t.} & Ay \geqq \beta^1 \\ & y \in \mathbf{Z}_+^n \end{array} \qquad (P^1)$$

Denote by y^* an optimal solution of (P^1), then,

LEMMA 5: $cy^* \leqq cx^*$.

Proof: Directly from Lemma 4. ■

The next lemma makes more precise the intuitive interpretation of the vector β^1 that we gave earlier.

LEMMA 6: *When applied to* (P^1), *αGreedy may pick the sequence of columns* $k_1, \ldots, k_l$ *and increase* y_{k_r} *by* p_{k_r} *in step r, r in* $\{1, \ldots, l\}$.

Proof: We prove this by induction on r. In fact we prove slightly more. For i in $\{1, \ldots, m\}$, j in $\{1, \ldots, n\}$ and r in $\{1, \ldots, l\}$, let

$$\alpha_{ij}^r = \min(a_{ij}, \beta_i^r)$$

We are going to prove that, when applied to the following problem (P^r):

$$\begin{array}{ll} \text{minimize} & cy \\ \text{s.t.} & \sum_{j=1}^{n} \alpha_{ij}^r y_j \geqq \beta_i^r \qquad i \text{ in } \{1, \ldots, m\} \\ & y_j \in \mathbf{Z}_+^n \end{array} \qquad (P^r)$$

αGreedy may pick column k_r in the first iteration; moreover, this column is equal to $A_{k_r}^r$, *i.e.*

$$\alpha_{ik_r}^r = a_{ik_r}^r \quad \text{for all } i \text{ in } \{1, \ldots, m\}$$

and, updating (P^r) yields (P^{r+1}). Lemma 6 directly follows from these claims.

To prove our claims, notice first that, for all i in $\{1, \ldots, m\}$, j in $\{1, \ldots, n\}$:

$$\alpha_{ij}^r = \min(a_{ij}, \beta_i^r) \leqq \min(a_{ij}, b_i^r) = a_{ik_r}^r, \tag{4}$$

(the inequality follows from Lemma 4). Moreover:

$$\alpha_{ik_r}^r = \min(a_{ik_r}, \beta_i^r) \geqq \min(a_{ik_r}^r, \beta_i^r) = a_{ik_r}^r,$$

since $a_{ik_r} \geqq a_{ik_r}^r$ and $\beta_i^r \geqq a_{ik_r}^r$ by the definition of β^r. Together with (4) this implies $\alpha_{ik_r}^r = a_{ik_r}^r$ for all i in $\{1, \ldots, m\}$ as claimed.

Observing that column k_r of (P^r) is equal to $A_{k_r}^r$, while each other column j of (P^r) is smaller than or equal to A_j^r by (4), it is clear that αGreedy may pick column k_r in the first step (since HyperαGreedy did pick it.) For each i in $\{1, \ldots, m\}$

$$\frac{\beta_i^r}{\alpha_{ik_r}^r} = \sum_{t=r}^{l} \frac{p_t\, a_{ik_t}^t}{a_{ik_r}^r} = p_r + \sum_{t=r+1}^{l} \frac{p_t\, a_{ik_t}^t}{a_{ik_r}^r} \geqq p_r. \tag{5}$$

Moreover, since row i_r was covered by HyperαGreedy applied to (P) in iteration r, $a_{i_r k_t}^t = 0$ for all t in $\{r+1, \ldots, l\}$. Thus

$$\frac{\beta_{i_r}^r}{\alpha_{i_r k_r}^r} = p_r. \tag{6}$$

From (5) and (6) it follows that, when applied to (P^r), αGreedy increases y_{k_r} by p_r in the first step, and decreases the right hand side by $p_r\, A_{k_r}^r$, *i.e.* β^r becomes β^{r+1}. Hence (P^r) is updated into (P^{r+1}). ■

Let $y'_j = \sum_{\substack{r=1 \\ k_r = j}}^{l} p_r$. From the previous proof, it follows that $y' = (y'_1, y'_2, \ldots, y'_n)$ is an αGreedy solution to (P^1). Thus, by Theorem 1 and Lemma 5,

$$cy' \leqq \alpha\, H\left(\max_{1 \leqq j \leqq n} \sum_{i=1}^{m} \alpha_{ij}^1\right) cy^* \qquad \text{(by Theorem 1)}$$

$$\leqq \alpha\, H\left(\max_{1 \leqq j \leqq n} \sum_{i=1}^{m} a_{ij}\right) cx^* \qquad \text{(by Lemma 5)}$$

Notice that the solution x' delivered by HyperαGreedy satisfies

$$x'_j = \sum_{\substack{r=1 \\ k_r=j}}^{l} h_r, \text{ and that } h_r \leqq 2\,p_r \text{ for all } r \text{ in } \{1, \ldots, l\}.$$

Hence $cx' \leqq 2\,cy' \leqq 2\,\alpha\,H(A_{\max})\,cx^*$, as announced. ■

It is not too difficult to see that the bound in Theorem 3 can not hold with equality. Indeed, if $x' = y'$, then the previous proof shows that $cx' \leqq \alpha\,H(A_{\max})\,cx^*$. If $x' \neq y'$, then $b > \beta^1$, and hence $Ax^* > \beta^1$. The last part of the proof of Theorem 1, in Section 2, implies that, under these conditions, $cy' < \alpha\,H(A_{\max})\,cx^*$, and thus $cx' \leqq 2\,cy' < 2\,\alpha\,H(A_{\max})\,cx^*$.

In fact, we conjecture that the bound in Theorem 3 can be tightened as follows:

$$cx'/cx^* \leqq 2\,\alpha \times H(d) \times \frac{A_{\max}}{(A_{\max}+1)}.$$

The following instance shows that this bound would be tight for all values of α and all integer values of $A_{\max}$:

$$\begin{array}{lccccccccc}
\text{Min} & \frac{d}{d\alpha}\,x_1 & + \frac{d}{(d-1)\,\alpha}\,x_2 & + \ldots + & \frac{d}{2\,\alpha}\,x_{d-1} & + \frac{d}{\alpha}\,x_d & + x_{d+1} & & \\
\text{s.t.} & dx_1 & & & & & + x_{d+1} & \geqq & d+1 \\
 & & dx_2 & & & & + x_{d+1} & \geqq & d+1 \\
 & & & \ddots & & & \vdots & & \\
 & & & & dx_{d-1} & & + x_{d+1} & \geqq & d+1 \\
 & & & & & dx_d & + x_{d+1} & \geqq & d+1.
\end{array}$$

For this instance, $A_{\max} = d$. The optimal solution is given by $x_i^* = 0\,(1 \leqq i \leqq d)$, $x_{d+1} = d+1$, and has value $d+1$. But HyperαGreedy could set $x_i = 2$ in iteration $i\,(1 \leqq i \leqq d)$, thus resulting in a solution with value $2\,\alpha\,dH(d)$.

4. APPLICATIONS

We conclude this paper with a brief discussion of two applications of the results presented in the previous sections.

The first application is to the Cutting Stock problem described in Section 1. Consider the rather general variant of this problem in which there is a cost

associated with each raw. Then, the Cutting Stock problem can be modelled in the format (*P*), where the cost c_j associated with the *j-th* cutting pattern is simply the cost of the corresponding raw. When this is the case, Gilmore and Gomory [5] observed that the column generation subproblem reduces to a sequence of knapsack problems, one for each raw (*see* also Chvátal [2]). Now, there is a simple approximation algorithm for (the maximization version of) the knapsack problem with worst-case ratio 1/2. This, together with Theorem 3, yields an approximation algorithm with worst-case ratio $4\,H\,(A_{\max})$ for the Cutting Stock problem. In fact, this ratio can even be made arbitrarily close to $2\,H\,(A_{\max})$ since the knapsack problem admits a fully polynomial time approximation scheme (*see* e.g. Nemhauser and Wolsey [15]).

A second application arises when probabilistic logic is used to model uncertain information (e.g. the information contained in the knowledge base of an expert system). A fundamental problem in this framework is the Probabilistic Satisfiability problem, which can be informally stated as follows (*see* Nilsson [16]): given a set of propositional clauses and the probability that each clause is true, decide whether this probability assignment is consistent. We are now going to show how one variant of this problem can be modelled as a covering problem with a large number of columns (*see* also Kavvadias and Papadimitriou [11], Hansen, Jaumard and Poggi de Aragão [8]). To describe this model, let $\{ C_1, \ldots, C_m \}$ be a set of clauses on the propositional variables $\{ V_1, \ldots, V_n \}$, and let $p = (p_1, \ldots, p_m)$, where p_i is the probability assigned to clause C_i, $i = 1, \ldots, m$. Let $W = \{ w_1, \ldots, w_{2^n} \} = \{ True, False \}^n$ denote the set of possible worlds, *i.e.* the set of possible truth assignments for $\{ V_1, \ldots, V_n \}$. We introduce now a $m \times 2^n$ matrix A, such that $a_{ij} = 1$ if w_j is a satisfying truth assignment for clause C_i, and $a_{ij} = 0$ otherwise. Then, the Probabilistic Satisfiability problem asks whether the following system has a feasible solution:

$$Ax \geqq p, \qquad \sum_{i=1}^{2^n} x_i = 1, \qquad x \geqq 0.$$

Assume now (without loss of generality for practical purposes) that all parameters p_i are rational numbers of the form $p_i = b_i/q$, with b_i and q integer ($i = 1, \ldots, m$). Then it is easy to see that the above system is feasible if and only if the optimal value of the following linear programming

problem is at most q:

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^{2^n} x_i \\ \text{s.t.} & Ax \geqq b \\ & x \in \mathbb{R}_+^{2^n} \end{array} \qquad (PS)$$

Kavvadias and Papadimitriou [11] have observed that problem (*PS*) can be solved by a column generation approach, and that the column generation subproblem turns out in this case to be a weighted maximum satisfiability problem (MAXSAT). Since there exist polynomial time approximation algorithms with worst-case ratio 3/4 for MAXSAT (*see* e.g. Goemans and Williamson [6]), Theorem 3 implies that one can find in polynomial time an integer-valued solution of (*PS*) whose value is at most 8/3 $H(m)$ times the optimal value of (*PS*). If this solution has value at most q, then it solves the Probabilistic Satifiability problem. Otherwise, it could provide a reasonable initial solution (viz, set of columns) in order to solve (*PS*) by column generation.

ACKNOWLEDGEMENTS

The authors gratefully acknowledge helpful suggestions by G. Gallo. The first author was partially supported in the course of this research by AFOSR (grant F49620-93-1-0041) and ONR (grants N00014-92-J1375 and N00014-92-4083).

REFERENCES

1. V. Chvátal, A greedy heuristic for the set-covering problem, *Mathematics of Operations Research*, 1979, *4*, p. 233-235.
2. V. Chvátal, *Linear Programming*, Freeman, New York, 1983.
3. H. Dyckhoff, A typology of cutting and packing problems, *European Journal of Operational Research*, 1990, *44*, p. 145-159.
4. G. Dobson, Worst-case analysis of greedy heuristics for integer programming with nonnegative data, *Mathematics of Operations Research*, 1982, *7*, p. 515-531.
5. P. C. Gilmore, R. E. Gomory, A linear programming approach to the cutting stock problem, *Operations Research*, 1963, *9*, p. 849-859.
6. M. X. Goemans, D. P. Williamson, *A new 3/4-approximation algorithm for MAX SAT*, In Proc. of the third IPCO Conference, G. Rinaldi and L. Wolsey eds., 1993, p. 313-321.

7. R. W. Haessler, P. E. Sweeney, Cutting stock problems and solution procedures, *European Journal of Operational Research*, 1991, *54*, p. 141-150.
8. P. Hansen, B. Jaumard, M. Poggi de Aragão, Column generation methods for probabilistic logic, *ORSA Journal on Computing*, 1991, *3*, p. 135-148.
9. D. S. Johnson, Approximation algorithms for combinatorial problems, *Journal of Computer and System Sciences*, 1974, *9*, p. 256-278.
10. D. S. Johnson, *Worst-case behavior of graph coloring algorithms*, In Proc. 5th Southeastern Conf. on Combinatorics, Graph Theory and Computing, p. 513-527. Utilitas Mathematica Publ. Winnipeg, Ontario, 1974 *b*.
11. D. Kavvadias, C. P. Papadimitriou, A linear programming approach to reasoning about probabilities, *Annals of Mathematics and Artificial Intelligence*, 1990, *1*, p. 189-205.
12. L. Lovász, On the ratio of optimal integral and fractional covers, *Discrete Mathematics*, 1974, *13*, p. 383-390.
13. C. Lund, M. Yannakakis, *On the hardness of approximating minimization problems*, Working Paper AT&T Bell Labs, NJ, 1992.
14. M. Minoux, A class of combinatorial problems with polynomially solvable large scale set covering/partioning relaxations, *R.A.I.R.O. Recherche opérationnelle/Operations Research*, 1987, *21*, p. 105-136.
15. G. M. Nemhauser, L. A. Wolsey, *Integer and Combinatorial Optimization*, Wiley-Interscience Series in Discrete Mathematics and Optimization, John Wiley & Sons, New York Chichester Brisbane Toronto Singapore, 1988.
16. N. J. Nilsson, Probabilistic logic, *Artificial Intelligence*, 1986, *28*, p. 71-87.
17. H. U. Simon, *The analysis of dynamic and hybrid channel assignment*, Working Paper, Universität des Saarlandes, Saarbrücken, 1988.
18. H. U. Simon, On approximate solutions for combinatorial optimization problems, *SIAM Journal on Discrete Mathematics*, 1990, *3*, p. 294-310.