RAIRO. RECHERCHE OPÉRATIONNELLE

M. MINOUX
E. PINSON

Lower bounds to the graph partitioning problem through generalized linear programming and network flows

RAIRO. Recherche opérationnelle, tome 21, n° 4 (1987), p. 349-364

<http://www.numdam.org/item?id=RO_1987__21_4_349_0>

R.A.I.R.O. Recherche opérationnelle/Operations Research
(vol. 21, n° 4, novembre 1987, p. 349 à 364)

LOWER BOUNDS TO THE GRAPH PARTITIONING PROBLEM THROUGH GENERALIZED LINEAR PROGRAMMING AND NETWORK FLOWS (*)

by M. MINOUX (1) and E. PINSON (2)

Abstract. — *The well-known Graph Partitioning Problem (GPP), has many applications, both in its weighted and unweighted versions: optimal VLSI circuit design and layout, systems analysis, data analysis and clustering, decomposition of large scale mathematical programming problems, etc. We investigate here a new way of getting lower bounds based on a reformulation as a large scale set partitioning problem, where the columns correspond to all subsets of the vertex set X. It is shown that the continuous relaxation of this large scale problem can be solved* exactly *by means of a generalized linear programming technique, the column generation process being reducible to maximum network flow computations. Preliminary computational results on a number of small-to-medium sized problems are reported, and directions for future investigations are suggested in the conclusion.*

Keywords : Graph partitioning; combinatorial optimization; generalized linear programming; quadratic pseudoboolean optimization; maximum network flows.

Résumé. — *Le problème de partitionnement de graphes est bien connu pour ses nombreuses applications, aussi bien dans sa version pondérée, que dans sa version non pondérée : optimisation de la conception et de l'implantation des circuits VLSI, analyse de systèmes, analyse de données et classification, décomposition de grands problèmes de programmation mathématique, etc. On étudie ici une nouvelle méthode pour obtenir des bornes inférieures, basée sur une reformulation en un problème de partitionnement de grande dimension, où les colonnes correspondent à tous les sous-ensembles de l'ensemble X des sommets du graphe. On montre que la relaxation continue de ce grand problème peut être résolue* de façon exacte *par une technique de programmation linéaire généralisée, la procédure de génération de colonnes pouvant se réduire à des calculs de flot maximum. Des résultats de calcul préliminaires sur des exemples de taille petite ou moyenne sont fournis, et des directions de recherche future sont suggérées en conclusion.*

Mots clés : Partitionnement de graphe; optimisation combinatoire; programmation linéaire généralisée; optimisation pseudo-booléenne quadratique; flot maximum.

(*) Received June 1987.
(1) Laboratoire M.A.S.I., Université Paris-VI, place Jussieu, 75005 Paris.
(2) Institut de Mathématiques Appliquées, Université Catholique de l'Ouest, Angers.

R.A.I.R.O. Recherche opérationnelle/Operations Research, 0399-0559/87/04 349 16/$ 3.60

1. INTRODUCTION AND PROBLEM STATEMENT

1.1. The (unweighted) graph partitioning problem (GPP)

Let $G=[X, U]$ a nondirected graph where X is the set of *nodes* ($|X|=N$) and U the set of *edges* ($|U|=M$) and let p be a given positive integer ($1<p<N$).

The *Graph Partitioning Problem* (GPP) is to find a *partition* of the vertex set X into p subsets $S_1, S_2, \ldots, S_p$, so as to maximize the quantity

$$z=\theta(S_1)+\theta(S_2)+\ldots+\theta(S_p)$$

(where, $\forall S \subset X$, $\theta(S)$ is the number of edges having both endpoints in S), the maximum above being taken over all possible partitions of X into p subsets. Note that an equivalent way of stating the problem would be to minimize the total number of edges linking two distinct subsets in the partition.

In this form, (GPP) arises in many contexts of application such as:

– decomposing an electronic circuit diagram into a prescribed number of modules so as to minimize the interconnections between distinct modules;

– in systems analysis, decomposing a given large scale system into smaller subsystems, in order to minimize the interdependencies between subsystems;

– Given a large scale linear programming problem, with *sparse* constraint matrix, find an optimal "decomposed form" of the problem so as to obtain as few coupling constraints (resp.: coupling variables) as possible. The interest here lies in the fact that, if a good decomposition can be found, then *decomposition techniques* such as Dantzig-Wolfe (case of coupling constraints) or Benders (case of coupling variables) can be applied (see Dantzig and Wolfe 1961, Benders 1962). The above problem is a special case of graph partitioning where the given graph is *bipartite* (the vertex set has elements corresponding to variables *and* constraints of the problem, and there is an edge between "variable j" and "constraint i" iff. variable j appears in the i-th constraint).

1.2. The weighted graph partitioning problem (WGPP) with an application to quadratic pseudoboolean programming

The weighted version of the Graph Partitioning Problem is an extension in which weights $w_{ij}>0$ (*) are assigned to the edges (i, j) in G. It is then

(*) It will be seen in § 3.1 below why the assumption $w_{ij}>0$ is necessary.

required to find a partition of the vertex set so that the total weight of the edges within the subsets of the partition is *maximized* (equivalently: the total weight of the edges joining pairs of distinct subsets be *minimized*).

Apart from the fact that many of the GPP's arising from practical applications (such as those mentioned in section 1.1) can be stated with weights, it is of interest to point out here one more application which essentially requires the (WGPP) formulation.

A fundamental problem in combinatorial optimization is to find the minimum of a quadratic function in n boolean variables such as:

$$f(x) = \sum_{i=1}^{n} l_i x_i + \sum_{(i,j) \in T} q_{ij} x_i x_j.$$

where:

- T is a subset of pairs of indices (i, j)
- $l_i (i = 1, \ldots, n)$ are the coefficients of the linear terms.
- q_{ij} are the coefficients of the quadratic terms.

The above problem is *NP*-hard since it includes as special cases:

- the maximum (weight) vertex packing problem (or stable set of a graph);
- the maximum cut in a network;

and other well-known difficult combinatorial problems. Also remember that the general problem of maximizing a pseudoboolean function of any degree can be reduced to the quadratic case (Rosenberg 1975). It follows that, in general, the quadratic pseudoboolean optimization problem can only be solved to optimality when the number of variables is rather small (say, not more than about 50 variables according to the computational results reported in Williams 1985).

Now, suppose that the matrix $Q = (q_{ij})$ is *sparse*, i.e. contains only a small proportion of nonzero coefficients. Then a possible approach to solve a large quadratic pseudoboolean minimization problem is to look for a partition of the variables into subsets, in such a way that the quadratic terms involving variables in two distinct subsets (the interactions) be as small as possible. If such a partition is obtained with subsets of sufficiently small size, then forgetting the interaction terms, we get an *approximate problem* which is solvable since it decomposes into a number of independent subproblems of small size.

In order to do that, the partitioning problem to be solved is of course a weighted GPP on the graph with n vertices corresponding the n 0-1 variables,

and where an edge with weight $|q_{ij}|$ is associated with every nonzero coefficient in matrix Q.

If, in addition, we want to get an approximation providing a *lower bound* to the quadratic problem, only edges with positive q_{ij} must be allowed to link two distinct subsets of vertices. One way of ensuring this is to assign to the edges (i, j) with negative associated q_{ij} a sufficiently large weight $\bar{w}$.

1.3. Previous attempts at solving (GPP) and (WGPP)

Due to the importance and diversity of its applications, the Graph Partitioning Problem has attracted much interest. However, apart from the case of problems with very small size ($N<10$ say) or the case of special instances of the problem (e. g. : $p=2$) no general algorithmic procedure, significantly more efficient than brute-force enumeration, seems to be available at present for getting exact optimal integer solutions (*NP*-hardness of the problem was proved by Hyafil and Rivest 1973). Given the large sizes of the instances arising in practice, as well as the apparent difficulty of the problem, research in the area mainly concentrated on efficient approximate procedures. Most of them actually derive from a few basic and simple heuristic ideas (such as "local improvement" methods, randomization techniques—e. g. the fairly recent "simulated annealing" method—) often rediscovered many times in the various contexts of application.

The first exact algorithm using branch and bound techniques for solving the optimal bipartition problem (i. e. the special case where $p=2$) can be found in Christofides and Brooker (1976). Much more recently, Hansen and Roucairol (1987) showed that, after reformulating the bipartition problem as a quadratic pseudoboolean program with a cardinality constraint and applying lagrangean relaxation, lower bounds significantly sharper than those used in Christofides and Brooker could be derived, resulting in improved tree search procedures.

In the rest of the paper, we will deal with the most general version of the problem, i. e. optimal graph partitioning with arbitrary (nonnegative) weights and arbitrary number p of subsets in the required partition.

2. FORMULATION AS A LARGE SCALE SET PARTITIONING PROBLEM

Suppose that the subsets of X are numbered $k=1, 2, \ldots, K$ ($K=2^N$) and let $A=(a_{ik})_{i=1, \ldots, N,\ k=1, \ldots, K}$ be the incidence matrix of $\mathscr{P}(X)$ in X.

Thus, for the subset having index number k ($1 \leqq k \leqq K$), $a_{ik}=1$ if i is an element of the subset, $a_{ik}=0$ otherwise. To each subset k, we let correspond a binary variable x_k to express the fact that the subset is, or is not, selected in the partition, and a *cost* c_k defined by:

$$c_k = \sum_{(i,\, j) \in U_k} w_{ij} \tag{1}$$

where U_k is the subset of edges (s, t) in U such as $a_{sk}=1$ and $a_{tk}=1$ (thus c_R is the sum of weights of edges with both endpoints in subset k).

With the above notation, (WGPP) can be restated as the (large scale) *set partitioning problem:*

$$\text{(SPP)} \quad \begin{cases} \text{Minimize} - \sum_{k=1}^{K} c_k . x_k \\ \text{subject to:} \\ \qquad A . x = \mathbf{1} & (2) \\ \qquad \sum_{k=1}^{K} x_k = p & (3) \\ \qquad x \in \{0, 1\}^K \end{cases}$$

($\mathbf{1}$ denotes the N-vector with all components equal to 1.)

Obviously, except for the case of very small values of N ($N \leqq 12$ say), there is no hope for solving it directly as an integer 0-1 program in view of the huge number of variables. However, as pointed out in Minoux (1986) even in the case of very large scale instances it is possible to exploit the special structure to solve *exactly* its *continuous relaxation* which reads:

$$\overline{\text{(SPP)}} \quad \begin{cases} \text{Minimize} - \sum_{k=1}^{K} c_k . x_k \\ \text{subject to:} \\ \qquad A . x = \mathbf{1} & (2) \\ \qquad \sum_{k=1}^{K} x_k = p & (3) \\ \qquad x_k \geqq 0 \quad (k = 1, \ldots, K) \end{cases}$$

Obviously, an optimal solution to $(\overline{SPP})$ provides a *lower bound* to the optimal integer solution value which, as is well-known, may subsequently be exploited (e. g. within branch and bound procedures) to help in the search for optimal integer solutions.

3. SOLUTION OF THE CONTINUOUS RELAXATION BY GENERALIZED LINEAR PROGRAMMING

3.1. Basic principle

Generalized linear programming was first discovered by Dantzig and Wolfe (1961) as a basic ingredient of the *decomposition principle.* But it has also been known for a long time as an efficient general tool for solving large scale, implicitly defined, linear programming problems with special structure (*see* e. g. Dantzig 1963).

We show below how it specializes to $(\overline{SPP})$.

Since the number of columns is usually much too large to state them all explicitly, one has to consider, at each step, a *restricted problem* consisting of only a few columns which are made explicit. At the start, we may consider for instance, the restricted problem containing the N columns corresponding to subsets of one element (each containing a single vertex of the graph), the associated cost coefficients being 0. Or, we may start with a restricted problem containing p columns associated with the p subsets in a partition forming a good approximate solution to the (integer) problem.

At any current step of the procedure, a (continuous) optimal solution to the restricted problem is looked for, by applying e. g. the primal revised simplex algorithm. As a result of this computation, we have optimal simplex multipliers $\pi_1, \pi_2, \ldots, \pi_N$ [associated with constraints (2)] and μ [associated with constraint (3)].

The next task to carry out is then, based on the above simplex multipliers, to *price-out* i. e. to look for the *minimum reduced cost column* over the whole set of columns of $(\overline{SPP})$. If this best column can be obtained efficiently enough (i. e. without need for enumerating all the columns) then all the ingredients necessary to have the primal simplex algorithm work are at hand: if the selected column has nonnegative reduced cost, then an optimal solution (to the whole problem) has been obtained, and the algorithm ends up. If the selected column has strictly negative reduced cost, then the current restricted problem is augmented by this new column and reoptimized (the newly

generated column thus enters the basis at the first pivot step of the simplex algorithm).

It remains to be shown that the crucial part of the above process – generating a minimum reduced cost column – can indeed be efficiently carried out in the case of $(\overline{SPP})$.

3.2. The column generation subproblem as a quadratic pseudoboolean minimization problem (Minoux 1986)

The column generation subproblem to be solved when applying generalized linear programming to $(\overline{SPP})$ can be stated formally as follows:

(CG) Given any set of multipliers $\pi_i (i=1, \ldots, N)$ and μ associated with constraints (2) and (3) of $(\overline{SPP})$ respectively, find a column k_0 with *minimum reduced cost*, i. e. satisfying:

$$\bar{c}_{k_0} = -c_{k_0} - \sum_{i=1}^{N} \pi_i a_{ik_0} - \mu = \operatorname*{Min}_{1 \leqq k \leqq K} \left\{ -c_k - \sum_{i=1}^{N} \pi_i a_{ik} - \mu \right\}.$$

To show that, indeed, (CG) can be solved efficiently (in polynomial time), we first reformulate it as the minimization of a *quadratic pseudoboolean function* of a special kind.

Associating with each element i of X a boolean variable y_i, the cost of any subset $S \subset X$ with characteristic vector $y=(y_i)_{i \in X}$ is given by:

$$c(S) = \sum_{(i,j) \in U} w_{ij} y_i y_j$$

(the w_{ij} being the weights attached to the edges of the graph, see § 1-2).

In view of this, the column generation subproblem (CG) can be rewritten as:

$$\operatorname*{Minimize}_{y \in \{0,1\}^N} \left\{ - \sum_{(i,j) \in U} w_{ij} y_i y_j - \sum_{i=1}^{N} \pi_i y_i \right\} \tag{4}$$

(the constant μ has no influence on the determination of the optimal subset).

Quadratic 0-1 programming is difficult in general (see § 1-2) but, since we assumed $w_{ij} \geqq 0$, it can be seen that problem (4) concerns the special case where all quadratic terms have nonpositive coefficients, thus can be reduced to a *maximum network flow problem* (see e. g. Rhys 1970).

3.3. Reduction of column generation to a maximum network flow problem

To show exactly how this can be done in the case of the quadratic problem (4), it is convenient to set up the *inverse reduction.*

To that aim, we build up a capacitated network Γ having vertex set $X \cup \{s\} \cup \{t\}$ where s and t are two additional vertices ("source" and "sink"). To each edge (i, j) of G we let correspond a (nondirected) link with capacity h_{ij} between vertices i and j in Γ. We also add N edges of the form (s, i) with capacities a_i and N edges of the form (j, t) with capacities b_j (see Figure 1). The values of h_{ij}, a_i and b_j will be specified below.

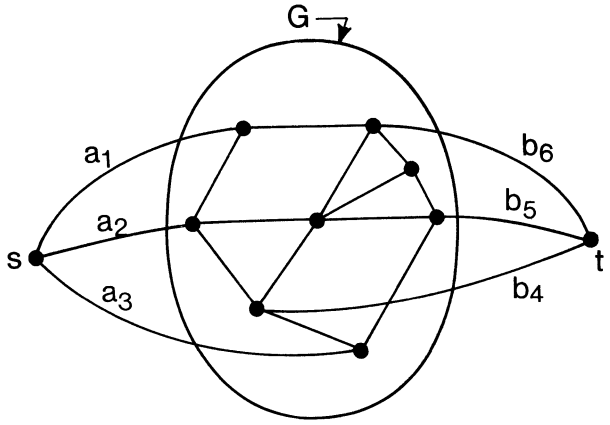

Figure 1. — Construction of the capacitated network Γ.

Let $S \subset X \cup \{s\} \cup \{t\}$ be any subset of nodes such that $s \in S$, $t \notin S$. If we associate, with each node $i \in X$ a 0-1 variable u_i, with the meaning that $u_i = 1$ iff $i \in S$, we see that the capacity of the $s-t$ cut in Γ defined by S is:

$$\sum_{i=1}^{N} a_i(1-u_i) + \sum_{i=1}^{N} b_i u_i + \sum_{(i,\, j) \in U} h_{ij}[u_i(1-u_j) + u_j(1-u_i)].$$

Thus, the minimum $s-t$ cut problem on Γ is equivalent to minimizing:

$$-\sum_{(i,\, j) \in U} 2 h_{ij} u_i u_j + \sum_{i=1}^{N} (\gamma_i - a_i + b_i) u_i \tag{5}$$

where, $\forall i$, γ_i denotes the sum: $\sum_{j/(i,\, j) \in U} h_{ij}$.

Now, simple identification between (4) and (5) shows how to assign the capacities on Γ in order to get the equivalence:

$$h_{ij} = \frac{w_{ij}}{2} \qquad (\forall (i, j) \in U)$$

and, after having computed the γ_i values:

$$a_i = \gamma_i + \pi_i \quad \text{and} \quad b_i = 0 \quad \text{if } \gamma_i + \pi_i > 0$$

$$a_i = 0 \quad \textit{and} \quad b_i = -\gamma_i - \pi_i \quad \text{if } \gamma_i + \pi_i < 0$$

$$a_i = 0 \quad \text{and} \quad b_i = 0 \quad \text{if } \gamma_i + \pi_i = 0.$$

In view of this, we conclude that the column generation subproblem (4) can be solved as a maximum network flow problem on a capacitated network having $N+2$ nodes. The computational complexity is thus $O(N^3)$ by using the algorithms described by Dinic (1970) and Karzanov (1974) or the simplified version given by Tarjan (1984).

Let us mention here that the polynomial solvability of the column generation subproblem also implies the polynomial solvability of the (large scale) linear relaxation $(\overline{SPP})$, and that this property holds for a wide class of combinatorial problems (including the Graph Partitioning Problem) as shown in Minoux (1986).

4. PRELIMINARY COMPUTATIONAL RESULTS

We present here some preliminary computational experience obtained with the column generation procedure when applied to large scale set partitioning problems arising from a number of example graphs with sizes ranging from 12 nodes and 19 edges, to 40 nodes and 90 edges (note that for the largest problem treated, the total number of implicit columns in the set partitioning model already reaches 2^{40}, a very large number indeed). In all the examples treated, the unweighted version was considered (all edge weights equal to unity).

The tests were carried out with the following implementation of the method:

(1) only one column is generated [by solving the quadratic pseudoboolean subproblem (4)] each time the column generation procedure is called for, i. e. each time a maximum network flow computation is performed.

Though *multiple pricing* (which consists of generating more than one negative reduced cost column each time the column generation subproblem is

solved) could easily be implemented in our context, this possibility was not used at this early stage.

(2) Each time one extra column with strictly negative reduced cost is generated, a complete reoptimization process (possibly implying several consecutive pivot steps) is applied to the augmented restricted problem before the next call to the column generation procedure.

(3) The starting restricted problem is formed by $N+p$ columns selected as follows:

(*a*) N columns consisting of the N unit vectors corresponding to all the node subsets with one element, the associated cost coefficients being 0;

(*b*) p columns corresponding to the particular subsets

$$S_1=\{1, 2, \ldots, k\}; \qquad S_2=\{k+1, k+2, \ldots, 2k\}$$

$$S_p=\{(p-1)k+1, (p-1)k+2, \ldots, N\},$$

where $k=[N/p]$. Of course the cost coefficients of the subsets $S_1, \ldots, S_p$ above are computed according to formula (1) of section 2 (also note that the p-th subset S_p usually contains more than k elements, except if N is a multiple of p).

For each of the 24 test problems treated, *Table I* indicates:

– the problem number;

– the size (number of nodes, number of edges) of the corresponding graph;

– the number p of subsets in the required partition;

– the cost function value $\hat{z}$ of the best available solution (it is thus an upper bound to the optimal integer value);

– the cost function value z_c^* corresponding to the optimum continuous solution to the large scale set partitioning problem: this is, of course, a lower bound to the optimum value of an integer solution;

– a mention about the integrality/nonintegrality of the generalized *LP* solution;

– the total number of columns generated in the course of the generalized linear programming procedure (i. e. the total number of maximum network flow computations);

– the value $\delta=|(\hat{z}-z_c^*)/\hat{z}|$, i. e. the percentage difference between $\hat{z}$ and z_c^*, an upper bound to the integrality gap.

Figures 2 to 5 provide details about the structure of the graphs corresponding to the first four examples in Table I, together with an indication of the best known solution to each problem.

The results displayed on Table I suggest the following observations.

(1) In spite of the fact that, in our experiments, the starting restricted problem was derived from a feasible integer solution usually rather far from the optimum, the total number of generated columns is seen to stay very low and seems to increase rather slowly with problem size. Even though the use of multiple pricing (alluded to above) might further reduce the total number

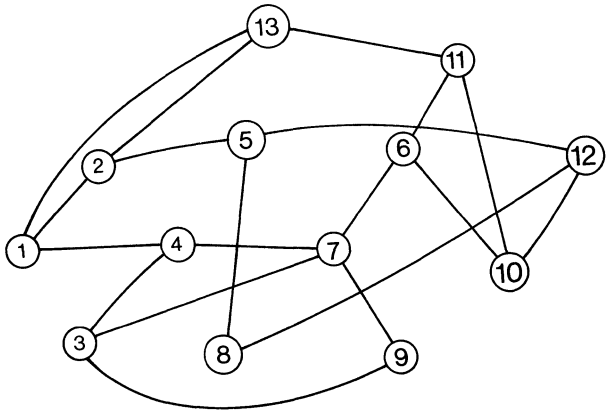

Figure 2. – Example 1 with 12 nodes and 19 edges. A solution for $p=4$ classes has value 14 (5 edges between distinct classes) the associated partition being: {1, 2, 13}, {3, 4, 7, 9}, {5, 8, 12}, {6, 10, 11}.

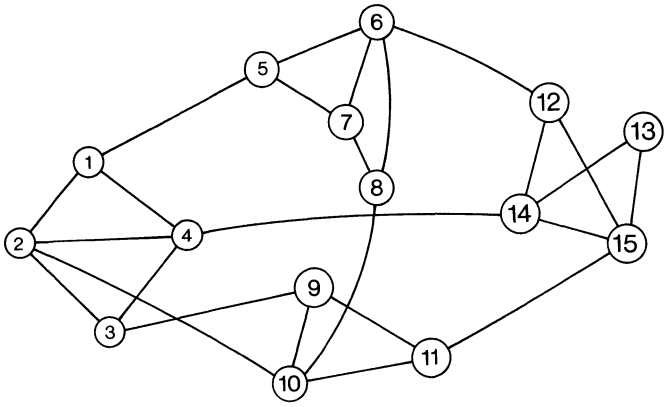

Figure 3. – Example 2 with 15 nodes and 26 edges. A solution for $p=4$ classes has value $z=18$ (8 edges between distinct classes) a possible partition being: {1, 2, 3, 4}, {5, 6, 7, 8}, {9, 10, 11}, {12, 13, 14, 15}.

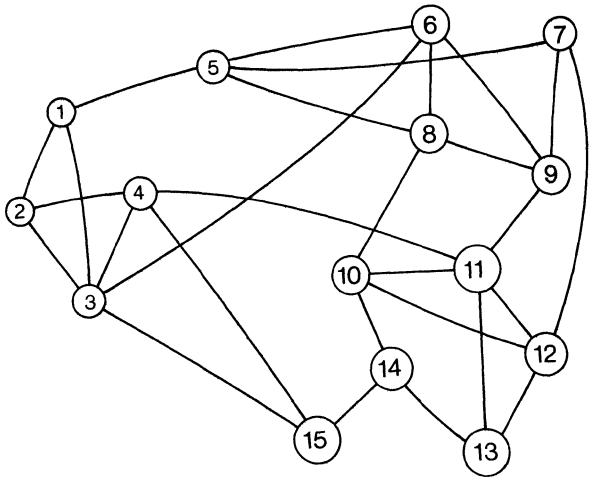

Figure 4. – Example 3 with 15 nodes and 28 edges. A solution for $p=3$ classes has value $z=22$ (6 edges between distinct classes) the associated partition being $=\{1\}, \{7\}, \{2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 13, 14, 15\}$.

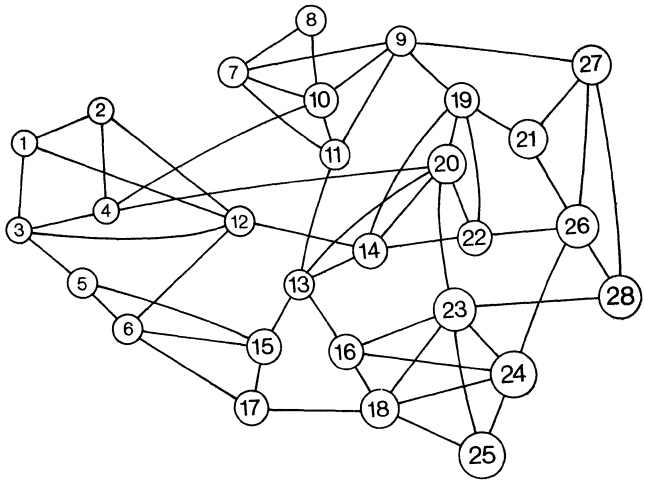

Figure 5. – Example 4 with 28 nodes and 58 edges. A solution for $p=6$ classes has value $z=44$ (14 edges between distinct classes) the associated partition being: $\{1\}, \{5\}, \{17\}, \{8\}, \{28\}, \{2, 3, 4, 6, 7, 9, 10, 11, 12, 13, 14, 15, 16, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27\}$.

TABLE I

Computational results obtained on a series of small to medium sized examples

Problem number	N, M number of nodes and edges	p number of classes	$\hat{z}$ value of best available solution	z_c^* optimum continuous generalized LP solution	Integrality of the generalized LP solution	Total number of generated columns	δ upper bound to integrality gap (%)
1	12,19	4	−14	−14	yes	6	0
2	15,26	4	−18	−19	no	13	5.5
3	15,28	3	−22	−24	no	15	9
4	28,58	6	−44	−47	no	23	6.8
5	35,72	7	−54	−60	no	42	11
6	34,69	3	−55	−59	no	48	7.2
7	35,75	3	−56	−60	no	53	7.1
8	36,78	3	−60	−65	no	55	8.3
9	34,60	4	−44	−52	no	47	18
10	35,58	4	−43	−47	no	49	9.3
11	36,70	4	−52	−56	no	52	7.6
12	34,69	5	−51	−54	no	52	5.8
13	35,70	5	−53	−55	no	54	3.7
14	36,76	5	−58	−63	no	52	8.6
15	29,62	6	−46	−50	no	42	8.6
16	30,66	6	−50	−54	no	40	8
17	31,70	6	−54	−58	no	63	7.4
18	32,74	6	−58	−61	no	62	5.1
19	33,78	6	−62	−65	no	60	4.8
20	36,75	7	−57	−60	no	50	5.2
21	37,79	7	−61	−64	no	45	4.9
22	38,82	7	−64	−68	no	56	6.2
23	39,86	7	−68	−71	no	91	4.4
24	40,90	7	−72	−75	no	85	4.1

of maximum network flow computations, these results already clearly demonstrate the efficiency of the column generation procedure at obtaining the relaxed LP lower bound.

(2) A significant proportion of zero δ values was only observed for very small sized problems, typically less than 10 nodes. (example 1 with 12 nodes was the largest one for which this occured in our experiments). For all the other problems treated, the corresponding optimal generalized *LP* solutions were always fractional, the values of δ (a measure of the integrality gap) typically ranging from 5 to 10%.

This tends to suggest that the sharp bounds which are usually necessary to have Branch and Bound procedures work efficiently cannot be expected from the generalized linear programming approach to the Graph Partitioning Problem. This contrasts, for instance, with the results obtained with the same technique on the optimum weighted edge coloring problem (a matrix decomposition problem arising in the context of optimal traffic assignment in Satellite Communications) studied in Minoux (1984): in this case, the lower bounds were observed to be very sharp, which subsequently made possible an efficient implementation of a Branch and Bound procedure for getting exact optimal integer solutions (*see* Ribeiro, Minoux and Penna 1986). Though both problems are known to be *NP*-hard, the large values of the integrality gaps obtained for the Graph Partitioning Problem indeed tend to suggest that this problem is intrinsically much harder than the optimum weighted edge coloring problem mentioned above.

5. CONCLUSIONS AND DIRECTIONS FOR FUTURE WORK

We have presented a computationally efficient technique based on generalized linear programming and the solution of maximum network flow subproblems for deriving lower bounds to the general weighted Graph Partitioning Problem with an arbitrary number p of classes. Clearly, using ideas similar to those presented in Ribeiro, Minoux and Penna (1986), this lower bounding scheme could be, at least "formally", combined with Branch and Bound Techniques to produce an exact solution algorithm. Preliminary computational experience suggests, however, that such an algorithm is likely to be more "conceptual" than practical.

In spite of this, the basic ideas suggested here indoubtedly deserve further investigations, and we think that this generalized linear programming approach could be a well-suited tool for dealing with some meaningful

variants of the pure graph partitioning problem. A typical example of such a variant is when the partitions which are to be determined are required to be "balanced" i. e. that all the subsets in a partition all have about the same cardinality. One practical way of ensuring this is to impose, on each subset of the partition looked for, a cardinality constraint of the form $q_{\min} \leqq$ cardinality of the subset $\leqq q_{\max}$, where both $q_{\min}$ and $q_{\max}$ are integers close to N/p (of course, $q_{\min}$ and $q_{\max}$ must satisfy: $p \times q_{\min} \leqq N$ and $p \times q_{\max} \geqq N$).

One of the interesting features of the generalized linear programming approach is its flexibility with respect to such additional constraints involving individual subsets in the partition. In the particular case of cardinality constraints mentioned above, the column generation subproblem now becomes a constrained maximum network flow problem which (though *NP*-hard) can be handled through lagrangean relaxation and branch and bound techniques. Implementation of an extension of the generalized linear programming approach to this variant of the Graph Partitioning Problem is currently being studied.

ACKNOWLEDGMENTS

We want to thank J. Teneur for his participation in the preliminary reflexions about the implementation of the method described here. Comments of an anonymous referee were also appreciated.

REFERENCES

BARNES E. R., *An Algorithm for Partitioning the Nodes of a Graph*, SIAM J. Alg. Discr. Meth. Vol. 4, 1982, pp. 541-550.

BENDERS J. F., *Partitioning Procedures for solving mixed variables programming problems*, Numerische Mathematik, Vol. 4, 1962, pp. 238-252.

CHRISTOFIDES N. and BROOKER P., *The Optimal Partitioning of Graphs*, SIAM J. Appl. Math. Vol. 30, No. 1, 1976, pp. 55-70.

DANTZIG G. B. and WOLFE P., *The Decomposition Algorithm for Linear Programming*, Econometrica, Vol. 29, No. 4, 1961, pp. 767, 778.

DINIC E. A., *Algorithm for Solution of a Problem of Maximum Flow in a Network with Power Estimation*, Soviet Math. Dokl., Vol. 11, 1970, pp. 1277-1280.

HAMMER P. L., HANSEN P. and SIMEONE B., *Roof Duality, Complementation and Persistency in Quadratic* 0-1 *Optimization*, Mathematical Programming, Vol. 28, 1984, pp. 121-155.

HANSEN P. and ROUCAIROL C., *Problème de la bipartition minimale d'un graphe*, RAIRO (à paraître).

HYAFIL L. and RIVEST R. L., *Graph Partitioning and Constructing Optimal Decision Trees are Polynomial Complete Problems*, Report n° 33, IRIA-Laboria, Rocquencourt, France, 1973.

KARZANOV A. V., *Determining the Maximal Flow in a Network by the Method of Preflows*, Soviet Math. Dokl. Vol. 15, 1974, pp. 434-437.

LU S. M. and WILLIAMS A. C., *Roof Duality for Nonlinear* 0-1 *Optimization*, Rutcor Research Report, RRR 2-85, 1985.

MINOUX M., *Programmation Mathématique : théorie et algorithmes*, Dunod, Paris, 1983 (English translation J. Wiley & Sons, 1986).

MINOUX M. *Optimal Traffic Assignment in a SS/TDMA Frame: a New Approach by Set Covering and Column Generation*, ORSA/TIMS meeting, Dallas, Texas, 1984, Appeared in RAIRO, Vol. 20, No. 4, 1986, pp. 1-13.

MINOUX M., *A Class of Combinatorial Problems with Polynomially Solvable Large Scale set Covering/Partitioning Relaxations*, Journées du 20e anniversaire du Groupe Combinatoire de l'AFCET, Paris-3, 5 décembre 1986, appeared in RAIRO, Vol. 21, No. 2, 1987, pp. 105-136.

RHYS J. M. W., *A Selection Problem of Shared Fixed Costs and Network Flows*, Management Science, Vol. 17, No. 3 (1970), pp. 200-207.

RIBEIRO C., MINOUX M. et PENNA C., *A Combined Branch and Bound/Column generation Approach to very large Scale Set Covering Problems with Special Structure*, ORSA/-TIMS meeting, Miami, Florida, November 1986, (to appear).

ROSENBERG I. G., *Reduction of Bivalent Maximization to the Quadratic Case*, Cahiers du Centre d'Études de Recherche Operationnelle, Vol. 17, 1975, pp. 71-79.

TARJAN R. E., *A Simple Version of Karzanov's Blocking Flow Algorithm*, Operations Research Letters, Vol. 2, No. 6 (1984), pp. 265-268.

WILLAMS A. C., *Quadratic* 0-1 *Programming Using the roof Dual with Computational Results*, RUTCOR Research Report RRR 8-85, 1985.