Titre: Title:	Network flow solution of some non linear 0-1 programming problems and applications to graph theory
Auteurs: Authors:	Jean-Claude Picard, & Maurice Queyranne
Date:	1979
Type:	Rapport / Report
Référence: Citation:	Picard, J.-C., & Queyranne, M. (1979). Network flow solution of some non linear 0-1 programming problems and applications to graph theory. (Rapport technique n° EP-R-79-14). https://publications.polymtl.ca/5991/

Institution:	École Polytechnique de Montréal
Numéro de rapport: Report number:	EP-R-79-14
URL officiel: Official URL:	
Mention légale: Legal notice:	

Génie Industriel

A NETWORK FLOW SOLUTION OF SOME NON LINEAR 0-1 PROGRAMMING PROBLEMS AND APPLICATIONS TO GRAPH THEORY

BY JEAN-CLAUDE PICARD

AND MAURICE QUEYRANNE

DEPARTEMENT DE GENIE INDUSTRIEL

ECOLE POLYTECHNIQUE DE MONTREAL.

APRIL 1979. RAPPORT TECHNIQUE NO. EP-79-R-14

Ecole Polytechnique de Montréal

Campus de l'Université de Montréal
Case postale 6079
Succursale 'A'
Montréal, Québec
H3C 3A7

A NETWORK FLOW SOLUTION OF SOME NON LINEAR 0-1 PROGRAMMING PROBLEMS AND APPLICATIONS TO GRAPH THEORY

DON

BY JEAN-CLAUDE PICARD

AND MAURICE QUEYRANNE

DEPARTEMENT DE GENIE INDUSTRIEL

ECOLE POLYTECHNIQUE

APRIL 1979

RAPPORT TECHNIQUE EP-79-R-14

ABSTRACT

A network flow technique is used for solving the unconstrained nonlinear 0-1 programming problem, which is maximizing the ratio of two polynomials, assuming that all the nonlinear coefficients in the numerator are non-negative and all the nonlinear coefficients in the denominator are non-positive. The proposed algorithm requires the solution of a sequence of minimum cut problems in a related network, and can be extended to some more general problems of the same type. This approach is applied to find the density of a graph (the maximum ratio, among its subgraphs, of the number of edges to the number of nodes) and its arboricity, for which polynomial algorithms are described. It is also useful by providing a bounding scheme for the maximum clique and vertex packing problems.

1. INTRODUCTION

Consider the following 0-1 programming problem (P)

$$\text{(P)} \qquad \text{Max} \quad f(X) = \sum_{S \in \mathcal{A}} a_S \prod_{i \in S} x_i$$

$$x_i = 0,1 \qquad i = 1,2,\ldots,n$$

where $\mathcal{A} \subseteq \mathcal{P}(\{1,2,\ldots,n\})$ is a family of subsets of $\{1,2,\ldots,n\}$ and $a_S \geq 0$ for all S such that $|S| \geq 2$. This problem can be seen as the unconstrained bivalent maximization of a posynomial, with linear terms of arbitrary sign. Let $\mathcal{A}'$ denote the set of all $S \in \mathcal{A}$ such that $|S| \geq 2$ and $y_S = \prod_{i \in S} x_i$ for all $S \in \mathcal{A}'$; problem (P) becomes equivalent to problem (P_1)

$$(P_1) \qquad \begin{aligned} &\text{Max} \quad f(X,Y) = \sum_{S \in \mathcal{A}'} a_S y_S + \sum_{i=1}^{n} a_i x_i \\ &\text{s.t} \qquad y_S \leq x_i \text{ all } i \in S \text{ and all } S \in \mathcal{A}' \\ &\qquad\qquad y_S = 0,1 \quad , \text{ all } S \in \mathcal{A}' \\ &\qquad\qquad x_i = 0,1 \qquad i = 1,2,\ldots,n \end{aligned}$$

(where a_i denotes the value $a_{\{i\}}$ of the singleton $\{i\}$). Without loss of generality we can assume $a_i < 0$, since $a_i \geq 0$ will imply $x_i = 1$ in an optimal solution. (P_1) can be seen as a selection problem, as defined by Rhys [23] and Balinski [2] or the problem of finding a maximal closure of a graph as defined by Picard [16].

A closure of a directed graph G is defined as a subset of nodes such that if a node belongs to the closure, then all its successors also belong to the set. If a weight is associated to each node of G the weight of a closure is the sum of the weights of the nodes in it and a maximal closure is defined as a closure of maximal weight. (P_1) is the selection problem, i.e. the problem of finding a maximal closure in the bipartite graph given in Fig. 1.

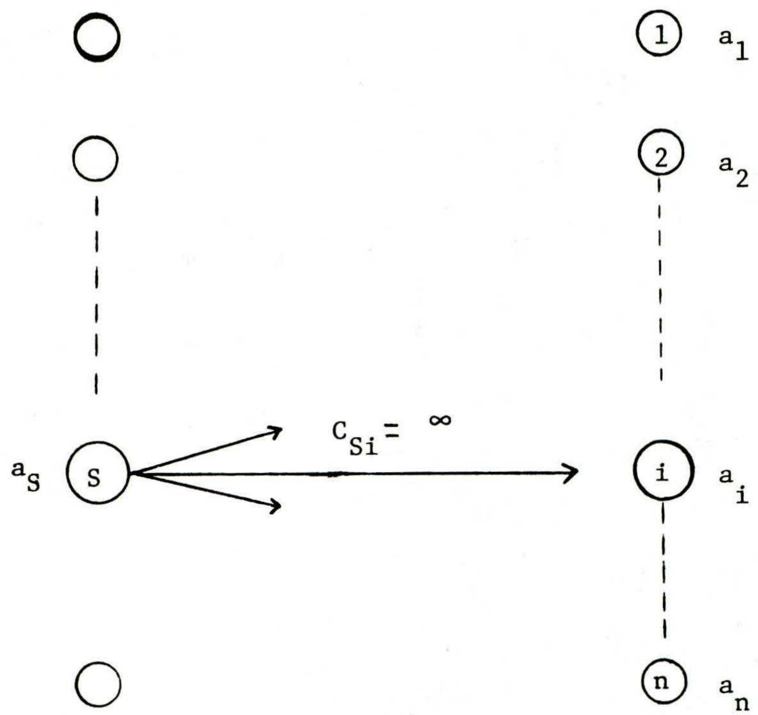


FIG. I

where an arc (y_S, x_i) exists iff $i \in S$.

Rhys, Balinski and Picard showed that the selection problem can be solved as a maximal flow problem in the network formed by the bipartite graph with infinite capacities on its arcs, a source linked to each node (y_S) by an arc of capacity a_S and a link from each node (x_i) by an arc of capacity $-a_i$. The variables which take on value 1 in an optimal solution of (P_1) correspond to the labelled vertices in the last iteration of the Ford-Fulkerson algorithm.

ILLUSTRATION 1:

$$(P)\quad \text{Max} \quad f(X) = 2x_1x_2 + 2x_1x_2x_3 + 6x_1x_2x_4 + x_2x_3x_4 + x_1x_2x_3x_4 - 2x_1 - x_2 - 5x_3 - 2x_4$$

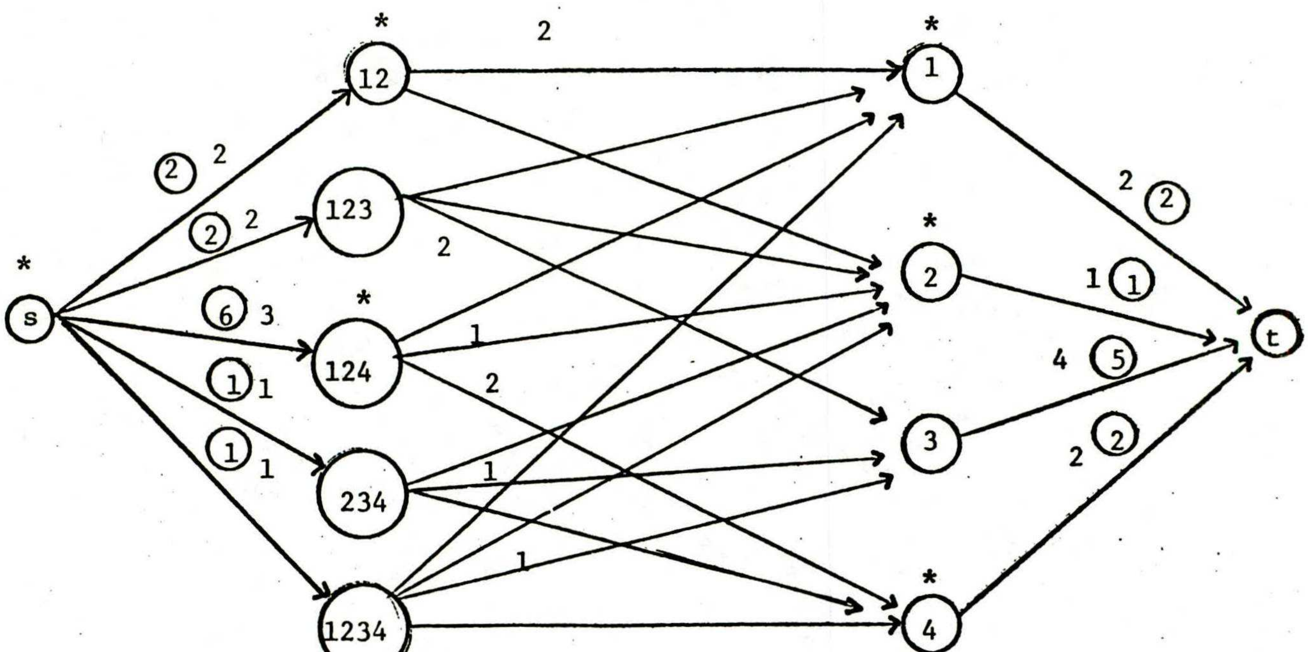
$$\text{s.t.} \quad x_j = 0,1 \quad j = 1,2,\dots,4$$

$$\text{Max} \quad f(X,Y) = 2y_{12} + 2y_{123} + 6y_{124} + y_{234} + y_{1234} - 2x_1 - x_2 - 5x_3 - 2x_4$$

$$(P_1)\quad \text{s.t.} \quad \begin{array}{lllll} y_{12} \leqslant x_1 & y_{123} \leqslant x_1 & y_{124} \leqslant x_1 & y_{234} \leqslant x_2 & y_{1234} \leqslant x_1 \\ y_{12} \leqslant x_2 & y_{123} \leqslant x_2 & y_{124} \leqslant x_2 & y_{234} \leqslant x_3 & y_{1234} \leqslant x_2 \\ & y_{123} \leqslant x_3 & y_{124} \leqslant x_4 & y_{234} \leqslant x_4 & y_{1234} \leqslant x_3 \\ & & & & y_{1234} \leqslant x_4 \end{array}$$

$x_j = 0,1 \quad j = 1,2,\ldots,4 \qquad y_{ij\ldots k} = 0,1$

The corresponding network is shown in Figure 2 with its maximal flow and the labelled nodes at the last iteration.



CAPACITIES ∞

FIGURE II

Hence an optimal solution X* to (P) is given by

$x_1^* = x_2^* = x_4^* = 1 \qquad x_j^* = 0$

2. A 0-1 FRACTIONAL PROGRAMMING PROBLEM

Consider the following 0-1 programming problem (F)

$$\text{(F)}\qquad \text{Max} \quad z(X) = \frac{\sum_{S\in A} a_S \prod_{i\in S} x_i}{\sum_{T\in B} b_T \prod_{j\in T} x_j} = \frac{f(X)}{g(X)}$$

$$\text{s.t.} \quad x_i = 0,1 \qquad i = 1,2,\ldots,n \qquad (1)$$

$$\text{and} \quad X = (x_1, x_2, \ldots, x_n) \neq 0 \qquad (2)$$

where $A, B \subseteq P(\{1,2,\ldots,n\})$

$a_S \geqslant 0$ for S such that $|S| \geqslant 2$

$b_T \leqslant 0$ for T such that $|T| \geqslant 2$

$g(X) > 0$ for any solution $X \neq 0$

and $z(X) \geqslant 0$ for any solution $X \neq 0$

For simplifying notations, define, as before

$$A' = \{S\in A \ / \ |S| \geqslant 2\}, \quad B' = \{T\in B \ / \ |T| \geqslant 2\}$$

and denote by a_i (resp. b_j) the singleton values $a_{\{i\}}$ (resp. $b_{\{j\}}$).

Then (F) can be written as:

$$\text{(F)}\qquad \text{Max} \quad z(X) = \frac{\sum_{S\in A'} a_S \prod_{i\in S} x_i + \sum_{i=1}^{n} a_i x_i}{\sum_{T\in B'} b_T \prod_{j\in T} x_j + \sum_{j=1}^{n} b_j x_j} = \frac{f(X)}{g(X)}$$

$$x_i = 0,1 \qquad i = 1,2,\ldots,n$$

By considering the unit-vector solutions $X^{(i)}$ (defined by $x_i^{(i)} = 1$ and $x_j^{(i)} = 0$ for all $j \neq i$) we observe that positivity assumptions imply $b_i > 0$ and $a_i \geqslant 0$ for all i.

LEMMA I:

Let $X^o \neq 0$ be an arbitrary 0-1 vector; then X^o is an optimal solution to (F) if and only if the maximal value of the function

$$w^o(X) = f(X) - z(X^o)\, g(X)$$

with the restriction (1) is zero.

PROOF: Obvious.

If X^o is not an optimal solution to (F) then an improved solution X^1 is found by solving the following 0-1 programming problem (F^o), which has the same structure as problem (P) defined in section 1.

$$(F^o)\quad \left[\begin{array}{l} \text{Max} \quad w^o(X) = \sum\limits_{S \in A'} a_S \prod\limits_{i \in S} x_i - z(X^o) \sum\limits_{T \in B'} b_T \prod\limits_{j \in T} x_j \\ \qquad\qquad\qquad + \sum\limits_{i=1}^{n} a_i x_i - z(X^o) \sum\limits_{j=1}^{n} b_j x_j \\ \\ \qquad\qquad\qquad x_i = 0,1 \qquad i = 1,2,\ldots,n \end{array}\right.$$

The following algorithm solves (F) by solving a (finite) sequence of problem (F^k) similar to (F^o) except that the constant term $z(X^o)$ is replaced by $z(X^k)$.

ALGORITHM:

(0) Let $X^o \neq 0$ be an arbitrary initial solution

Set $k = 0$

(1) Solve (F^k)

(2) If the maximal value of $w^k(X)$ is zero, then terminate: X^k is an optimal solution to (F).

(3) Otherwise: Let X^{k+1} be an optimal solution to (F^k); replace k by k+1 and go to (1). Note that the number of iterations is finite since $z(X^k)$ increases at each iteration and that the feasible solutions is finite $(2^n - 1)$. As it was recalled in the introduction, (F^k) is equivalent to a maximal flow problem in a network with n+2 nodes.

ILLUSTRATION 2: Consider the following problem (F):

$$
(F)\quad \left[\begin{array}{l}
\text{Max } z(X) = \dfrac{x_1 + x_2 + 2x_5 + 2x_1x_3 + 4x_1x_4 + 3x_2x_5 + x_1x_3x_5 + 2x_2x_3x_4x_5}{5x_1 + 3x_2 + 5x_3 + 6x_4 + 2x_5 - x_1x_2 - 2x_1x_2x_4 - 2x_1x_2x_3x_4x_5} \\[2em]
\text{s.t.} \quad x_j = 0,1 \qquad j = 1,2,\ldots,5 \\
\qquad\quad X \neq 0
\end{array}\right.
$$

1^{st} ITERATION: starting with $X^o = (1,1,1,1,1)$ and $z(X^o) = \frac{16}{16} = 1$

we first solve the following 0-1 programming problem

$$(F^o)\quad \begin{aligned} \text{Max } w^o(X) = {} & x_1 + x_2 + 2x_5 + 2x_1x_3 + 4x_1x_4 + 3x_2x_5 + x_1x_3x_5 \\ & + 2x_2x_3x_4x_5 - (5x_1 + 3x_2 + 5x_3 + 6x_4 + 2x_5 \\ & - x_1x_2 - 2x_1x_2x_4 - 2x_1x_2x_3x_4x_5) \\ & x_j = 0,1 \qquad j = 1,2,\ldots,5 \end{aligned}$$

or

$$(F^o)\quad \begin{aligned} \text{Max } w^o(X) = {} & -4x_1 - 2x_2 - 5x_3 - 6x_4 + x_1x_2 + 2x_1x_3 + 4x_1x_4 \\ & + 3x_2x_5 + 2x_1x_2x_4 + x_1x_3x_5 + 2x_2x_3x_4x_5 + 2x_1x_2x_3x_4x_5 \\ & x_j = 0,1 \qquad j = 1,2,\ldots,5 \end{aligned}$$

A minimum cut in the related network is characterized by $N = \{s,(2),(5),(2,5)\}$; hence an optimal solution to (F^o) is:

$$X^1 = (0,1,0,0,1) \text{ and } z(X^1) = \frac{6}{5} > z(X^o) = 1$$

2^{nd} ITERATION: we have to solve (F^1)

$$(F_1)\quad \begin{cases} \text{Max } w^1(X) = x_1 + x_2 + 2x_5 + 2x_1x_3 + 4x_1x_4 + 3x_2x_5 + x_1x_3x_5 \\ \quad + 2x_2x_3x_4x_5 - 6/5\,(5x_1 + 3x_2 + 5x_3 + 6x_4 + 2x_5 \\ \quad - x_1x_2 - 2x_1x_2x_4 - 2x_1x_2x_3x_4x_5) \\ \text{s.t.} \qquad x_j = 0,1 \qquad j = 1,2,\ldots,5 \end{cases}$$

or

$$(F_1)\quad \begin{cases} \text{Max } w^1(X) = -25x_1 - 13x_2 - 30x_3 - 36x_4 - 2x_5 + 6x_1x_2 \\ \quad + 10x_1x_3 + 20x_1x_4 + 15x_2x_5 + 12x_1x_2x_4 \\ \quad + 5x_1x_3x_5 + 10x_2x_3x_2x_5 + 12x_1x_2x_3x_4x_5 \\ \text{s.t.} \qquad x_j = 0,1 \qquad j = 1,2,\ldots,5 \end{cases}$$

A minimum cut is characterized by $N = \{s\}$; hence an optimal solution to (F^1) is $x_i = 0$ $(i = 1,2,\ldots,5)$ i.e. the maximal value of $w^1(X)$ is zero. An optimal solution to (F) is then $X^* = X^1 = (0,1,0,0,1)$ with $z(X^*) = 6/5$.

Before stating the main result, assume that the optimal solution X^k used in the algorithm is minimal among all the optimal solutions to F^k in the following sense: if $\tilde{X}$ is a nonzero binary vector,
$w^{k-1}(\tilde{X}) = w^{k-1}(X^k) \Rightarrow X^k \leqslant \tilde{X}$

The existence of such a minimal solution and the fact that it is precisely produced by the labelling method of Ford and Fulkerson are well-known results (see [5], chap. 1, thm 5.5 [11] p. 109-11).

THEOREM 2: $X^{k+1} \leqslant X^k$ for all $k \geqslant 1$

PROOF:

For $k \geqslant 1$

X^k solves Max $w^{k-1}(X) = f(X) - z(X^{k-1})\ g(X)$

s.t. $x_j = 0,1 \quad j = 1,2,\ldots,n$

X^{k+1} solves Max $w^k(X) = f(X) - z(X^k)\ g(X)$

s.t. $x_j = 0,1 \quad j = 1,2,\ldots,n$

Let $I = \{ i \mid x_i^k = 1;\ x_i^{k+1} = 0 \}$

$J = \{ i \mid x_i^k = 1;\ x_i^{k+1} = 1 \}$

$K = \{ i \mid x_i^k = 0;\ x_i^{k+1} = 1 \}$

We will prove that $K = \emptyset$

Let $\overline{X}$ be defined by $\overline{x_i} = 1$ if $i \in J$

$\overline{x_i} = 0$ otherwise

and $\hat{X}$ be defined by $\hat{x}_i = 1$ if $i \in I \cup J \cup K$

$\hat{x}_i = 0$ otherwise

Let $c = \sum_{\substack{S \in A' \\ S \subseteq J \cup K \\ S \cap K \neq \emptyset}} a_S + \sum_{i \in K} a_i$

and $d = \sum_{\substack{T \in B' \\ T \subseteq J \cup K \\ T \cap K \neq \emptyset}} b_T + \sum_{j \in K} b_j$

Since $K \neq \emptyset$, then $c \geqslant 0$

Since X^{k+1} is an optimal solution to F^k and

$\overline{X} \leqslant X^{k+1}$ then $0 < w^k (X^{k+1}) - w^k (\overline{X}) = c - z(X^k)d$ (1)

Now, we claim that:

$$c - z(X^{k-1})\, d > 0 \tag{2}$$

(i) if $d = 0$, the inequality (2) is equivalent to (1)

(ii) if $d < 0$ and $z(X^{k-1}) = 0$, the inequality (2) is proven by contradiction: if (2) does not hold, then $c = 0$; in that case, $\overline{X}$ is also an optimal solution to F^k, a contradiction with the assumption that X^k is minimal.

(iii) if $d < 0$ and $z(X^{k-1}) > 0$, the inequality (2) follows only from $c \geqslant 0$.

(iv) finally, and the most important case, if $d > 0$ the inequality (2) follows from $z(X^{k-1}) < z(X^k)$ and (1)

On the other hand, define.

$$\hat{c} = \sum_{\substack{S \in A' \\ S \subseteq I \cup K \\ S \cap I \neq \emptyset \\ S \cap K \neq \emptyset}} a_S \qquad \text{and} \qquad \hat{d} = \sum_{\substack{T \in B' \\ T \subseteq I \cup K \\ T \cap I \neq \emptyset \\ T \cap K \neq \emptyset}} b_T$$

From the sign assumptions, we have $\hat{c} \geqslant 0$ and $\hat{d} \leqslant 0$

then

$$W^{k-1}(\hat{X}) - w^{k-1}(X^k) = c + \hat{c} - z(X^{k-1})\, d - z(X^{k-1})\, \hat{d}$$
$$\geqslant c - z(X^{k-1})\, d > 0$$

a contradiction with the assumption that X^k is an optimal solution fo F^k. #

This theorem implies that the number of iterations (maximal flow problems) is in fact at most n. Using Karzanov's algorithm [12], which finds the maximal flow in a network with v vertices in at most $0(V^3)$ operations, the above algorithm solves the original problem (F) in at most $0(n^4 + na^3)$ operations, where $a = |A'|$.

3. EXTENSIONS

The approach described in section 1 could be extended to the following most general 0-1 programming problem (G).

$$(G)\quad \begin{cases} \text{Max } f(X) = \sum\limits_{S \in A} a_S \prod\limits_{i \in S} x_i \\ x_i = 0,1 \qquad i = 1,2,\ldots n \end{cases}$$

with a_S arbitrary in sign.

With the same notations as in section 1 and 2, G can be written as:

$$(G)\quad \begin{cases} \text{Max } f(X) = \sum\limits_{S \in A'} a_S \prod\limits_{i \in S} x_i + \sum\limits_{i=1}^{n} a_i x_i \\ x_i = 0,1 \qquad i = 1,2,\ldots,n \end{cases}$$

Lefting $y_S = \prod\limits_{i \in S} x_i$, we can define the problem (G'):

$$G' \quad \begin{cases} \text{Max } f(X,Y) = \sum\limits_{S \in A'} a_S\, y_S + \sum\limits_{i=1}^{n} a_i x_i \\ \qquad\qquad y_S \leqslant y_{S'} \quad \text{all } S,\ S' \in A' \text{ such that } S' \subseteq S, \\ \text{s.t.} \qquad y_S \leqslant x_i \quad \text{all } i \in S \text{ and all } S \in A' \\ \qquad\qquad y_S = 0,1 \text{ all } S \in A' \\ \qquad\qquad x_i = 0,1 \qquad i = 1,2,\ldots,n \end{cases}$$

In general, (G) and (G') are not equivalent. An optimal solution (X^*,Y^*) to G' can be infeasible to G in the sense that $x^*_i = 1$ for all $i \in S$ does not imply necessarily that $y^*_S = 1$ in G'.

However an optimal solution (X^*, Y^*) to G' gives an upper bound $f(X^*, Y^*)$ for the maximum of $f(X)$ in problem (G). These remarks could be use in a branch and bound scheme.

ILLUSTRATION 3: Consider the following 0-1 programming problem

$$(G) \quad \begin{aligned} \text{Max} \quad f(X) = {} & 2x_1 - x_2 + 3x_3 - 2x_4 - x_1x_2 - 2x_1x_4 \\ & + 2x_2x_3 - 2x_1x_2x_3 + 4x_1x_2x_4 + x_1x_2x_3x_4 \\ & x_j - 0,1 \qquad j = 1,2,\dots,4 \end{aligned}$$

G' is given by:

$$\begin{aligned} \text{Max } f(X,Y) = {} & 2x_1 - x_2 + 3x_3 - 2x_4 - y_{12} - y_{14} + 2y_{23} \\ & - 2y_{123} + y_{124} + y_{1234} \end{aligned}$$

$$(G') \quad \text{s.t.} \quad \begin{array}{lll} y_{1234} \leqslant y_{123} & y_{124} \leqslant y_{12} & y_{123} \leqslant y_{12} \\ \phantom{y_{1234}} \leqslant y_{124} & \phantom{y_{124}} \leqslant y_{14} & \phantom{y_{123}} \leqslant y_{23} \\ \phantom{y_{1234}} \leqslant y_{12} & \phantom{y_{124}} \leqslant x_1 & \phantom{y_{123}} \leqslant x_1 \\ \phantom{y_{1234}} \leqslant y_{14} & \phantom{y_{124}} \leqslant x_2 & \phantom{y_{123}} \leqslant x_2 \\ \phantom{y_{1234}} \leqslant y_{23} & \phantom{y_{124}} \leqslant x_4 & \phantom{y_{123}} \leqslant x_3 \\ \phantom{y_{1234}} \leqslant x_1 & & \\ \phantom{y_{1234}} \leqslant x_2 & & \\ \phantom{y_{1234}} \leqslant x_3 & & \\ \phantom{y_{1234}} \leqslant x_4 & & \end{array}$$

$$y_{12} \leq x_1 \quad y_{14} \leq x_1 \quad y_{23} \leq x_2$$
$$\leq x_2 \quad\quad \leq x_4 \quad\quad \leq x_3$$

$$x_i = 0,1 \qquad i = 1,2,\ldots,4 \qquad y_S = 1 \quad \text{all } S \in A'$$

An optimal solution to G', using the maximal flow algorithm, would yield

$$x^*_1 = x^*_2 = x^*_3 = x^*_4 = 1$$
$$y^*_{12} = y^*_{14} = y^*_{23} = y^*_{124} = 1$$
$$y^*_{1234} = y^*_{123} = 0$$

hence an upper bound of value 5 for the maximum of $f(X)$. In fact for our example these exists an optimal solution to (G) of value 5 which is given by $x_1 = x_3 = 1$ and $x_2 = x_4 = 0$

The above approach can be also extended to some (continuous or integer) fractional programming problems with "box constraints".

Consider the following problem:

$$(P') \quad \left[\begin{array}{l} \text{Max } z'(X) = \dfrac{Q(X)}{R(X)} \\ \\ \text{s.t.} \quad l_i \leq x_i \leq u_i \qquad \text{all } i=1, 2,\ldots,n \qquad (3) \end{array} \right.$$

where Q and R are polynomials linear in each variable, R is positive on the feasible set defined by (3), and $l_i < u_i$.

PROPOSITION 1:

There is an optimum solution to (P') which is an extreme point of the feasible set defined by (3).

PROOF:

Consider an optimum solution X^* and assume that some component x^*_i is strictly between its bounds.

By fixing all the other components at their present value, one obtains a function h of the single variable x_i:

$$f(x_i) = z(x^*_1, \ldots, x^*_{i-1}, x_i, x^*_{i+1}, \ldots, x^*_n)$$

and, since Q and R are linear in each variable:

$$f(x_i) = \frac{ax_i + b}{cx_i + d}$$

Since the denominator $cx_i + d$ does not vanish in the interval $[l_i, u_i]$, this function $f(x_i)$ either assumes its unique maximum at one end of the interval, or is constant on it. The first alternative implies a contradiction, and, in the second case, one can arbitrarily set $x^*_i = u_i$ or l_i and consider another component x^*_j strictly between its bounds, and so on until either a contradiction or an optimum extreme point is attained. Consequently, the problem (P') is reduced to a problem of the type discussed above, by the variable change

$$y_i = \frac{x_i - l_1}{u_i - l_i} \qquad \text{for all } i = 1, 2, \ldots, n$$

In the case where X (and the bounds L and U) is restrained to be integer, this transformation is much better than the introduction of the binary expansion of the components $x_i - l_i$.

4. APPLICATIONS

4.1 The Density of a Graph

Consider an undirected graph $G = (N,E)$ with n_G nodes and e_G edges. Hereafter, the notation $H \subset G$ will mean that H is a partial subgraph of G.

The density of the graph G, denoted by d(G), is defined as the maximum, over all partial subgraphs of G, of the ratio of the number of edges to the number of nodes, i.e.:

$$d(G) = \max_{H \subset G} \frac{e_h}{n_H}$$

Clearly, we can restrict the maximization to subgraphs generated by subsets of N. Associating to each node i of N a 0-1 variable x_i where $x_i = 1$ means that the node i belongs to H, then the problem of finding d(G) and the corresponding subgraph H can be formulated as;

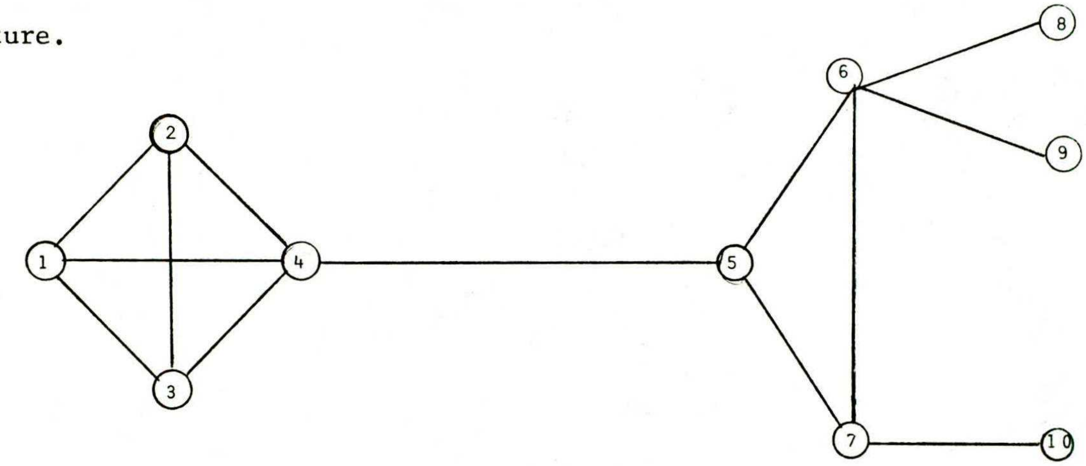
$$d(G) = \frac{1}{2} \max \frac{\sum_{i=1}^{n_G} \sum_{j=1}^{n_G} a_{ij} x_i x_j}{\sum_{j=1}^{n_G} x_j}$$

$$x_j = 0,1 \qquad j = 1,2,\ldots n_G$$

where a_{ij} are the elements of the node-to-node adjacency matrix of G.

This last problem is a particular case of problem (F) of section 2.

ILLUSTRATION: Consider the graph $G = (N,E)$ pictured in the following picture.



Starting with $z(X^o) = \frac{e_G}{n_G} = \frac{13}{10} = 1.3$, we find the subgraph H_1 generated by $\{1,2,3,4,5,6,7\}$. Another iteration with $z(X^1) = \frac{e_{H_1}}{n_{H_1}} = \frac{10}{7}$ yields the subgraph H_2 generated by $\{1,2,3,4\}$. No improvement can be done with $z(X^2) = \frac{e_{H_2}}{n_{H_2}} = \frac{6}{4} = 1.5$; hence $d(G) = 1.5$ and the corresponding subgraph is the subgraph H_2.

This algorithm has been applied on a set of ten 20-nodes graphs and nineteen 50-nodes graphs. In 90% of the cases, only one iteration was required and the average CPU solution time (with one IBM 360-75) for the nineteen 50-nodes graphs was 0.4 second (for more detail see [22]).

4.2 Pseudo-arboricity and pseudo-forest decomposition of a graph

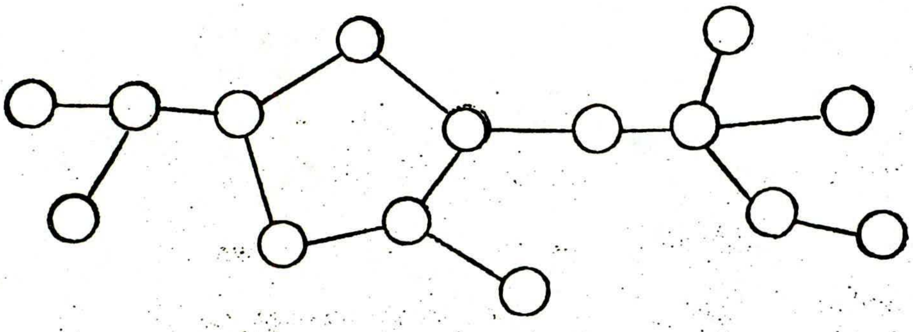
In this section, we will show that the density of a graph gives the minimum number of edge disjoint pseudo-forests into which G can be decomposed; furthermore the flow solution of d(G) will provide this decomposition. Before proving these results, we need some definitions.

Pseudo-Tree: A Pseudo-Tree is a tree which contains exactly one cycle (or equivalently a connected graph with n nodes and n edges).

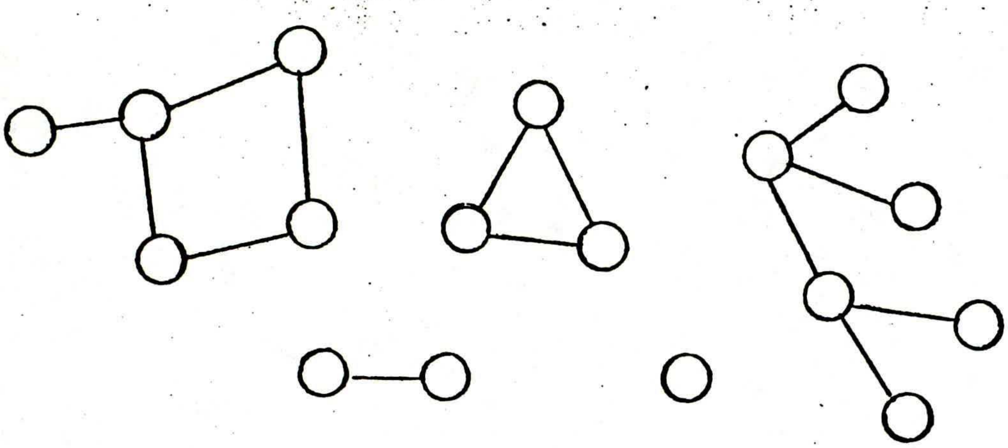
Pseudo-Forest: A Pseudo-Forest is a graph each connected component of which is a pseudo-tree or a tree.

Remark: If $G = (X,U)$ is a pseudo-forest, then $|U| \geqslant |X|$ and we have the equality if each component is a pseudo-tree.

<u>ILLUSTRATION 3</u>: (See Figure 4)



<u>PSEUDO-TREE</u>



<u>PSEUDO-FOREST</u>

<u>FIGURE 4</u>

Pseudo-Arboricity of a Graph: The Pseudo-Arboricity $P(G)$ of a graph G is defined as the minimum number of edge-disjoint pseudo-forests into which G can be decomposed.

ILLUSTRATION 4: K_5 (complete graph with 5 nodes) can be decomposed into 2 pseudo-forests, i.e. $P(K_5) = 2$, see Figure 5.

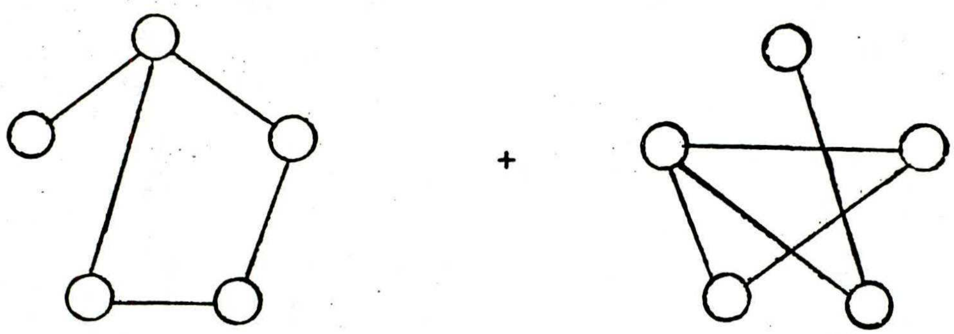


FIGURE 5

Remark: while K_5 admits several pseudo-forest decompositions (e.g in two spanning cycles), picking an arbitrary pseudo-forest does not necessarily produce such a decomposition: for instance, there is no decomposition of K_5 including a pseudo-forest in which a node has degree 4.

The problem of finding a minimum pseudo-forest decomposition of a given graph can be solved using the theorems 9-6 and 9-7 of [9] in the case of complete graphs. The following theorem solves this problem in the general case:

THEOREM 2: Let $G = (N,E)$ be a graph of density $d(G)$, then $P(G) = \lceil d(G) \rceil$, where $\lceil d(G) \rceil$ is the smallest integer greater or equal to the density $d(G)$ of the graph G.

PROOF: Let $G'=(N',E')$ be the subgraph which yields the density of G;i.e. $d(G) = \frac{m'}{n'}$, where $n'= |N'|$ and $m'=|E'|$.

For simplicifation, we will denote $\lceil d(G) \rceil$ by k.

We have $P(G) \geqslant k$.

If not, we could decompose G, so G', into $k'(k'<k)$ pseudo-forests $G'=(N',E'_1)$, $G'_2=(N',E'_2),\ldots,G'_{k'}=(N', E'_{k'})$ with $|E'_1|+|E'_2|+\ldots+|E'_{k'}|=m'$

But $|E'_i| \leqslant n'$ for $i = 1, 2, \ldots, k$; the last inequalities imply that

$$|E'_1|+|E'_2|+\ldots+|E'_{k'}| \leqslant n'k'$$

i.e. $m' \leqslant n'k'$

or $\frac{m'}{n'} \leqslant k'$

Hence $\left\lceil \frac{m'}{n'} \right\rceil \leqslant k' < k$ contradicting the fact that $\left\lceil \frac{m'}{n'} \right\rceil = k$

So the theorem will be proved if we can show that it is always possible to decompose G into k pseudo-forests. This can be proved in the following way.

By definition of d(G), we have:

$$\min_{x_i = 0,1} \left(\sum_{i=1}^{n} \sum_{j=1}^{n} -a_{ij}x_i x_j + d(G) \sum_{i=1}^{n} x_i \right) = 0$$

so

$$\min_{x_i = 0,1} \left(\sum_{i=1}^{n} \sum_{j=1}^{n} -a_{ij}x_i x_j + \lceil d(G) \rceil \sum_{i=1}^{n} x_i \right) = 0$$

i.e. the maximal flow value corresponding to the following minimum cut problem (Q) :

$$\text{(Q)} \qquad \min z = \sum_{i=1}^{n} \sum_{j=1}^{n} - a_{ij}x_i x_j + P(G) \sum_{i=1}^{n} x_i$$

$$x_i = 0,1 \quad \text{for } i=1, 2, \ldots, n$$

is equal to m; in other words, all arc-sources of the related network are saturated.

Let us now define, from the maximal flow in this network, a bipartite graph $B=(S \cup T, A)$ by deleting

-all the arc-sources,

-all the arc-sinks,

-the intermediate arcs (x_{ij}, x_i) or $(x_{ij} x_j)$ which have a flow of value 0 and the nodes x_i which have no inflow.

The graph $B=(S \cup T, A)$ has the following characteristics:

$$|S| = m \quad ; \ |T| \leq n \quad \text{and } |A| = m$$

where n and m are the number of nodes and the number of edges, respectively, of the original graph G.

The degree of each node of S is exactly one and the degree of each node of T is at least one and no more than k; furthermore there exists at least one node of T with degree equal to k; if not, d(G) would be $<$k.

By the theorem of König-Hall, its is obvious that we can decompose B into exactly k matchings $M_1(S_1 \cup T_1, A_1), M_2(S_2 \cup T_2, A_2), \ldots, M_k(S_k \cup T_k, A_k)$.

To the matching $M_i(S_i \cup T_i, A_i)$ $(i=1,2,\ldots,k)$ corresponds in G a subgraph G_i defined by the nodes of T_i and the edges related to the nodes of S_i; furthermore G_i is a pseudo-forest since each of its connected component has a number of edges which is not greater that its number of nodes.

ILLUSTRATION 5: Consider the graph G = (N,E) of Figure 6.

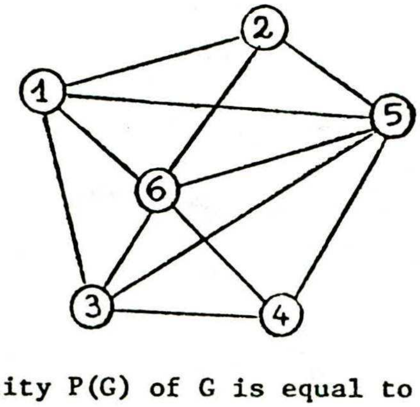


FIGURE 6

The pseudo-arboricity P(G) of G is equal to 2 and is given by G itself.

A maximal flow corresponding to the minimum cut problem which yields the pseudo-arboricity i.e.:

$$\text{Min} \quad z(X) = \sum_{i=1}^{6} \sum_{j=1}^{6} -a_{ij}x_i x_j + 2\sum_{i=1}^{6} x_i$$
$$x_j = 0,1 \qquad \text{for } j = 1, 2, \ldots, 6$$

is given in the following tableau.

	α12	α13	α15	α16	α25	α26	α34	α35	α36	α45	α46	α56
1	1	1			X	X	X	X	X	X	X	X
2		X	X	X	1	1	X	X	X	X	X	X
3	X		X	X	X	X	1	1		X	X	X
4	X	X	X	X	X	X		X	X	1	1	X
5	X	X	1	X		X	X		X		X	1
6	X	X	X	1	X		X	X	1	X		

The corresponding bipartite graph B= (S T,A) is given in Fig. 7

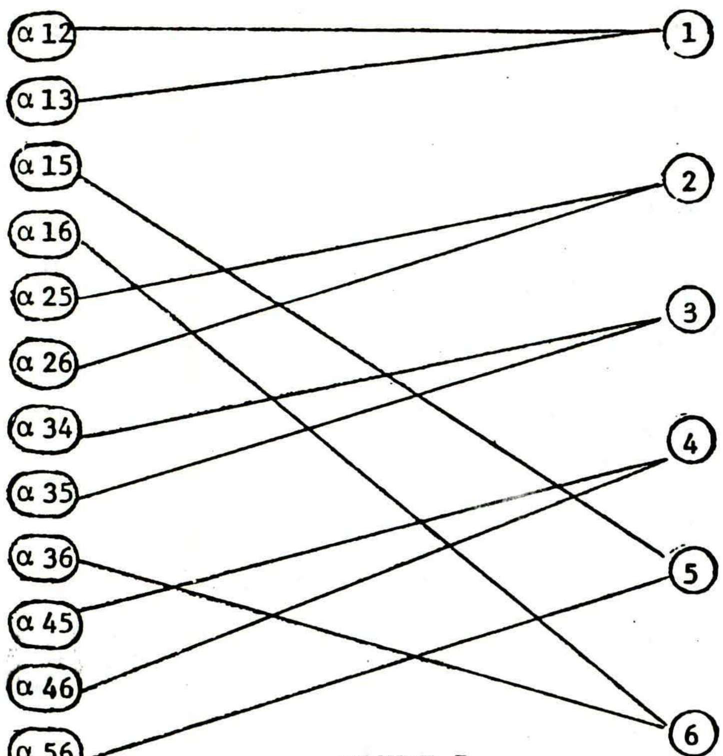


FIGURE 7

Two matchings in B lead, for example, to the decomposition of G into two pseudo-forests, given in Figure 8.

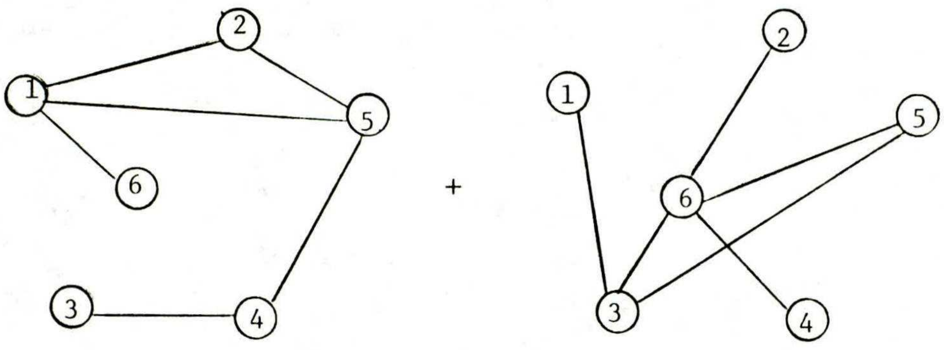


FIGURE 8

It is worthwile to note that in this case, the components of each pseudo-forest are pseudo-tree; this is always the case if d(G) is given by G itself and is an integer number.

4.3 The Arboricity of a Graph

The arboricity $\Gamma(G)$ of a graph $G=(N,E)$ is defined as the minimum number of edge disjoint forests into which G can be decomposed; it is also the minimum number of colours necessary to colour the edges of G so that no cycle has all its edges with the same colours. The arboricity of a graph finds applications in Matroid Theory.

Nash-Williams [14] has proved the following result.

Theorem of Nash-Williams: The arboricity $\Gamma(G)$ of a graph $G=(N,E)$ is given by

$$\Gamma(G) = \text{Max} \left\{ \left\lceil \frac{e_G(A,A)}{|A| - 1} \right\rceil : A \subset N, |A| > 1 \right\}$$

where $A \subset N$, $e_G(A,A)$ is the number of edges having their extremities in A and $\lceil x \rceil$ denotes the smallest integer greater than or equal to x.

The search for $\Gamma(G)$ can be defined as:

$$\text{find } \Gamma(G) = \lceil \gamma(G) \rceil$$

$$\text{where } \gamma(G) = \underset{x_j=0,1}{\text{Max}} \frac{\sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} x_i x_j}{\sum_{j=1}^{n} x_j - 1} \quad \text{s.t.} \sum_{j=1}^{n} x_j > 1.$$

The arboricity $\Gamma(G)$ could be found by the Matroid Partitioning Algorithms of Edmonds [13]. In this section we will show that using network flow this number can be determined in at most $O(n^4 \log n)$ iterations.

THEOREM 3: **The following relation exists between the density $d(G)$ of G and $\gamma(G)$: $d(G) < \gamma(G) < d(G) + 1$**

PROOF: **Let $G_1 = (N_1, E_1)$ and $G_2 = (N_2, E_2)$ be two subgraphs of G which yield $d(G)$ and $\gamma(G)$ respectively. The first inequality is obvious.**

Letting $n_1 = |N_1|$, $m_1 = |E_1|$, $n_2 = |N_2|$ and $m_2 = |E_2|$

it comes: $d(G) = \frac{m_1}{n_1} \geqslant \frac{m_2}{n_2}$

and

$$\frac{m_1}{n_1 - 1} \leqslant \frac{m_2}{n_2 - 1} = \gamma(G)$$

Let us assume that $\gamma(G) \geqslant d(G) + 1$

then $$\frac{m_2}{n_2 - 1} \geqslant \frac{m_1}{n_1} + 1 \geqslant \frac{m_2}{n_2} + 1$$

Hence $$\frac{m_2}{n_2 - 1} \geqslant \frac{m_2}{n_2} + 1$$

or $m_2 \geqslant n_2^2 - n_2$ **, contradicting the fact that m_2 cannot be greater than $\frac{n_2^2 - n_2}{2}$** $\therefore$

Corollary 3-1. **We have : $P(G) \leqslant \Gamma(G) \leqslant P(G) + 1$**

Corollary 3-2. If $d(G)$ is an integer number then:

$$\Gamma(G) = P(G) + 1 = d(G) + 1$$

Corollary 3-3. Let $d(G) = \frac{m_1}{n_1}$, where m_1 and n_1 represent the number of edges and the number of nodes, respectively, of the subgraph $G_1 = (N_1, E_1)$ which yields $d(G)$; if $\left\lceil \frac{m_1}{n_1 - 1} \right\rceil > \left\lceil \frac{m_1}{n_1} \right\rceil$, then

$$\Gamma(G) = \lceil d(G) \rceil + 1 = P(G) + 1$$

If d(G) does not satisfy the assumption of one of the corollaries 3-2 or 3-3, then we have to solve the problem:

$$\gamma(G) = \max_{x_j = 0,1} \frac{\sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} x_i x_j}{\sum_{j=1}^{n} x_j - 1} \quad \text{s.t.} \quad \sum_{j=1}^{n} x_j > 1.$$

Because of the presence of the number -1 in the denominator of the function to be maximized, the problem is a little more complicated to solve than the density problem; A Branch and Bound approach is described in [22]; this algorithm solves at most n maximal flow problems and hence necessitates at most $O(n^4 \log n)$ operations.

4.4 The Maximum Clique and Vertex Packing Problems

From the density $d(G)$ of a graph may be deduced an upper bound on the maximum size $c(G)$ of a clique contained in G and, consequently, on the maximum size of a stable (vertex packing) in the complementary graph $\bar{G}$. Indeed, recall that the complementary graph $\bar{G}$ of G is a graph with the same node set as G, but such that two nodes are adjacent in $\bar{G}$ if and only if they are not adjacent in G. Clearly, to a clique in G corresponds a stable in $\bar{G}$ and conversely, hence $c(G)$ equals the stability number $\alpha(\bar{G})$ of its complement. So, the density $d(G)$ may also be used in a branch-and-bound algorithm for solving the vertex packing problem on $\bar{G}$ (i.e find $\alpha(\bar{G})$ and a corresponding maximum stable set in $\bar{G}$).

The density (K_q) of a clique K_q is $\frac{1}{2}$ (q-1). Hence, if $c(G)$ denotes the maximum size of a clique in G, we must have

$$d(G) \geqslant \frac{1}{2}\,(c(G)-1)$$

i.e. $$c(G) \leqslant 2d\,(G)+1$$

and, denoting by $[\,x\,]$ the integer part of x,

$$c(G) \leqslant [\,2d(G)] + 1$$

This bound may be used in a branch-and-bound algorithm for finding $c(G)$.

4.5 Generalisation of the Selection Problem

The Selection problem [23] can be stated in terms of "activities" and "facilities", associated with benefits and costs respectively, and available only on a (0,1) basis. Any activity depends on the existence of a particular set of facilities necessary to that activity. The problem is the selection of activities and facilities to maximise the the excess of benefit over cost A 0-1 formulation of this problem is:

$$\text{Max} \quad z(X,Y) = \sum_{j=1}^{n} p_j x_j - \sum_{i=1}^{n} c_i y_i$$

$$x_j \leqslant y_i \quad \text{if activity j implies facility i}$$

$$x_j = 0,1 \quad i = 1,2,\dots,n$$

$$y_i = 0,1 \quad j = 1,2,\dots,m$$

where p_j and c_i are the profit associated to activity j and the cost associated to facility i respectively.

Instead of maximizing the excess of benefit over costs we may also maximize the ratio benefit over cost. The resulting problem is:

$$\text{Max} \quad z(X,Y)x = \frac{\sum_{j=1}^{n} p_j x_j}{\sum_{i=1}^{m} c_i y_i}$$

$$x_j \leqslant y_i \quad \text{for } (j,i) \in A$$

$$x_j = 0,1 \qquad j = 1,2,\ldots,n$$

$$y_i = 0,1 \qquad i = 1,2\ldots,m$$

With the results of sections 1 and 2 this problem can be solved as a sequence of at most n+m selection problems.

A generalization could be the following problem:

$$\text{Max } z(X,Y) = \frac{\sum_{j=1}^{n} p_j x_j + \sum_{S\in A} p_S \prod_{j\in S} x_j}{\sum_{i=1}^{m} c_i y_i - \sum_{T\in B} c_t \prod_{i\in T} y_i}$$

$$x_j \leqslant y_i \quad \text{if activity j implies facility i}$$

$$x_j = 0,1 \qquad j = 1,2,\ldots,n$$

Where p_S represents an additional profit if the subset S of activities is selected and c_T is a reduction on cost if the subset T of facilities is selected.

This problem also can be solved as a sequence of (at most n+m) selection problems.

CONCLUSION

In this study, we have shown how a network technique, namely the identification of a minimum cut through solution of a maximum flow problem, can be useful for solving a nonlinear 0-1 programming problem. We have discussed several graph-theoretic applications to this approach, including a polynomial algorithm for finding the arboricity of a graph.

REFERENCES

[1] BALAS, E. and SAMUELSSON, H. "Finding a minimum node cover in an arbitrary graph", Management Science Research Rept. 325, Graduate School of Business Administration. Carnegie-Mellon University, Pittsburgh, Pa. (November 1973.)

[2] BALINSKI, M.L. "On a Selection Problem", Management Science, 17 (1970), 230-231.

[3] BEINEKE, L.W. "Decomposition of Complete Graphs into Forests", Magyar Tud. Akad, Mat. Kutato Int. Kozl. 9 (1964), 589-594.

[4] DINIC, E.A. "Algorithm for Solution of a Problem of Maximum Flow in a Network with Power Estimation", Soviet Math. Doklady 11 (1970), 1277-1280.

[5] FORD, L.R. and FULKERSON, D.R. "Flows in Networks", Princeton University Press, Princeton, N.J. (1962).

[6] HAMMER (Ivanescu), P.L. "Some Network Flow Problems Solved with Pseudo-Boolean Programming", Operations Research, 13 (1965) 388-389.

[7] HAMMER, P.L. and RUDEANU, S. "Boolean Methods in Operations Research", Springer Verlag, New York (1968).

[8] HANSEN, P. "Programmes mathématiques en variables 0-1", Thèse, Université Libre de Bruxelles, Belgique (1974).

[9] HARARY, F. "Graph Theory", Addison-Wesley, Reading, Ms (1969).

[10] HOUCK, D. and VEMUGANTI, R.R., "An Algorithm for the Vertex Packing Problem", Mathematics Research Report 7507, University of Maryland Baltimore County, Baltimore, Md, (August 1975).

[11] HU, T.C., "Integer Programming and Network Flows", Addison-Wesley, Reading, Mass., (1969).

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

[13] LAWLER, E.L., "Combinatorial Optimization: Networks and Matroids", Holt, Rinehart and Winston, New York, N.Y., (1976).

[14] NASH-WILLIAMS, C.St.J.A., "Edge-Disjoint Spanning Trees of Finite Graphs", J. London. Math. Soc., 36 (1961), 445-450.

[15] NEMHAUSER, G.L. and TROTTER, L.E. "Vertex Packings: Structural Properties and Algorithms", Math. Programming, 8 (1975), 232-248.

[16] PICARD, J.-C., "Maximal Closure of a Graph and Application to Combinatorial Problems", Management Science, 22 (1976), 1268-1272.

[17] PICARD, J.-C. and QUEYRANNE, M., "Vertex-Packings: (VLP)-Reductions through Alternate Labeling", Techn. Rep. EP75-R-47, Ecole Polytechnique de Montréal, Canada, (September 1975) (submitted for publication).

[18] PICARD, J.-C. and QUEYRANNE, M., "Simple Validation of Maximal Closure of a Graph", Techn. Rep. EP75-R-60, Ecole Polytechnique de Montréal, Canada, (November 1975).

[19] PICARD, J.-C. and QUEYRANNE, M., "On the Integer-Valued Variables in the Linear Vertex Packing Problem", Mathematical Programming 12, (1977), 97-101.

[20] PICARD, J.-C. and RATLIFF, H.D., "A Graph-Theoretic Equivalence for Integer Programs", Operations Research 21, (1973), 261-269.

[21] PICARD, J.-C. and RATLIFF, H.D., "Minimum Cuts and Related Problems", Networks 5, (1975), 357-370.

[22] PICARD, J.-C. and QUEYRANNE, M., "Networks, Graphs and Some Non-Linear 0-1 Programming Problems"., Techn. Rep. EP77-R-32, Ecole Polytechnique de Montréal, Canada , (1977).

[23] RHYS, J.M.W., "A Selection Problem of Shared Fixed Costs and Network Flows", Management Science 17, (1970), 200-207.