Titre: Title:	Selected applications of maximum flows and minimum cuts in networks
Auteurs: Authors:	Jean-Claude Picard, & Maurice Queyranne
Date:	1979
Type:	Rapport / Report
Référence: Citation:	Picard, J.-C., & Queyranne, M. (1979). Selected applications of maximum flows and minimum cuts in networks. (Rapport technique n° EP-R-79-35). https://publications.polymtl.ca/6192/

Institution:	École Polytechnique de Montréal
Numéro de rapport: Report number:	EP-R-79-35
URL officiel: Official URL:	
Mention légale: Legal notice:	

Génie Industriel

SELECTED APPLICATIONS OF MAXIMUM FLOWS AND MINIMUM CUTS IN NETWORKS

BY: PICARD, JEAN-CLAUDE AND QUEYRANNE, MAURICE

ECOLE POLYTECHNIQUE DE MONTREAL
DEPARTEMENT DE GENIE INDUSTRIEL

RAPPORT TECHNIQUE NO.: EP79-R-35

NOVEMBRE 1979.

Ecole Polytechnique de Montréal

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

SELECTED APPLICATIONS OF MAXIMUM FLOWS AND MINIMUM CUTS IN NETWORKS

BY

PICARD, JEAN-CLAUDE

AND

QUEYRANNE, MAURICE

ECOLE POLYTECHNIQUE DE MONTREAL
DEPARTEMENT DE GENIE INDUSTRIEL

RAPPORT TECHNIQUE NO.: EP-79-R-35

NOVEMBRE 1979.

SELECTED APPLICATIONS OF MAXIMUM FLOWS AND MINIMUM CUTS IN NETWORKS

BY

J.-C. PICARD AND M. QUEYRANNE
ECOLE POLYTECHNIQUE DE MONTREAL

ABSTRACT

This paper provides a survey and a synthesis of a number of applications of maximum flows and minimum cuts in networks, emphasizing applications to integer (linear and nonlinear) programming, sequencing and scheduling theory and location theory. In particular, a binary quadratic programming formulation of the minimum cut problem has allowed the solution of a number of problems in investment selection, mining engineering, graph theory and location theory as minimum cut problems or as sequences of minimum cut problems. The paper concludes with a discussion of more difficult minimum cut problems, resulting from negative capacities or additional constraints, and of potential applications of such models.

NOTE: This is a working paper, being submitted for publication in a Journal. The authors welcome any comments and criticisms.

The maximum flow problem, studied by L.R. Ford and D.R. Fulkerson around 1955, payed a central role in the development of the network flow theory. This is well illustrated by the contents of their classical book titled Flows in Networks (1962): The first half of this book is devoted to the maximum flow problem, feasibility theorems and combinatorial applications. Moreover the second half is devoted to minimum cost flow problems, for which they describe an algorithm which can be viewed as a process of solving a sequence of maximum flow problems[33 p.94], and to multi-terminal maximum flows, another generalization of the maximum flow theory.

The maximum flow theory has attracted a renewed interest in the 70's, mostly for two reasons. First, it was possible to devise maximum flow algorithms,involving many new and fascinating ideas, and much more satisfying, form the point of view of theoretical efficiency, than existing minimum cost flow or more general network flow algorithms. Second, a number of new and interesting applications of maximum flows and, in particular, of minimum cuts have also been explored during that period.

The purpose of this paper is to provide a survey and a synthesis of a number of applications of maximum flows and minimum cuts, emphasizing applications to integer (linear and nonlinear) programming, to sequencing and scheduling theory and to location theory. Of course, this selection reflects the interests of the authors, and this paper also outlines other applications to investment selection, mining engineering, graph theory and other areas of combinatorial optimization.

The paper is organized as follows. The first section contains a brief review of basic results and classical extensions of maximum flow theory. The next three sections deal mostly with maximum flows. Section 2 discusses very shortly on finding maximum flows. Section 3 is a review of direct applications of maximum flow theory, including one application to pre-emptive scheduling. Section 4 indicates in which directions the maximum flow theory has been extended. In these first four sections we have gone into some detail only for topics which we felt were not covered in the first two chapters of Ford and Fulkerson's book[33].

The remainder of this paper deals with minimum cuts and applications, in somewhat greater detail. Section 5 reviews methods for finding minimum cuts. The direct applications of minimum cuts, described in Section 6, are those which arise most naturally from the concept of a cut in a network. Section 7 introduces a binary quadratic programming formulation of minimum cuts, due to Hammer (Ivanescu)[47] and Picard and Ratliff[95]. This formulation leads to a number of interesting applications of minimum cuts. For all these applications it is possible to directly justify the minimum cut model, but the approach by the binary quadratic programming formulation presents two advantages: it is more direct, since it avoids complicated or boresome demonstrations, and more fruitful, since it constructs the minimum cut network directly from the problem at hand. Section 8 describes a number of problems which can be solved as a sequence of minimum cut problems, extending the results of Section 7. Finally Section 9 explores some limits of this approach by describing two classes of more difficult minimum cut problems: those involving negative capacities and those involving additional constraints.

We provide a number of references at the end of the paper. In order to keep this list to a manageable size we usually indicate, for every topic, only a few essential references and some most recent papers, from which a more complete set of references can usually be retraced. A number of excellent surveys, including surveys on integer (linear and non-linear) programming, sequencing and scheduling theory, location theory and other areas of discrete optimization are found in the two volumes of [48]. Since the book by Fork and Fulkerson [33], a number of textbooks have been partly or totally devoted to network flows, including Berge and Ghouila-Houri [6] in 1962; Busacker and Saaty [12] in 1964; Iri [55] in 1969; Berge [5], Elmaghraby [27] and Hu [54] in 1970; Frank and Frisch [34] in 1971; Christofides [15] in 1975; Lawler [72] in 1976; Bazaraa and Jarvis [4] in 1977 and Minieka [81] in 1978. We also mention the collection of papers edited by Boesch [8].

* * *
* *

1. BASIC RESULTS

Consider a finite directed network $N = (V,A,c)$, where V is the set of vertices, A is the set of arcs, and c is a positive real-valued function defined on A. For every arc a, we define its head (or destination) $h(a) \in V$, its tail (or origin) $t(a) \in V$ and its capacity $c(a)$. Given two vertices s and t called the source and the sink, we call an (s,t) - flow (or more simply a flow) any real-valued function f defined on A satisfying the (Kirchhoff) conservation law (or balance equations)

$$\sum_{a:h(a) = i} f(a) = \sum_{a:t(a) = i} f(a) \qquad (1\text{-}1)$$

for all vertices $i \in V$ distinct from s and t. By summing all these equations, we have

$$\sum_{a:t(a)=s} f(a) - \sum_{a:h(a)=s} f(a) = \sum_{a:h(a)=t} f(a) - \sum_{a:t(a)=t} f(a) = v(f) \qquad (1\text{-}2)$$

and we call this quantity $v(f)$ the value of the flow f. If in addition

$$0 \leq f(a) \leq c(a) \qquad (1\text{-}3)$$

for all arcs a, we call f a feasible flow. The maximum flow problem is the problem of finding a feasible flow of maximum value. This problem may be stated as a linear programming problem, with $|V|-2$ constraints (1-1) and $|A|$ nonnegative variables $f(a)$ with upper bounds.

Given two subsets S,T of V, and any real function g defined on A, we denote by (S,T) the set of all arcs with tail in S and head in T:

$$(S,T) = \{a \in A: t(a) \in S \text{ and } h(a) \in T\}, \tag{1-4}$$

and by g(S,T) the sum of the values g(a) for all $a \in (S,T)$:

$$g(S,T) = \sum_{a \in (S,T)} g(a) \tag{1-5}$$

A set (S,T) is a cut (separating s and t) if $T = \bar{S}$ (i.e. V-S, the complement of S in V) $s \in S$ and $t \in T$, and we call the quantity c(S,T) (as defined by (1-5)) the capacity of the cut (S,T). The minimum cut problem is the problem of finding a cut (separating s and t) with minimum capacity.

It turns out[20] that the dual of the linear programming formulation of the maximum flow problem can be interpreted as a formulation of the minimum cut problem. Thus the fundamental Max-flow min-cut theorem[31]:

> *For any network the maximum flow value from s to t is equal to the minimal cut capacity of all cuts separating s and t.*

The most elegant proof due to Ford and Fulkerson is constructive and makes use of the labelling rules *(see* [33]*)*, which are a translation of the complementary slackness conditions:

$$f^*(a) > 0 \Rightarrow (h(a) \in S^* \Rightarrow t(a) \in S^*) \tag{1-6}$$

$$f^*(a) < c(a) \Rightarrow (t(a) \in S^* \Rightarrow h(a) \in S^*) \tag{1-7}$$

for any maximum flow f^* and any minimum cut $(S^*,\bar{S}^*)$. The Labelling Method uses these rules to derive a minimum cut, if possible, from a given flow, or to improve this flow. While some irrational capacities may cause the labelling method to make an infinite number of iterations (and possibly to converge toward a flow which is not maximum), the process terminates in at most $v(f^*)$ iterations when the capacities (and also the initial flow-usually the zero flow) are integer. A consequence is the important Integrity Theorem:

> *If the capacity function c is integer, there exists a maximum flow f* that is also integer.*

Classical extension of these results are also found in the same reference. Vertex capacities are treated by splitting every capacitated vertex and introducing a capacitated arc connecting its two parts. Multiple sources and multiple sinks are dealt with by extending the network by addition of a supersource and a supersink, respectively. Lower bounds (in other words "lower capacities") on arc flows, i.e. restrictions

$$\ell(a) \leq f(a) \leq u(a) \tag{1-8}$$

instead of (1-3) are handled in two steps. First, a feasible flow, if any, can be found by solving a maximum flow problem in a related network, an

equivalent to the classical variable change used to accomodate lower bounds in lenear programming; this point will be considered in Section 3. When a feasible flow is given, the labelling rules can be applied with the following modification to (1-6)

$$f^*(a) > \ell(a) \Rightarrow (h(a) \in S^* \Rightarrow t(a) \in S^*) \tag{1-9}$$

(and u(a) instead of c(a) in (1-7)).

The max.-flow min.-cut theorem applies, by replacing the definition of the capacity $c(S,\overline{S})$ of a cut by the following

$$c(S,\overline{S}) = u(S,\overline{S}) - \ell(\overline{S},S) \tag{1-10}$$

The above results extend to undirected and mixed networks. The concept of flow in an edge $e = \{x,y\}$ is interpreted to mean that

$$\begin{aligned} f(x,y) &\leq c(e) \\ f(y,x) &\leq c(e) \\ \text{and } f(x,y).f(y,x) &= 0 \end{aligned} \tag{1-11}$$

(see (33 p.23)), i.e. every edge may be replaced with a pair of oppositely directed arcs, each having capacity equal to the capacity of the edge. Another (equivalent) device is to arbitrarily direct the edge, say from x to y, and to impose

$$-c(e) \leq f(x,y) \leq c(e) \tag{1-12}$$

instead of (1-11); this transformation avoids doubling the number of arcs and taking care of cancelling oppositely directed flows. When lower bounds

are introduced in an undirected network, and the flow in an edge is unidirectional but no direction is a priori specified, Ford and Fulkerson consider[33,p.51] the problem of finding a feasible flow. They state that the problem is not equivalent to the directed problem. Indeed, we can prove that this feasibility problem in an undirected network is NP-complete, by reduction from the following NP-complete problem:

PARTITION[62]: given a positive vector $(a_1, a_2, \ldots, a_n)$, does there exist a subset I of the index set $J = \{1, 2, \ldots, n\}$ such that $\sum_{i \in I} a_i = \sum_{i \notin I} a_i = \frac{1}{2} \sum_{i \in J} a_i$? (1-13)

The reduction goes as follows: construct a network with n + 3 vertices, with v_0 as source and v_{n+2} as sink; for every index $i \in J$ define two edges $\{v_0, v_i\}$ and $\{v_i, v_{n+1}\}$, both with upper and lower capacity equal to a_i; add an edge $\{v_{n+1}, v_{n+2}\}$ with upper and lower capacity equal to zero. Verifying that the existence of a feasible flow is equivalent to the existence of a partition I satisfying (1-13) is left to the reader.

* * *
* *

2. FINDING MAXIMUM FLOWS

The simplest way to solve a maximum flow problem is to state it as a linear programming problem and to solve it with any linear programming algorithm. A minimum cost flow algorithm, when available, is by far more efficient. There are two slightly distinct ways to state a max. flow problem in min. cost flow format. In both methods, the network is augmented with zero arc costs and zero vertex demands. In the first method a "return" arc (t,s) is added, with cost equal to minus one, zero lower bound and large capacity (the capacity of any cut will do). The second method avoids the use of negative costs (which may be troublesome for certain network codes): the source s has a large supply, the the sink has an equal demand; a "bypass" arc (t,s) is added with cost plus one, zero lower bound and large capacity. For both methods a feasible flow (corresponding to the zero flow in the max. flow network) is available when there are no lower bound restrictions.

There is also a number of specific max. flow algorithms. We resist to the temptation of going into greater detail and refer the reader to recent books, e.g. (72),(17) and to (75) for a review. Other recent references include (38)(57)(79)(101). From a practical point of view it seems important to note that theoretically efficient algorithms are space-consuming requiring several arc-length working arrays (however see (85) for an implementation of Karzanov's $O(|V|^3)$ algorithm using only four arc-length arrays - a FORTRAN listing is also provided in this reference).

From the experience reported in [71] about augmenting path algorithms, it appears that the algorithm REVERSE proposed by Edmonds and Karp [25] is amenable to much more efficient implementation (using only three arc-length arrays) than their most celebrated algorithm SHORT (incidentally, this last algorithm is often referred to as "the" Edmonds-Karp algorithm, while these authors proposed in fact three different max. flow algorithm in their paper). It appears that specializations of the simplex algorithm would provide the most space-preserving implementations, using only two arc-length arrays. Solving maximum flow problems by the simplex algorithm has received some attention [59],[30],[19],[41] but (except for the last reference) little computational experience has yet been reported.

* * *
* *

3. DIRECT APPLICATIONS OF MAXIMUM FLOW THEORY

A number of combinational applications of maximum flows, and of the dual relations with minimum cuts, is described in the second chapter of the book[33] by Ford and Fulkerson. We will briefly outline these, before describing an interesting application to pre-emptive scheduling.

Tools provided by the maximum flow theory help to establish feasibility theorems, including the supply-demand theorem of Gale (conditions under which given demands at some vertices may be satisfied from supplies at some other vertices, subject to capacity constraints on the arcs or edges), a symmetric supply-demand theorem of Fulkerson (when there are, in addition, both upper and lower limits on the net flow in every vertex) and the circulation theorem of Hoffman (conditions of existence of flows that are source and sink free, that satisfy prescribed lower and upper bounds on arcs). In particular the problem of finding a feasible flow (subject to both lower and upper arc capacities, and prescribed net flow in every vertex), which occurs as a "Phase I" in minimum cost flow problem can be formulated as a maximum flow problem: by the variable change

$$f'(a) = f(a) - l(a) \tag{3-1}$$

for all arcs a, the bounds on arc flows

$$l(a) \leq f(a) \leq c(a) \tag{3-2}$$

become

$$0 \leq f'(a) \leq c'(a) = c(a) - l(a) \tag{3-3}$$

and the balance equations

$$\sum_{h(a)=i} f(a) - \sum_{t(a)=i} f(a) = b_i \qquad (3\text{-}4)$$

for all vertices v_i (where $\sum_i b_i = 0$), become

$$\sum_{h(a)=i} f'(a) - \sum_{t(a)=i} f'(a) = b'_i = b_i - \sum_{h(a)=i} l(a) + \sum_{t(a)=i} l(a) \qquad (3\text{-}5)$$

A (super-)source is added and connected to all vertices v_i with negative b'_i by an arc with capacity $-b'_i$, and all vertices v_j with positive b'_j are connected to a (super-)sink by an arc with capacity b'_j. Then a maximum flow is sought from source to sink. There exists a flow which saturates all the source-arcs (and by the way all the sink-arcs) if and only if the initial problem admits a feasible flow.

The <u>minimum flow problem</u> is the problem of finding a flow of minimum value v(f) (defined by (1-2)) subject to balance equations (1-1) and to lower bounds on arc flows:

$$l(a) \leq f(a) \qquad (3\text{-}6)$$

A feasible flow f^0 is found by a straightforward $O(|A|.|V|)$ procedure and the following variable change is performed

$$f'(a) = f^0(a) - f(a), \qquad (3\text{-}7)$$

This result in a maximum flow problem in a network $N' = (V,A')$ deduced from N by reversing the arc directions, imposing the flow constraints

$$f'(a) \leq c'(a) = f^0(a) - l(a) \tag{3-8}$$

(note that lower capacities are $-\infty$) and maximizing the flow from t to s. The max.-flow min.-cut theorem applied to N' translates into: the minimum value of a (s,t)-flow in N is equal to the maximum capacity of a cut $(S,\bar{S})$ separating s from t and such that there is no arc in A with head in S and tail in $\bar{S}$. This result is sometimes called the "min.-flow max.-cut theorem", but we believe that this terminology may be misleading: finding a maximum cut (without the above restriction) is usually much harder than finding a minimum flow, see section 9 . Note that this restriction comes from the lower capacities equal to $-\infty$: indeed, if no cut satisfying this restriction exist, then there is a directed path from t to s in N, and the min. flow problem is unbounded.

Another classical area of application of the max.-flow min.-cut theory is to bipartite matching. In a bipartite graph, a matching is a set of arcs which have no vertex in common. The reader is referred to Chapter 5 of [72] for an extensive treatment. Among many applications we mention the following[102]: given a large sparse square matrix to be inverted, find a matching of rows to columns using only nonzero elements. If the matrix is regular we know that such a matching exists. This is used as a first step in inversion subroutines for mathematical programming systems.

A matching containing a maximum number of arcs can be found, in bipartite graphs, by the Ford-Fulkerson Labelling algorithm, which may be implemented in $O(|V|.|A|)$ time, or by the $O(|V|^{5/2})$ algorithm by Hopcroft and Karp[53]. The max. flow min. cut theorem translates into the celebrated König-Egervary theorem, which related the maximum cardinality of a matching to the minimum number of vertices which "cover" the arcs of a bipartite graph (i.e. which include at least one end point of every arc), and also to the maximum number of vertices, no two of which are connected by an arc in this bipartite graph (the latter vertex set being the complement of the former). Also related to bipartite matching are the systems of distinct representatives, see[33 pp.67-75].

Other classical applications include the study of the (s,t)-connectivity of directed graphs and a proof of Menger's theorem (see also [109] and section 6); the chain decomposition of a partial order and a proof of Dilworth theorem, with a nice application to finding the minimum number of individuals to meet a fixed schedule of tasks[33 pp.61-67]; the degree-constrained partial graph problem, which is applied to find a Euler tour in directed, undirected or mixed graphs - a step in the solution of the Chinese Postman Problem[24] (involved in garbage collection or snow removal) and in Christofides' heuristic for the Traveling Salesman Problem[16] - and to 0-1 matrices with prescribed row and column sums (see[33 pp.79-91]).

Maximum flows may also be used to find a pre-emptive scheduling of jobs to parallel machines, subject to job release and due dates[10],[52]. Given n jobs $J_j (j = 1, \ldots, n)$ with release dates r_j and due dates d_j, we define a set of time intervals $E_k (k = 1, \ldots, p \text{ with } p \leq 2n - 1)$ by sorting all these 2n dates into a sorted list $(e_1, e_2, \ldots, e_p)$ and setting $E_k = [e_k, e_{k+1}]$. Then we construct a network with n job vertices, p interval vertices, a source s and a sink t. The arcs are as follows.

(s, J_j) with capacity equal to the processing time requirement p_j, for all jobs J_j:

(J_j, E_k) with capacity equal to $e_{k+1} - e_k$, for all jobs J_j and intervals E_k such that $r_j \leq e_k$ and $e_{k+1} \leq d_j$;

(E_k, t) with capacity equal to $m(e_{k+1} - e_k)$ for all intervals E_k, where m is the number of parallel machines.

A feasible schedule exists if and only if there is a flow of value $\sum_j p_j$ in this network, and a schedule is easily deduced from the corresponding flow. This construction may be used in an iterative approach to minimize the maximum flow-time and the maximum lateness. For both problems a trial value for the objective function is tested for feasibility and then adjusted for a next step. In the first approach the trial value F is inserted in the list $(e_1, e_2, \ldots, e_p)$ and all entries greater than F are discarded, such that the list becomes $(e_1, e_2, \ldots, e_q)$ with $e_q = F$.

In the second problem, for a trial L the entries $l_j = r_j + L$ are introduced in the list whenever $l_j < d_j$. See[69] for a detailed description of a polynomially bounded adjustment procedure, and[11] and[69] for an extension to the case of two machines with different speeds.

* * *
* *

4. EXTENSIONS OF MAXIMUM FLOW THEORY

The classical maximum flow theory has been extended in several directions which will be briefly reviewed below.

In multiterminal analysis, the problem is to find all the maximum flows (and/or minimum cuts) between all couples of nodes in a network, without solving all the $O(|V|^2)$ such problems. The subject has been pioneered by Gomory and Hu[42],[43]; see[33, chap.4], [54, chap.9], [34, chap.5] and also[106],[107] for further references.

In dynamic flows, consideration is given to transit times on arcs, in an attempt to best modelling real-life flow behavior. The basic reduction to static network flows was given by Ford and Fulkerson[32] (see also[33]). A detailed review of dynamic flow algorithms is found in[81, section 4.5], see also[46] and the survey[9]. Kaufman[64] describes an application to rail networks in France. Another very interesting application to the modelling of building evacuation has been recently developed by R. Francis at the National Bureau of Standards (Washington, D.C.): such models can recommend routings of people from work places to exits so as to minimize building evacuation time, and can also identify evacuation bottlenecks.

Other network problems can be seen as extensions of maximum flow theory: these include minimum cost flow problems, multicommodity problems and flow with gains (or generalized network flows). Since these problems are dealt with elsewhere in this issue, we shall not develop any longer about them here.

* * *
* *

5. FINDING MINIMUM CUTS

The most direct method for finding a minimum cut is the brute enumeration of the $2^{|V|-2}$ possible cuts. Clearly, when the network has more than a very few vertices, approaches based on maximum flow are preferable. When a maximum flow f is available, identifying one minimum cut may be achieved in $O(|E|)$ time by applying the Labelling Procedure. This procedure, initiated from the source, identifies the minimum cut with the smallest possible source-set S. Conversely this Labelling Procedure may be adapted to start from the sink, and thus identifies the minimum cut with largest source-set S' (see[33, pp.13-14]). When these two sets differ, there may exist several other minimum cuts.

The problem of finding all minimum cuts may be difficult: consider for example a network $N = (V,A)$ with $A = \{(s,i) : v_i \in V - \{s,t\}\} \cup \{(i,t) : v_i \in V - \{s,t\}\}$, and all capacities are unity; in this network all of the $2^{|V|-2}$ possible cuts are minimum cuts. The complementary slackness conditions using the maximum flow f at hand, may be interpreted as follows:

if $f_{ij} < c_{ij}$ or $f_{ji} > 0$ then there is no cut $(S,\overline{S})$ with $i \in S$ and $j \in \overline{S}$

These conditions may be used to define a binary relation R on the vertex set V such that $(S,\overline{S})$ is a minimum cut if and only if S is a closure with respect to R, i.e.

$$(v_i \in S \text{ and } v_i R v_j) \Rightarrow V_j \in R,$$

containing the source and not the sink. This relation R may be used to enumerate all closures, and also to streamline sensitivity and parametric analysis for maximum flow and minimum cut, see[94].

There are other approaches for finding minimum cuts without necessarily identifying a maximum flow. In the Cut Search Algorithm of Phillips and Dessouky[87], the Decomposition Algorithm of Jarvis and Tufekci[58], as well as in an algorithm suggested by Topkis[111], sequences of minimum cut problems are solved for networks smaller than the initial network, usually by adjusting maximum flows from a previous iteration. While none of these methods presents a worst-case behaviour better than the Labelling Method, say in its $O(|V|^3|E|)$ version, the computational experience reported in[87](for networks with both lower and upper capacities) suggests than such algorithms may offer computational efficiency.

There is an intriguing asymmetry in the duality relationship between maximum flows and minimum cuts. Given a maximum flow, a minimum cut may be located by a $O(|E|)$ procedure. But giving a minimum cut (or even all the minimum cuts) does not help much for producing a maximum flow. It might be that the minimum cut problem is in fact easier than the maximum flow problem. For instance, Karp[63] has shown that in almost all networks (in a precise probabilistic sense) the minimum cut is defined by the source- or the sink-arcs. Such observations indicate that it could be fruitful to look at approaches, such as those mentionned in the previous paragraph, which identify a minimum cut in a network without producing a maximum flow.

* * *
* *

6. DIRECT APPLICATIONS OF MINIMUM CUTS

The most straightforward applications of minimum cuts arise within the context of disconnection of networks: a minimum cut can be seen as a set of arcs which intersects any path from the source s to the sink t, at minimum cost. This can be interpreted as the interruption of communication between s and t, or as the interdiction of the physical transportation of supplies, troops, ... in the context of defense or attack of network. Indeed, it is a concrete project to evaluate the capacity of the Eastern European rail network to support a large scale conventional war, and the effort required for interdiction, formulated by General F.S. Ross and T.E. Harris, which motivated the interest of L.R. Ford and D.R. Fulkerson in network flows and led to their discovery of the max.-flow min.-cut theorem (see[7],[51]). More general problems concerning the vulnerability of networks involve the attack of both arcs and vertices. The maximum flow approach can be extended to such problems by introducing vertex capacities, as it was seen before. In fact the Labelling Algorithm can be efficiently specialized to such problems, see[35]. A detailed exposition of connectivity and vulnerability of both deterministic and probabilistic graphs is found in the classical book of Frank and Frisch[34]; see also[110],[115].

A related problem is that of partitionning a graph. A minimum cut provides a way to partition a graph in two disconnected parts, with the minimum number (or total weight) of edges between them[104]. A good

discussion of this as well as other approaches for graph partitionning is found in[66]. Another related problem is the determination of the reliability of a communication network. A simple model[21] is the following: every arc has a probability p of failure, and all the failures are independent events. The probability that s and t are disconnected is

$$\sum_{k=1}^{|A|} A_k p^k (1-p)^{|A|-k},$$

where A is the arc set and A_k is the number of subsets of k arcs in A which disconnect s from t. For very reliable networks, where p is very small, a good approximation for this probability is $A_{k*}p^{k*}(1-p)^{|A|-k*}$, where k* is the minimum number of arcs in a cut separating s from t; if a numerical value for P(s,t) is sought, A_{k*} may be computed by enumerating all these minimum cuts. For "less reliable" networks, this approximation is no longer valid, and the computation of the exact probability is quite difficult, see[3] and[112].

Other applications of minimum cuts arise in the study of project networks. Given a project network (of the PERT or CPM type, see[28]), assume that some activities may be compressed from their "normal duration", but cannot require less than some "crash durations", and the cost for such compressions is assumed to be linear in this duration interval. It is desired to produce a project cost curve giving, for every possible budget, the optimum project duration and a corresponding set of activities to be

compressed. This parametric problem was solved independently by Fulkerson[36] and Kelley[65] in 1961, through parametric solution of a minimum cost flow problem obtained as a dual of a linear programming formulation of the problem. Phillips and Dessouky[86] have shown that the problem may be solved as a sequence of minimum cut problems: it is clear that, for a small enough budget, the only activities which need to be compressed are the critical ones (i.e. those which any delay causes a delay to the overall project); these critical activities form a "critical network" (a union of paths from the source, representing the project start, and the sink, representing the project completion) and a minimum cost cut in this critical network defines an optimum set of activities to be compressed. In subsequent stages, other activities become critical. In addition, it is necessary to allow a previously compressed activity to be lengthened in a later stage. This leads to consider cuts with both upper and lower capacities, the latter representing the savings made in lengthening previously compressed activities. It is to be noted that this approach, intuitively more appealing, is in fact equivalent to the Fulkerson-Kelley approach, since same sequences of minimum cuts may be produced by both approaches. The problem has been generalized to include a linear penalty for tardiness of a set of key events, see[29]. Other related parametric problems, of <u>increasing</u> the length of <u>shortest</u> paths are considered in[37] and[40].

* * *
* *

7. A BINARY QUADRATIC PROGRAMMING FORMULATION OF MINIMUM CUTS[47][95]

For simplyfying the following developments, let us assume that the vertices of the network are numbered as follows: let $n = |V|-2$, v_0 be the source, v_{n+1} be the sink, and $v_1, v_2, \ldots, v_n$ be the other vertices. With every cut $(S,\bar{S})$ we associate its characteristic vector $x = (x_1, x_2, \ldots, x_n)$ where:

$$x_i = \begin{cases} 1 \text{ if } v_i \in S \\ 0 \text{ otherwise.} \end{cases} \tag{7-1}$$

There is a one-to-one correspondence between cuts in N and binary n-vectors. Note that we omitted the two components x_0 and x_{n+1} from x since we may set $x_0 = 1$ and $x_{n+1} = 0$ for every cut. The capacity of the cut associated with x is:

$$c(x) = \sum_{i=0}^{n+1} \sum_{j=0}^{n+1} c_{ij} x_i (1-x_j) \tag{7-2}$$

(where c_{ij} is the capacity of the arc $(v_i,v_j) \in A$, and zero if $(v_i,v_j) \notin A$). After substituting for x_0 and x_{n+1}, and noting that $x_i^2 = x_i$ for every binary x_i, we obtain

$$c(x) = a + \sum_{i=1}^{n} b_i x_i - \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_i x_j \tag{7-3}$$

where $a = \sum_{j=1}^{n+1} c_{oj}$, the capacity of the cut defined by $S = \{v_0\}$ (7-4)

and $$b_i = \sum_{j=1}^{n+1} c_{ij} - c_{io} \quad \text{for all } i = 1, 2, \ldots, n. \tag{7-5}$$

Thus the minimum cut problem may be stated as the following quadratic binary programming problem:

$$\text{minimize } \{bx - x^t Cx : x \text{ is binary}\} \tag{7-6}$$

or $$\text{maximize } \{x^t Cx - bx : x \text{ is binary}\}. \tag{7-7}$$

Conversely any problem of type (7-6) or (7-7) can be interpreted as a minimum cut problem in a directed network. Note that we must have $C \geq 0$ for applying the max.-flow min.-cut theory. The linear part b is unrestricted in sign.

Minimum cuts in undirected and mixed networks [96][97]

Consider the problem of finding a minimum cut, in an undirected or mixed network: it follows from the formulation (7-1), (7-2) that an equivalent problem is obtained by arbitrarily directing the undirected arcs (edges) and appropriately redefining the source- and sink-arcs in accordance with (7-5). Conversely, any minimum cut problem, stated in the format (7-6) may be set into the symmetric form, i.e. a form in which the matrix C is symmetric, by redefining c_{ij} as $\frac{1}{2}(c(i,j) - c(j,i))$ for all i,j with $i \neq j$.

Lower capacities

As in Section 3, assume we have a network with two numbers associated with every arc (i,j): upper capacity u_{ij} and lower capacity ℓ_{ij}. Defining the capacity of a cut $(S,\bar{S})$ as

$$c(S,\bar{S}) = u(S,\bar{S}) - \ell(S,\bar{S})$$

we may state the corresponding minimum cut problem as:

$$\text{minimize } \{c(x) : x_0 = 1,\ x_{n+1} = 0 \text{ and } x \text{ binary}\}$$

where $$c(x) = \sum_{i=0}^{n+1} \sum_{j=0}^{n+1} (u_{ij} - \ell_{ji})\, x_i(1 - x_j),$$

that is exactly in the format (7-6) with $c_{ij} = u_{ij} - \ell_{ji}$. Note that we may have some $c_{ij} < 0$. Source- or sink-arcs with negative capacities are simply redefined as sink- and source-arcs, respectively, with the opposite capacity; this implies a translation of the objective function c(x). When an arc (i,j) with intermediate endpoints has a negative c_{ij}, it must be true that

$$c_{ji} \geq - c_{ij}$$

(since $\ell_{ij} \leq u_{ij}$ and $\ell_{ji} \leq u_{ji}$). Thus we may reduce c_{ji} and increase c_{ij} by the same amount, e.g. by c_{ij} (to suppress the arc (i,j)) or by $\frac{1}{2}(c_{ji} + c_{ij})$ (to make the network symmetric), and again appropriately redefine the source- and sink arcs incident to v_i and v_j.

Maximum-weighted closure of a graph[88]

Given a directed graph $G = (V,A)$ with vertex weights w_i (of arbitrary sign), we define as a closure any subset $U \subseteq V$ such that $v_i \in U$ and $(v_i,v_j) \in A$ imply $v_j \in U$ (this is also named a hereditary subset[44], an initial subset[74] or a selection[103],[2]). The maximum-weighted closure problem is the problem of finding a closure U with maximum weight $w(U) = \sum_{i \in U} w_i$. By using the characteristic vector y of U, this problem can be formulated as:

$$\text{maximize} \quad \sum_i w_i y_i \tag{7-8}$$

$$\text{subject to} \quad y_i(1-y_j) = 0 \quad \text{for all } (v_i,v_j) \in A \tag{7-9}$$

$$\text{and} \quad y \text{ is binary} \tag{7-10}$$

Since $y_i(1-y_j)$ is always nonnegative for any binary y, we may introduce (7-9) in the objective function with a suitably large Lagrange multiplier λ:

$$\text{maximize} \quad \left\{ \sum_i w_i y_i - \sum_{(v_i,v_j) \in A} \lambda y_i(1-y_j) \;:\; y \text{ is binary} \right\} \tag{7-11}$$

(taking $\lambda = 1 + \sum_i |w_i|$ will suffice). This problem is of type (7-7), and thus is equivalent to a minimum cut problem in a network constructed as follows: assign to the arcs of G a capacity λ; add a "dummy" source s connected to the vertices v_i such that $w_i \geq 0$ by an arc with capacity w_i; add a dummy sink t connected to the vertices v_i such that $w_i < 0$ by an arc with capacity $-w_i$. A minimum cut $(S,\bar{S})$ in this network defines a maximum-

weighted closure U = S -{s}. (For subsequent sensitivity and parametric analysis, it might be useful to connect every node to both the source and the sink, one of every such pair of arcs having null capacity).

A typical problem that can be formulated as a maximum-weighted closure problem is the selection of contingent investments[113]: we are given a set of "projects" and "contingency" relations between them; project i is contingent to project j means that if we decide to select project i that we must also select project j (for instance the acquisition of a memory extension is contingent to the acquisition of a computer, while the converse is not necessarily true); with every project is associated a net profit (which may be negative for a project presumably useful to the selection of other more profitable projects) and we seek a selection with maximum net profit.

One instance of considerable importance is the determination of the optimal contour of an open-pit mine[61]: in this planning problem there are slope constraints which prevent the walls of the mine to be too steep, for otherwise they might cave in; given drill hole data for estimating the distribution of ore grade within the deposit, and relevant economic data, the problem is to determine the most profitable ultimate pit limits. It is useful to approach this problem by dividing the deposit (i.e. all of the volume which may potentially be included in the ultimate limit) into blocks of appropriate sizes, such that the removal of a given

block implies the removal of the blocks located above it in the next upper layer. Then, for every block, is computed an economic measure such as the net profit - value of the ore contained in the block minus the exploitation and processing costs. This result in a maximum-weighted closure problem in a highly-structured graph with thousands or tens of thousands of vertices (i.e. blocks).

Other notable applications[103] involve the selection of profitable activities (e.g. production lines, trade financing, transportation of goods) requiring the shared use of expensive facilities (e.g. machines, trade offices, terminals). In this case the graph is bipartite, and the minimum cut equivalence was noted first by Rhys[103] and Balinski[2]. It is possible to reduce a closure problem with an arbitrary graph to the bipartite case[103],[60],[74],[90] but this does not seem to be the best thing to do. For the open-pit mine application, T.B. Johnson[61] reported that this (bipartite) "network flow technique" involves solution times close to that of the classical Lerchs-Grossman method[76]. On the other hand, recent experiments by Plasse and Elbrond (Département de Génie Minéral, Ecole Polytechnique de Montréal), using the original nonbipartite sparse graph (and a crude maximum flow algorithm) indicated a marked superiority over a state-of-the-art implementation of the Lerchs-Grossman method, for a "small" example with 400 blocks; and Chvàtal (Department of Computing Science, McGill University, Montreal) have undertaken to develop an appropriate implementation of this nonbipartite minimum cut method for problems of realistic size.

In [78] McGinnis and Nuttle have shown how this activity selection can be combined with CPM (or PERT) analysis for solving the following project coordinator's problem: select a subject of activities to be scheduled and determine their start time so that given prerequisite and corequisite restrictions are satisfied and no activity is in progress beyond a fixed planning horizon, in order to maximize the total net profit.

Binary posynomial maximization [93]

Given a vector $y = (y_1, y_2, \ldots, y_n)$ of binary variables, a posynomial $P(y)$ is any polynomial

$$P(y) = \sum_{T \subseteq N} q_T \prod_{i \in T} y_i \qquad (7\text{-}12)$$

where $q_T \geq 0$ for all $T \subseteq N = \{1, 2, \ldots, n\}$. Maximizing $P(y)$ has the trivial solution $y = \underline{1}$. Define as a binary posynomial maximization problem a problem of the following type:

$$\text{maximize } \{ P(y) - by : y \text{ is binary} \} \qquad (7\text{-}13)$$

where $P(y)$ is a posynomial and b is an arbitrary real n-vector. For instance , problem (7-7) is of this type. The problem (7-13) can be reduced to a maximum-weighted closure problem in a graph defined as follows: its node set includes all the subsets $T \subseteq N$ such that $q_T > 0$ with weight equal to q_T, and all singletons $\{i\}$ with weight $q_{\{i\}} - b_i$; there is an arc (T,T') in this graph whenever $T \supseteq T'$ (in fact, if $T \supseteq T' \supseteq T''$, the arc (T,T'') is superfluous).

* * *
* *

8. PROBLEMS SOLVABLE BY A SEQUENCE OF MINIMUM CUT PROBLEMS

The binary quadratic programming formulation of the minimum cut problem allows to solve several problems as a sequence of minimum cut problems in related networks. In the following applications, the number of nodes is the successive networks is nonincreasing (and generally strictly decreasing). Except for one case, the number of iterations is bounded by the number of nodes in the initial network.

Maximum-ratio closure problem [92],[93]

Consider a directed graph with two weights $w_i \geq 0$ and $p_i > 0$ attached with every vertex v_i. The problem considered is to find a nonempty closure U which maximizes the ratio $w(U)/p(U)$. In the context of investment selection, the weights p_i are interpreted as the cost of selecting project i, and thus it is wished to maximize the profit over cost ratio. Consider also the problem of sequencing jobs on a single machine to minimize the total weighted completion times subject to precedence constraints; Sidney [108] has proposed the following decomposition procedure: find a closure U_1 (relative to the precedence graph) of the job set N with maximum total weight over total processing time ratio; then continue finding a maximum ratio closure U_2 in $N-U_1$, and so on until the job set N is exhausted (in addition it is required that those optimal closures be minimal for the inclusion, a condition which derives from a property of the Labelling Procedure).

A useful scheme for dealing with a fractionnal objective $w(U)/p(U)$ is due to Isbell and Marlow[56]: pick a trial value r (possibly the ratio corresponding to a current feasible solution) and find a feasible solution which maximize

$$z(U) = w(U) - rp(U) \tag{8-1}$$

If $z(U) > 0$, U is a feasible solution with ratio larger than r, while if $z(U) = 0$ then U is an optimum solution to the maximum ratio problem. Finally if $z(U) < 0$ then there is no feasible solution with ratio greater than or equal to r.

In a primal method, we start with a feasible solution and successively improve it by solving (8-1) with r set at the value of the ratio of the current solution. For the maximum-ratio closure problem, starting with the trivial full closure $U = V$ generates a sequence $U^1, U^2, \ldots, U^k$ of closures such that $U^{k+1} \subsetneq U^k$ [92]. It follows that there will be at most $|V|$ minimum cut problems to solve.

In a primal-dual method, we maintain a couple (r^-, r^+) of values such that segment $[r^- r^+[$ contains the optimum ratio. Lawler[74] proposed to carry out a binary search in which the trial value r is $\frac{1}{2}(r^- + r^+)$. He shows that, when all weights are integer, a maximum ratio closure is obtained after at most $(2\log_2|V| + \log_2(\max_i w_i) + 2\log_2(\max_i p_i) + 1)$ iterations. For "not too large" integer weights, this bound improves on the

bound for the primal method, while the latter seems, in general, more attractive since it does not depend at all on the weights.

Hyperbolic bivalent posynomial maximization[93]

The previous approaches extend immediately to the following problem:

$$\text{maximize } \{(P(y) - by)/(dy - Q(y)) : y \neq 0, \text{ bivalent}\} \tag{8-2}$$

where $P(y)$, $Q(y)$ are posynomials,

$P(y) - by \geq 0$ and $dy - Q(y) > 0$ for all bivalent $y \neq 0$.

In this case, the primal method (starting with $y^0 = 1$) generates a sequence $y^0, y^1, \ldots, y^k$ of solutions with $y^{k+1} \leq y^k$. Thus there are at most n minimum cut problems to solve.

Applications to graph theory[92],[93]

Consider an undirected graph $G = (V(G), E(G))$. We will denote by $H \leq G$ the fact that H is a subgraph of G, i.e. $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$. A forest is a subgraph with no cycle and a pseudoforest is a subgraph which contains at most one cycle in each of its connected components. Two invariant numbers of a graph are defined: the arboricity $\alpha(G)$ (resp. pseudoarboricity $p\alpha(G)$) of a graph G is the minimum number of forests (resp. pseudoforests) which cover all the edges of G.

Theorem (Nash-Williams[83])

$$\alpha(G) = \left\lceil \max \left\{ \frac{|E(H)|}{|V(H)|-1} : H \leq G,\ |V(H)| \geq 2 \right\} \right\rceil$$

Theorem[92]

$$\alpha(G) = \left\lceil \max \left\{ \frac{|E(H)|}{|V(H)|} : H \leq G,\ |V(H)| \geq 1 \right\} \right\rceil .$$

($\lceil x \rceil$ denotes the smallest integer greater than or equal to x).

The pseudoarboricity $p\alpha(G)$ may be computed by solving the following hyperbolic bivalent quadratic problem

$$\max_{\substack{x_i=0,1 \\ x \neq 0}} \frac{\sum_{(v_i,v_j) \in E(G)} x_i . x_j}{\sum_{v_i \in V(G)} x_i}$$

This problem can be solved in $O(\log_2 |V|)$ iterations (for simplifying the notations, we set $V = V(G)$) by using binary search, each of these iterations involving the solution of a minimum cut problem in a network with a most $|V| + 2$ nodes. Thus the overall complexity of this approach is $O(|V|^3 \log |V|)$. In addition, the maximum flow produced in the last iteration may be used to construct a decomposition of G into pseudoforests (see[93]).

Computing the arboricity is a little bit more complicated due to the presence of the constant -1 in the denominator. In[93] an approach requiring the computation of $O(|V|)$ pseudoarboricity of subgraphs of G is

described; its complexity is thus $O(|V|^4 \log|V|)$. This approach does not seem to yield a decomposition of G into forests. Such a decomposition can be obtained by the Matroid Partitionning Algorithm of Edmonds[23].

A class of quadratic integer programming problems[99]

Consider a bounded integer program of the form:

$$\text{minimize} \quad \{f(y) : \ell \leq y \leq u, \ y \text{ integer}\}$$

$$\text{where} \quad f(y) = y^T Q y + by \tag{8-3}$$

Q is a symmetric matrix with $q_{jk} < 0$ for $j \neq k$

$$\text{and} \quad \sum_{j=1}^{n} q_{jk} \geq 0 \quad \text{for all } j$$

ℓ and u are nonnegative integer vectors.

As an application, consider the problem of locating, in the p-dimensional Euclidian space, n new facilities among the points with integer coordinates, with respect to m existing facilities, in order to minimize a weighted combination of all the squared-Euclidian distances

$$g(x) = \sum_{j=1}^{n} \left[\sum_{k=1}^{n} v_{jk} \left(\sum_{s=1}^{P} (x_{js}-x_{ks})^2 \right) + \sum_{i=1}^{m} w_{ji} \left(\sum_{s=1}^{P} (x_{js}-a_{is})^2 \right) \right] \tag{8-4}$$

where v_{jk} (resp. w_{ji}) is a measure of the traffic between facilities j and k (resp. j and i),

$a_{i1}, \ldots, a_{ip}$ are the (integer) coordinates of the existing facilities, and

$x_{j1}, \ldots, x_{jp}$ are the (integer) coordinates of the new facilities to be located.

This integer squared-Euclidian distance location problem decomposes into p disjoint one-dimensional location problems, all of which are easily put into the form (8-4).

If u is tentatively set to $u = \ell + \underline{1}$, problem (8-3) reduces to the form (7-6). If x* denotes an optimum solution to this restricted problem, Picard and Ratliff show[99] that there exists an optimum solution x^0 to the original problem (8-3) such than $x^0 \geq x^*$. Hence, for each j with $x_j^* = \ell_j + 1$, we can increase the lower bound on x_j by one. This process is repeated until an optimum solution x* is obtained, in which any x_j^* is equal to its current lower bound. It is shown that such a solution is optimum to the original problem (8-3), which is thus solved by a sequence of at most $\sum_{j=1}^{n} (u_j - \ell_j)$ minimum cut problems.

The rectilinear distance facility location problem[98]

This problem is similar to the previous location problem, but with rectilinear distances, instead of the squared-Euclidian distances. In other

words, in (8-4) every term $(x_{js} - x_{ks})^2$ (resp. $(x_{js} - a_{is})^2$) is replaced by $|x_{js} - x_{ks}|$ (resp. $|x_{js} - a_{is}|$). In addition we no longer insist on having integer coordinates for either the new or existing facilities. In[77], Love and Yerex present an application of this model to the location of production facility in the prestressed concrete industry. As it was the case for the squared-Euclidian distance problem, this rectilinear distance location problem decomposes into p disjoint one-dimensional problems. Wesolowski and Love[114] have shown that it reduces to a linear programming problem, while Cabot, Francis and Starry[13] showed that it is in fact the dual of a minimum-cost flow problem. A consequence of these reductions is the following: there is an optimum solution in which every coordinate of every new facility is the corresponding coordinate of an existing facility.

In[98], Picard and Ratliff showed that, in the one-dimensional problem, the optimum location of the new facilities is dependent on the relative positions of the old facilities, but not on the distances between them. Let the old facilities be numbered (for a particular dimension considered) in nondecreasing order of their coordinates, i.e. such that $a_1 < a_2 < \ldots < a_m$ (here, old facilities with the same coordinate are assumed to be confounded). Then consider the restricted problem of locating the new facilities only among the coordinates a_q or a_{q+1}. Since the distance are not relevant (or just by factoring $(a_{q+1} - a_q)$) we may set $a_{q+1} = a_q + 1$ and thus the restricted problem reduces to

$$\text{minimize} \quad \sum_{j=1}^{n}\left[\sum_{k=1}^{m} v_{jk}\left(y_j(1-y_k) + y_k(1-y_k)\right) + b_j y_j\right] - b_0 \qquad (8\text{-}5)$$

where $y_j = 1$ if the new facility is located in a_{q+1}

or 0 if it is located in a_q ,

$$b_j = \sum_{i=1}^{q} w_{ji} - \sum_{i=q+1}^{m} w_{ji}$$

and $$b_0 = \sum_{j=1}^{n} \sum_{i=q+1}^{m} w_{ji}.$$

This problem can be set into the format (7-6). Let y* be an optimum solution to this problem. It is shown that there exists an optimum solution to the original problem in which a new facility j is located among the coordinates $a_1, \ldots, a_q$ if $y_j^* = 0$, and among the coordinates $a_{q+1}, \ldots, a_m$ if $y_j^* = 1$. The problem thus reduces to two disjoint smaller problems which are decomposed in the same fashion. This solution procedure involves at most (m-1) minimum cut problems to be solved for every dimension. Subsequently, Cheung[14], Drezner and Wesolowsky[22] and Koolen[67] have shonw how this procedure relates to the minimum-cost flow formulation of Cabot et al.[13].

* * *
* *

9. MORE DIFFICULT MINIMUM CUT PROBLEMS

Other problems may be formulated as minimum cut problems involving either negative capacities or additional constraints. This usually results is NP-complete problems. However this minimum cut formulation may be useful for providing relaxations and insights into the structure of the problem at hand.

Negative capacities

It is possible to extend the definition of the capacity of a cut to negative capacities. If for some pair (v_i, v_j) of vertices we have $c_{ij} + c_{ji} < 0$, the the approaches based on flows are no longer applicable.

The maximum cut problem can be formulated this way, by simply taking the negative of all capacities. Since it is NP-complete[62], it follows that the minimum cut problem with negative capacities is NP-complete.

For dealing with negative capacities, Hansen[49] suggested to replace some variables x_i by their complement $\bar{x}_i = 1 - x_i$. Finding a subset of variables, if any, the substitution of which by their complement disposes of all negative capacities, is equivalent to "balancing" a "signed graph" (see[50]), and can be performed very efficiently. If this fails, it may be useful to find a subset of variables which minimizes the total absolute value of all negative capacities remaining after this complementation. While this task is in general as difficult as solving the original

problem, heuristics may be useful to reduce this "gap" for the subsequent application of another algorithm (e.g. a branch-and-bound algorithm).

A general formulation for the integer linear programming problem with bounded variables is the following (see[39]).

$$\text{minimize} \quad \{cy : Ay = b, \ y \text{ is binary}\} \tag{9-1}$$

The equations $Ay = b$ are equivalent to the single equation $(Ay-b)^T(Ay-b)=0$ which can be introduced into the objective function with a suitably large Lagrange multiplier λ. The resulting problem[95] is in the format

$$\text{minimize} \quad \{dy - y^T Q y : y \text{ is binary}\} \tag{9-2}$$

where $Q = -\lambda A^T A$ and $d = c - 2\lambda b^T A$. Unfortunately there may be negative nondiagonal entries in the matrix Q.

However, we may use this approach to gain some insight in the structure of the problem: for instance, assume that we have found (maybe within a branch-and-bound algorithm) a feasible solution $\bar{y}$. By defining

$$\tilde{y}_j = y_j, \ \tilde{c}_j = c_j \text{ and, for all } i, \ \tilde{a}_{ij} = a_{ij} \quad \text{if} \quad \bar{y}_j = 0, \tag{9-3}$$

$$\tilde{y}_j = 1-y_j, \ \tilde{c}_j = -c_j \text{ and, for all } i, \ \tilde{a}_{ij} = -a_{ij} \quad \text{if } \bar{y}_j = 1, \tag{9-4}$$

we observe that problem (9-1) is equivalent to

$$\text{minimize} \quad \{\tilde{c}\tilde{y} : \tilde{A}\tilde{y} = 0, \ \tilde{y} \text{ is binary}\} \tag{9-5}$$

or minimize $\{\tilde{c}\tilde{y} - \tilde{y}^T\tilde{Q}\tilde{y} : \tilde{y} \text{ is binary}\}$ (9-6)

where $\tilde{Q} = -\lambda\tilde{A}^T\tilde{A}$. Now we use a strategy of <u>restriction</u> , that is we add to problem (9-5) the following constraints:

$$\tilde{y}_j = \tilde{y}_k \text{ for all } j,k \text{ such that } j \neq k \text{ and } \tilde{Q}_{jk} < 0 \tag{9-7}$$

The problem remains feasible since $\tilde{y} = 0$ works, and these constraints are enforced by simple substitution, reducing the number of variables in the resulting problem. Let the resulting problem, after these reductions, be

$$\text{minimize} \quad \{\hat{c}\hat{y} : \hat{A}\hat{y} = 0, \hat{y} \text{ is binary}\} \tag{9-8}$$

$$\text{or minimize} \quad \{\hat{c}\hat{y} - \hat{y}^T\hat{Q}\hat{y} : \hat{y} \text{ is binary}\} \tag{9-9}$$

Since $\hat{Q} = -\lambda\hat{A}^T\hat{A}$ contains no negative nondiagonal entry, problem (9-9) is solvable as a minimum cut problem. Indeed by taking very large λ, the problem is seen to reduce to a minimum-weighted closure problem in a graph defined by all the nonzero entries of $\hat{A}^T\hat{A}$. If a solution $\hat{y}$ with negative weight $\hat{c}\hat{y}$ is found, then it induces an improved feasible solution to problem (9-1). Let us consider the following relation: y is <u>derivable</u> from $\bar{y}$ if $\tilde{y}$, as defined by (9-3) satisfies all the constraints (9-7). It can be shown that this is in fact an equivalence relation, which partitions the set of feasible solutions into equivalence classes. Thus, whenever we obtain a feasible solution in a class, we deduce a best solution within this class by solving a minimum-weighted closure problem. Then we may discard this entire class of solutions by adding a disjunctive constraint:

$$\tilde{y}_j = 1 - \tilde{y}_k \quad \text{for at least one pair } j,k \text{ such that } j \neq k \text{ and } \tilde{Q}_{jk} < 0 \tag{9-10}$$

Empirical evaluations are necessary to check whether, in practical integer programming problems, the classes of derivable solutions usually contain more than a single solution each.

Of course, there are solvable instances of minimum cut problems with negative capacities, for instance the maximum cut problem in a planar graph, see[45] and[1] and another instance of the maximum cut problem considered in[70]: consider a complete undirected graph with a numbering of its vertex set such that the source is v_0, the sink v_{n+1}, as before, and now for every vertex v_i ($i = 0, 1, \ldots, n$) all the capacities c_{ij} are equal, for all j such that $i < j < n + 1$ (the sink-arc capacities may be arbitrary); the maximum cut problem can be solved, in such a network, by dynamic programming; as an application, consider the problem of nonpre-emptively sequencing jobs on two non-identical parallel machines to minimize the mean flow time subject to a given total order (see also[80] for a discussion of such a model which may occur in modelling a hospital emergency clinic): the problem reduces to the allocation of jobs to machines, since the ordering of jobs on every machine is thus imposed, and this allocation problem can be formulated as a maximum cut problem through the quadratic programming formulation (7-6).

Additional constraints

It is remarkable that very simple additional constraints may make the minimum cut problem NP-complete. Consider for instance the following cardinality-constrained minimum cut problem:

$$\text{minimize} \quad \{bx - x^T Cx : \sum_i x_i = k, \quad x \text{ is binary}\} \tag{9-11}$$

It is readily seen that we can solve the maximum cut problem by replacing the matrix C by M-C where M is a matrix with entries $m_{ij}=m \geq \max c_{ij}$, and solving problem (9-11) for every value of k = 0, 1, ..., n. Thus the cardinality-constrained minimum cut problem is NP-complete. Similarly, deciding whether an arbitrary graph contains a cut with exactly k arcs (instead of k vertices as in (9-11)) is also an NP-complete problem; for this would provide an effective means of solving the maximum cut problem.

As an example of a problem that can be formulated as a constrained minimum cut problem, consider the following quadratic assignment problem[73]: given n locations with distance matrix D (d_{pq} is the distance between location p and q), and n facilities with a traffic matrix T (t_{ij} is the traffic between facilities i and j), find a one-to-one assignment of facilities to locations which minimizes the total traval distance, i.e.

$$\text{minimize} \sum_i \sum_p \sum_j \sum_q t_{ij} d_{pq} x_{ip} x_{jq} \tag{9-12}$$

(for all $h \in V$) of its endpoints i and j; then G' contains a closure with exactly $nq + \frac{1}{2} q(q-1)$ vertices if and only if G contains a clique with q vertices.

An example of a problem amenable to a constrained maximum-weighted closure problem is the simple plant location problem: given m possible locations for new facilities, with a fixed cost f_i for constructing a facility in location i, given n customers, each of which to be assigned to exactly one facility, and a cost matrix C (c_{ij} is the cost of assigning customer j to a facility constructed in location i), find in which locations to construct facilities and how to assign the customers to them in order to minimize the resulting total cost. A standard integer programming formulation[26] is the following:

$$\text{minimize} \quad \sum_j \sum_i c_{ij} x_{ij} + \sum_i y_i \tag{9-16}$$

$$\text{subject to} \quad \sum_i x_{ij} = 1 \quad \text{all } j \tag{9-17}$$

$$x_{ij} \leq y_i \quad \text{all } i,j \tag{9-18}$$

$$x,y \text{ binary} \tag{9-19}$$

corresponding to negative capacities in the minimum cut problem). We seek a way to enforce the following constraints

$$\{i\} \in U \quad \text{for all} \quad i \in T \Rightarrow T \in U \tag{9-22}$$

We may use the substitution of some variables by their complement, as it was suggested by Hansen for the quadratic case, but we observe that turning a coefficient q_T positive may create several negative coefficients $q_{T'}$ with $T' \subseteq T$. It might be useful to characterize polynomials which are amenable to posynomial maximization, maybe by studying the "balancing" of "signed hypergraphs".

Another example of such a minimum cut approach, consider the vertex packing problem: in an undirected graph $G = (V,E)$ with vertex weights w_i, find a subset $P \subseteq V$ with maximum total weight, such that to two vertices of P are adjacent in G. (82) describes an interesting application to the reconstruction of level schemes, in nuclear physics: given the set of γ-ray transitions observed during the decay of a certain nucleus, their intensities and their coincidence, find a set of energy levels and the pattern constituted of energy levels and of transitions that agrees to the best with the data. This project resulted in the elaboration of an interactive software system named SIRENE which is being used by physicists at l'Institut des Sciences Nucléaires de Grenoble. A possible formulation of the vertex-packing problem is the following:

$$\text{maximize} \quad \sum_i w_i x_i \tag{9-23}$$

$$\text{subject to} \quad x_i x_j = 0 \qquad \text{all } \{i,j\} \in E \tag{9-24}$$

$$x \text{ is binary} \tag{9-25}$$

where $x_i = 1$ if $i \in P$, and
0 otherwise.

If G is bipartite, say $V = S + T$ and $E \subseteq S \times T$, we may replace every variable x_i for $i \in T$ by its complement $\bar{x}_i = 1 - x_i$, yielding constraints of type (7-9). Thus the vertex-packing problem in a bipartite graph can be solved as a maximum-weighted closure problem, i.e. as a minimum cut problem, a well-known fact (e.g.[33]). When the graph is not bipartite, we can replace every constraint (9-24) by the two constraints

$$x_i(1 - \bar{x}_j) = 0 \qquad \text{all } \{i,j\} \in E \tag{9-24a}$$

$$x_j(1 - \bar{x}_i) = 0 \qquad \text{all } \{i,j\} \in E \tag{9-24b}$$

provided that we append the $|V|$ additional constraints

$$\bar{x}_i = 1 - x_i \qquad \text{all } i \in V \tag{9-26}$$

When we relax those last constraints (9-26) we obtain a maximum-weighted closure problem in a bipartite graph twice the size of G. This relaxed formulation is attributed to Edmonds and Pulleyblank; see Nemhauser and Trotter[84] for a proof of the equivalence with a linear programming relaxation of another formulation of the vertex packing problem, and[89] and[91] for further properties of the resulting solution. Pulleyblank[100] has shown that "almost all" graphs (in a precise probabilistic sense) yield to a solution which violates all the constraints (9-26). The design of more efficient ways to enforce these constraints might deserve further investigation.

By considering the complementary graph $\overline{G} = (V,\overline{E})$ of a graph $G = (V,E)$, where $\overline{E} = K_{|V|} - E$ and $K_{|V|}$ is the complete graph on vertex set V (clique), the vertex-packing problem reduces to the maximum-weighted clique problem, i.e.: find a maximum-weighted subset $Q \subseteq V$ such that all vertices in Q are adjacent in $\overline{E}$. The maximum clique problem occurs when all weights are unity. The pseudoarboricity $p\alpha(\overline{G})$ provides an upper bound to the maximum cardinality $\gamma(\overline{G})$ (number of vertices) of a clique contained in $\overline{G}$, by the following relation:

$$\gamma(G) < 2p\alpha(\overline{G}) + 1 \qquad (9\text{-}27)$$

The authors have embodied this bound in a branch-and-bound algorithm for the maximum clique problem; they report in[92] experimental comparisons with a similar branch-and-bound algorithm using the bound derived from the

the formulation (9-23), (9-24a),(9-24b), (9-25); it appears that the former works best for "dense" graphs G (i.e. for finding a maximum clique in a "sparse" graph $\bar{G}$) while the latter is better for sparse graphs G. The idea used in the bound (9-27) can be refined as follows: assume that we enumerate the set $S_p(G)$ of all p-cliques (i.e. cliques with p vertices) contained in G, where p is a (small) fixed integer, and consider the following problem:

$$\text{maximize} \left\{ \frac{\sum_{K \in S_p(G)} \prod_{v_i \in K} x_i}{\sum_i x_i} : x \text{ is binary, } x \neq 0 \right\} \tag{9-28}$$

Let $r_p(G)$ denote the maximum value of this ratio for a given graph G. Any $\gamma(G)$-clique Q of G contains $B(\gamma(G),P) = \binom{\gamma(G)}{p}$ p-cliques and the value of the ratio for the solution defined by $x_i = 1$ if $v_i \in Q$, is $B(\gamma(G),p)/\gamma(G)$. Thus we have

$$B(\gamma(G),p)/\gamma(G) \leq r_p(G). \tag{2-29}$$

This equation can be used to derive an upper bound on $\gamma(G)$ for any $p \geq 2$ (note that by setting $p = 2$ we obtain precisely the bound (9-27).

* * *
* *

REFERENCES

(1) K. Aoshima and M. Iri, "Comments on F. Hadlock's paper: 'Finding a Maximum Cut of a Planar Graph in Polynomial Time'", SIAM J. Comput. 6, 1 (1977), 86-87.

(2) M.L. Balinski, "On a Selection Problem", Management Science 17, 3 (1970), 230-231.

(3) M.O. Ball, "Computing Network Reliability", Operations Research 27, 4, (1979), 823-838

(4) M.S. Bazaraa and J.J. Jarvis, Linear Programming and Network Flows, Wiley, New York (N.Y.), 1977.

(5) C. Berge, Graphes et hypergraphes, Dunod, Paris (France) 1970. (English translation: Graphs and Hypergraphs, North-Holland, Amsterdam (The Netherlands) 1973).

(6) C. Berge et A. Ghouila-Houri, Programmes, jeux et réseaux de transport, Dunod, Paris (France) 1962. (English translation: Programming, Games and Transportation Networks, Wiley, New-York (N.Y.) 1965.

(7) L.J. Billera and W.F. Lucas, "Delbert Ray Fulkerson: August 14, 1924 - January 10, 1976", Math. Programming Study 8, (1978), 1-16.

(8) F.T. Boesch (ed.), Large-Scale Networks: Theory and Design, IEEE Press, Piscataway (N.J.) 1976.

(9) J.H. Bookbinder and S.P. Sethi, "The Dynamic Transportation Problem: a Survey", working paper 77-12, Faculty of Management Studies, University of Toronto (1977).

(10) P. Bratley, M. Florian and P. Robillard, "Scheduling with Earliest Start and Due-Date Constraints", Nav. Res. Log. Q. 18 (1971), 511-519.

(11) J. Bruno and T. Gonzalez, "Scheduling Independent Tasks with Release Dates and Due-Dates on Parallel Machines", Technical report 213, Computer Science Department, Pennsylvania State University (1976).

(12) R.G. Busacker and T.L. Saaty, Finite Graphs and Networks, McGraw Hill, New York (N.Y.), 1964.

(13) A.V. Cabot, R.L. Francis and M.A. Stary, "A Network Flow Solution to a Rectilinear Distance Facility Location Problem", AIIE Trans. 2, (1970), 132-141.

(14) T.-Y. Cheung, "On the Minimum-Cost and Minimum-Cut Network Flow Approaches for Multifacility Location Problems", presented at Les Journées de l'Optimisation, Montreal, May 1977.

(15) N. Christofides, Graph Theory: an Algorithmic Approach, Academic Press, New York (N.Y.) 1975.

(16) N. Christofides, "Worst-Case Analysis of a New Heuristic for the Travelling Salesman Problem", Working Paper 62-75-76, GSIA, Cargenie-Mellon University, Pittsburgh (Pa.) 1976.

(17) V. Chvâtal, A Course in Linear Programming, Freeman, San Francisco (Ca.), to appear.

(18) G. Cornuéjols, M.L. Fisher and G.L. Nemhauser, "Location of Bank Accounts to Optimize Float: an Analytic Study of Exact and Approximate Algorithms", Management Sc. 23, 8 (1977), 789-810.

(19) W.H. Cunningham, "A Network Simplex Method", Mathematical Programming 11, (1976), 105-116.

(20) G.B. Dantzig and D.R. Fulkerson, "On the Max.-Flow Min.-Cut Theorem of Networks", Linear Inequalities and Related Systems, Annals of Mathematics Study 38, Princeton University Press (1956) 215-221.

(21) D.W. Davies and D.L.A. Barber, Communication Networks for Computers, Wiley, Chichester (G.-B.) 1973.

(22) V. Drezner and G.O. Wesolowsky, "A Comparison of the Cut and the Minimal Cost Flow Approaches to the Rectilinear Distance Facility Location Problem", unpublished manuscript (1978).

(23) J. Edmonds, "Minimum Partition of a Matroid into Independent Subsets", J. Res. NBS 69B (1965) 67-72.

(24) J. Edmonds and E.L. Johnson, "Matching, Euler Tours and the Chinese Postman", Mathematical Programming 5 (1973), 88-124.

(25) J. Edmonds and R.M. Karp, "Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems", J. ACM 19 (1972) 248-264 (also reprinted in[8]).

(26) M.A. Efroymson and T.L. Ray, "A Branch-and-Bound Algorithm for Plant Location", Operations Research 14 (1966), 361-368.

(27) S.E. Elmaghraby, Some Network Models in Management Science, Springer-Verlag, Berlin, 1970.

(28) S.E. Elmaghraby, Activity Networks: Project Planning and Control by Network Models, Wiley, New York (N.Y.) 1977.

(29) S.E. Elmaghraby and P.S. Pulat, "Optimal Project Compression with Due-Dated Events", Nav. Res. Log. Q. 26, 2 (1979) 331-348.

(30) C.O. Fong and M.R. Rao, "Accelerated Labelling Algorithms for the Maximal Flow Problem with Applications to Transportation and Assignment Problems", Working Paper 7222, GSM, University of Rochester (1972).

(31) L.R. Ford and D.R. Fulkerson, "Maximal Flow Through a Network", Canad. J. Math. 9 (1957), 210-218.

(32) L.R. Ford and D.R. Fulkerson, "Constructing Maximal Dynamic Flows from Static Flows", Operations Research 6 (1958), 419-433.

(33) L.R. Ford and D.R. Fulkerson, Flows in Networks, Princeton University Press, 1962.

(34) H. Frank and I.T. Frisch, Communication, Transmission and Transportation Networks, Addison-Wesley, Reading (Mass.) 1971.

(35) I.T. Frisch, "An Algorithm for Vertex Pair Connexticity, Intern. J. Control 6 (1967), 579-593.

(36) D.R. Fulkerson, "A Network Flow Computation for Project Cost Curves", Management Science 7 (1961), 167-178.

(37) D.R. Fulkerson and G.C. Harding, "Maximizing the Minimum Source-Sink Path Subject to a Budget Constraint", Mathematical Programming 13 (1977), 116-118.

(38) Z. Galil and A. Naamad, "Network Flow and Generelized Path Compression", Dept. of Mathematical Sciences, Tel Aviv University (Israel), 1979.

(39) R.S. Garfinkel and G.L. Nemhauser, Integer Programming, Wiley, New York (N.Y.) 1972.

(40) B. Golden, "A Problem in Network Interdiction", Nav. Res. Log. Q. 25, 4 (1978),711-713.

(41) D. Goldfarb and M.D. Grigoriadis, "An Efficient Steepest-Edge Algorithm for Maximum Flow Problems", presented at the Tenth International Symposium on Mathematical Programming, Montreal 1979.

(42) R.E. Gomory and T.C. Hu, "Multi-Terminal Network Flows", SIAM J. Appl. Math. 9 (1961), 551-570.

(43) R.E. Gomory and T.C. Hu, "An Application of Generalized Linear Programming to Network Flows", SIAM J. Appl. Math. 10 (1962) 260-283.

(44) G. Grätzer, Lattice Theory: First Concepts and Distributive Lattices, W.H. Freeman and Co., San Francisco (Ca.),1971.

(45) F. Hadlock, "Finding a Maximum Cut of a Planar Graph in Polynomial Time", SIAM J. Comp.4, 3 (1975) 221-225.

(46) J. Halpern, "A Generalized Dynamic Flow Problem", Networks 9, 2 (1979), 133-167.

(47) P.L. Hammer (Ivanescu), "Some Network Flow Problems Solved with Pseudo-Booleen Programming, Operations Research 13, (1965) 388-389.

(48) P.L. Hammer, E.L. Johnson and B. Korte (eds.) Discrete Optimization, Annals of Discrete Mathematics 4 and 5, North-Holland, Amsterdam, 1979.

(49) P. Hansen, "Methods of Nonlinear 0-1 Programming", in[48], vol.5, 53-70.

(50) F. Harary, R.Z. Norman and D. Cartwright, Structural Models: An Introduction to the Theory of Directed Graphs, Wiley, New York (N.Y.), 1965.

(51) A.J. Hoffman, "Ray Fulkerson's Contributions to Polyhedral Combinatories", Math. Programming Study 8, (1978), 17-23.

(52) W.A. Horn, "Some Simple Scheduling Algorithms", Nav. Res. Log. Q 21 (1974), 177-185.

(53) J.E. Hopcroft and R.M. Karp, "A $n^{5/2}$ Algorithm for Maximum Matchings in Bipartite Graphs", SIAM J. Comp. 2, (1973), 225-231.

(54) T.C. Hu, Integer Programming and Network Flows, Addison-Wesley, Reading (Mass.), 1970.

(55) M. Iri, Network Flows, Transportation and Scheduling, Academic Press, New York (N.Y.) 1969.

(56) J.R. Isbell and W.H. Marlow, "Attrition Games", Nav. Res. Log. Q.3 (1956) 71-93.

(57) A. Itai and Y. Shiloach , "Maximum Flow in Planar Networks", SIAM J. Comp. 8, 2 (1979), 135-150.

(58) J.J. Jarvis and S. Tufekci, "A Decomposition Algorithm for Locating Minimal Cuts in a Directed Network", School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, (Ga.) 1977.

(59) E.L. Johnson, "Networks and Basic Solutions", Operations Research 14, (1966), 619-623.

(60) T.B. Johnson, "Optimum Open Pit Mine Production Scheduling", Operations Research Center, University of California at Berkeley, May 1968.

(61) T.B. Johnson, "A Comparative Study of Methods for Determining Ultimate Open Pit Mining Limits", presented at 11th Annual Symposium on Computer Applications in the Mineral Industry, Tucson, April 1973.

(62) R.M. Karp, "Reducibility Among Combinatorial Problems", in R. Miller and J. Thatcher (eds.), Complexity of Computer Computations, Plenum Press, New York (N.Y.) 1972.

(63) R.M. Karp, "The Probabilistic Analysis of Combinatorial Optimization Algorithms", presented at the Tenth International Symposium on Mathematical Programming, Montreal, August 1979.

(64) A. Kaufman, Dynamic Programming and Finite Games, Academic Press, New York (N.Y.), 1967.

(65) J.E. Kelley, Jr., "Critical Path Planning and Scheduling: Mathematical Basis", Operations Research 9 (1961), 296-320.

(66) B.W. Kernighan and S. Lin, "An Efficient Heuristic Procedure for Partitioning Graphs", Bell Syst. Tech. J. 49 (1970) 291-307 (Also reprinted in[8]).

(67) A. Koolen, "On 'A Cut Approach to the Rectilinear Distance Facility Location Problem' by J.-C. Picard and H.D. Ratliff", report BW 103/79, Stichting Mathematisch Centrum, Amsterdam (The Netherlands), April 1979.

(68) J. Krarup and P.M. Pruzan, "Selected Families of Location Problems", in[48], vol. 5, 327-387.

(69) J. Labetoulle, E.L. Lawler, J.K. Lenstra and A.H.G. Rinnooy Kan, "Preemptive Scheduling of Uniform Machines Subject to Release Dates", report BW 99/79, Stichting Mathematisch Centrum, Amsterdam, (The Netherlands), 1979.

(70) S. Lakshminarayan, R. Lakshmanan, R.L. Papineau and R. Rochette, "Order Preserving Allocation of Jobs to Two Non-Identical Parallel Machines: a Solvable Case of the Maximum Cut Problem", INFOR 17, 3 (1979), 230-241.

(71) A. Langevin, "Etude d'algorithmes pour le problème du flot maximum", M.Sc. Thesis, Département de Génie Industriel, Ecole Polytechnique de Montreal, April 1979.

(72) E.L. Lawler, Combinatorial Optimization: Networks and Matroids, Holt Rinehart and Winston, New York (N.Y.), 1976.

(73) E.L. Lawler, "The Quadratic Assignment Problem: a Brief Review", in B. Roy (ed.) Combinatorial Programming: Methods and Applications, Reidel, Boston (Mass.) 1965.

(74) E.L. Lawler, "Sequencing Jobs to Minimize Total Weighted Completion Time Subject to Precedence Constraints", Annals of Discrete Mathematics 2 (1978) 75-90.

(75) E.L. Lawler, "Shortest Path and Network Flow Algorithms", in[48], vol. 4, 251-263.

(76) H. Lerchs and I.F. Grossman, "Optimum Design of Open Pit Mines", The Canadian Mining and Metallurgical Bulletin 58 (1965) 47-54.

(77) R.F. Love and L. Yerex, "An Application of a Facilities Location Model in the Prestressed Concrete Industry, Interfaces 6, 4 (1976) 45-49.

(78) L.F. McGinnis and H.L.W. Nuttle, "The Project Coordinator's Problem", Omega 6, 4 (1978) 325-330.

(79) N. Megiddo and Z. Galil, "On Fulkerson's Conjecture about Consistent Labelling Processes", Math. of Op. Res. 4, 3 (1979), 265-267.

(80) S. Mehta, R. Chandrasekaran and H. Emmons, "Order Preserving Allocation of Jobs to Two Machines", Nav. Res. Log. Q 21 (1974) 361-364.

(81) E. Minieka, Optimization Algorithms for Networks and Graphs, Marcel Dekker, New York (N.Y.) 1978.

(82) C. Mukendi wa Maloji and J.P. Uhry, "Méthodologie combinatoire pour la reconstitution des schémas de niveau d'énergie", rapport de recherche 106, Mathématiques Appliquées et Informatique, Université de Grenoble (France) 1978.

(83) C. St. J.A. Nash-Williams, "Edge-Disjoint Spanning Trees of Finite Graphs", J. London. Math. Soc. 36 (1961), 445-450.

(84) G.L. Nemhauser and L.E. Trotter, "Vertex Packings: Structural Properties and Algorithms", Math Programming 8 (1975), 232-248.

(85) A. Nijenhuis and H.S. Wilf, Combinatorial Algorithms, 2nd edition, Academic Press, New York (N.Y.) 1978.

(86) S. Phillips Jr. and M.I. Dessouky, "Solving the Project Time/Cost Tradeoff Problem Using the Minimal Cut Concept", Management Science 24 (1977), 393-400.

(87) S. Phillips Jr. and M.I. Dessouky, "The Cut-Search Algorithm with Arc Capacities and Lower Bounds", Management Science 25 (1979), 396-404.

(88) J.C. Picard, "Maximum Closure of a Graph and Application to Combinatorial Problems", Management Science 22 (1976) 1268-1272.

(89) J.C. Picard and M. Queyranne, "Vertex-Packings: (VLP)-Reductions Through Alternate Labelling", Techn. Rep. EP75-R-47, Ecole Polytechnique de Montréal, Canada (September 1975).

(90) J.C. Picard and M. Queyranne, "Simple Validation of Maximal Closure of a Graph", Techn. Rep. EP75-R-60, Ecole Polytechnique de Montréal, Canada (November 1975).

(91) J.C. Picard and M. Queyranne, "On the Integer-Valued Variables in the Linear Vertex Packing Problem", Mathematical Programming 12 (1977), 97-101.

(92) J.C. Picard and M. Queyranne, "Networks, Graphs and Some Non-Linear 0-1 Programming Problems", Techn. Rep. EP77-R-32, Ecole Polytechnique de Montreal, Canada, (1977).

(93) J.C. Picard and M. Queyranne, "A Network Flow Solution of Some Non-Linear 01 Programming Problems and Applications to Graph Theory", Techn. Rep. EP79-R-14, Ecole Polytechnique de Montreal, Canada, April 1979.

(94) J.C. Picard and M. Queyranne, "On the Structure of All Minimum Cuts in a Network and Applications", Techn. Rep. EP79-R-15, Ecole Polytechnique de Montreal, Canada, March 1979 (to appear in V.J. Rayward-Smith (ed.) CO79 Proceedings).

(95) J.C. Picard and H.D. Ratliff, "A Graph Theoretic Equivalence for Integer Programs", Operations Research 21, 2 (1973), 261-269.

(96) J.C. Picard and H.D. Ratliff, "Minimal Cost Cut Equivalent Networks", Management Science 19, 9 (1973), 1087-1092.

(97) J.C. Picard and H.D. Ratliff, "Minimum Cuts and Related Problems", Networks 5, 4 (1975), 357-370.

(98) J.C. Picard and H.D. Ratliff, "A Cut Approach to the Rectilinear Distance Facility Location Problem", Operations Research 26 (1978), 422-433.

(99) J.C. Picard and H.D. Ratliff, "A Cut Approach to a Class of Quadratic Integer Programming Problems", Research Report, Department of Industrial and Systems Engineering, University of Florida at Gainesville, December 1977.

(100) R.W. Pulleyblank, "Minimum Node Covers and 2-Bicritical Graphs", Mathematical Programming 17 (1979), 91-103.

(101) M. Queyranne, "Theoretical Efficiency of the Algorithm Capacity for the Maximum Flow Problem", to appear in Math. of Op. Res. (1980).

(102) J.K. Reid, "Solution of Linear Systems of Equations: Direct Methods (general)", in V.A. Barker (ed.), Sparse Matrix Techniques, Springer-Verlag, Berlin (Germany) 1977, 119-130.

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

(104) M. Richetin, "Algorithme de décomposition optimale et sous-optimale des graphes", notes internes CH-LAAS73133, LAAS, Toulouse (France), 1973.

(105) B. Roy, Algèbre moderne et théorie des graphes, 2 volumes, Dunod, Paris (France), 1970.

(106) C.P. Schnorr, "Bottlenecks and Edge Connectivity in Unsymmetrical Networks", SIAM J. Comput. 8, 2 (1979), 265-274.

(107) Y. Shiloach, "Multi-terminal 0-1 Flow", SIAM J. Comput. 8, 3 (1979), 422-430.

(108) J.B. Sidney, "Decomposition Algorithms for Single-Machine Sequencing with Precedence Relations and Deferral Costs", Operations Research 22 (1975), 283-298.

(109) R.E. Tarjan, "Testing Graph Connectivity", Proc. 6th Annual Symp. on Theory of Comput. (1974), 185-193 (also reprinted in[8]).

(110) R.E. Tarjan and S. Even, "Network Flow and Testing Graph Connectivity", SIAM J. Comput. 4, 4 (1975), 507-518.

(111) D.M. Topkis, "Monotone Minimum Node-Cuts in Capacitated Networks", Research report ORC70-39, Operations Research Center, University of California at Berkeley, December 1970.

(112) L.G. Valiant, "The Complexity of Enumeration and Reliability Problems", SIAM J. Comput. 8, 3 (1979) 410-421.

(113) H.M. Weingartner, "Capital Budgeting of Interrelated Projects: Survey and Synthesis", Management Science 12, 7 (1966), 485-516.

(114) G.O. Wesolowsky and R.F. Love, "The Optimal Location of New Facilities Using Rectilinear Distances", Operations Research 19 (1971), 124-130.

(115) R.S. Wilkov, "Analysis and Design of Reliable Computer Networks", IEEE Trans. Commun. 20 (1972), 660-678 (also reprinted in[8]).

* * * * * * * * * *

A CONSULTER
SUR PLACE