SCHOOL OF OPERATIONS RESEARCH
AND INDUSTRIAL ENGINEERING
COLLEGE OF ENGINEERING
CORNELL UNIVERSITY
ITHACA, NEW YORK 14853

TECHNICAL REPORT NO. 607

September 1983
(Revised March 1984)
(Revised April, 1987)

PACKING AND COVERING WITH INTEGRAL FEASIBLE
FLOWS IN INTEGRAL SUPPLY-DEMAND NETWORKS

By

R.E. Bixby⁺
O.M.-C. Marcotte⁺⁺
L.E. Trotter, Jr.⁺⁺⁺

⁺ Department of Mathematical Sciences, Rice University, Houston, Texas, U.S.A.
Research partially supported by NSF grant MCS81-05Z327 at Northwestern University, and by the Alexander von Humboldt Foundation and the Institut für Ökonometrie und Operations Research der Universität Bonn (Federal Republic of Germany), while the author was on leave from Rice University.

⁺⁺ Département de mathématiques et d'informatique, Université du Québec à Montréal, Montréal, Canada. Research partially supported by NSERC grant A9126 and by NSF grants ECS-8005350 and ECS-8113534 to Cornell University.

⁺⁺⁺ School of OR&IE, Cornell University, Ithaca, N.Y., U.S.A.
Research partially supported by NSF grants ECS-8113534 and ECS-8504077.

Short title: Integral Flow Packings and Coverings

Key words: Network flows, covering and packing, polynomial-time algorithm

Abstract

Polynomial-time algorithms are presented for solving combinatorial packing and covering problems defined from the integral feasible flows in an integral supply-demand network. These algorithms are also shown to apply to packing and covering problems defined by the minimal integral solutions to general totally unimodular systems. Analogous problems arising from matroid bases are also discussed and it is demonstrated that a means for solving such problems is provided by recent work of Cunningham.

1. Introduction

Suppose $G \equiv (V, E, a, c)$ is a (capacitated) supply-demand network. That is, suppose the vertices (nodes) V of G are partitioned into subsets S (source nodes), T (sink nodes) and R (intermediate nodes) and that to each $v \in V$ there corresponds a net supply a_v , where $a_v > 0$ for $v \in S$, $a_v < 0$ for $v \in T$ and $a_v = 0$ for $v \in R$, and finally, that to each edge (arc) $e \in E$ there corresponds a capacity $c_e \geq 0$. A feasible flow in G is a vector $(x_e: e \in E)$ that satisfies

$$x(\delta^+(v)) - x(\delta^-(v)) = a_v, \quad v \in V,$$

$$0 \leq x_e \leq c_e, \quad e \in E,$$

where we define $\delta^+(v) = \{(v, u) \in E\}$, $\delta^-(v) = \{(u, v) \in E\}$, and for $F \subseteq E$, $x(F) = \sum(x(e): e \in F)$.

When vectors $(a_v: v \in V)$ and $(c_e: e \in E)$ have integral components, the supply-demand network is termed integral; similarly, any feasible flow $(x_e: e \in E)$ with integral components is called an integral feasible flow. For a given supply-demand network the number of distinct integral feasible flows is obviously finite. Thus we can consider a matrix A whose columns correspond to the edges of G and whose rows correspond to the integral feasible flows in G indexed $i = 1, 2, \dots, m$. That is, the entry a_{ie} of matrix A is the value assigned to $e \in E$ by the i th integral feasible flow.

For matrix A defined as above and any nonnegative integral vector $(w_e: e \in E)$, we consider two combinatorial optimization problems ($\mathbf{1}$ denotes the vector $(1, \dots, 1)$, each of whose components has value one):

Packing problem $P(A, w)$

$$\begin{aligned} \max \quad & \mathbf{1} \cdot y \\ \text{s.t.} \quad & yA \leq w \\ & y \geq 0, \text{ integral} \end{aligned}$$

Covering problem $C(A, w)$

$$\begin{aligned} \min \quad & \mathbf{1} \cdot y \\ \text{s.t.} \quad & yA \geq w \\ & y \geq 0, \text{ integral.} \end{aligned}$$

These problems have been studied extensively in [9,15,17]. For the remainder of this paper we shall focus attention on the packing problem $P(A,w)$. Results for $C(A,w)$ follow analogously, and we comment briefly on this at several points throughout the paper.

The optimum value (scalar) for $P(A,w)$ can be computed in time polynomial in the size of G and w . This was pointed out by Weinberger [17] who posed the question of whether an optimum solution vector for $P(A,w)$ could also be determined in polynomial time, again measured as a function of the input length of G and w . Note that in general the number of rows of matrix A is an exponential function of the input length of G and, further, that one does not know a priori whether $P(A,w)$ has an optimum solution vector with only "polynomially many" nonzero components. In Section 2 below we present polynomial-time algorithms for solving $P(A,w)$.

In the final section of this paper we show how the algorithms of Section 2 can be adapted to solve packing and covering problems defined by integral solutions to general totally unimodular systems and how recent results of Cunningham [4, 5] can be used to solve matroid basis packing and covering problems.

2. Efficient solution of $P(A,w)$, $C(A,w)$

First consider the optimum value of $P(A,w)$. For any positive integer k , we denote by kG the capacitated supply-demand network (V,E,ka,kc) . Let y be any feasible solution vector for $P(A,w)$; i.e., $yA \leq w$ with $y \geq 0$ and y integral, and suppose $k = 1 \cdot y$. Since $1 \cdot (y/k) = 1$, the vector $(y/k)A$ is a convex combination of feasible flows in G and thus yA is an integral feasible flow in kG satisfying $yA \leq w$. Furthermore, it was shown in [9] that any integral feasible flow in kG can be "decomposed" into a sum of k

vectors, each an integral feasible flow of G . Thus an integral feasible flow in kG corresponds in a natural way to a feasible (integral) solution of value k for $P(A,w)$. It thus follows that determining the optimum solution value for $P(A,w)$ is equivalent to determining the largest integer k^* for which there exists an integral feasible flow, say x , in k^*G satisfying $x \leq w$.

The above remarks suggest, as observed in [17], that the value k^* of an optimum solution for $P(A,w)$ can be determined by using binary search on the interval $[0,K]$, where, for example,

$$K \equiv \min \{ \lfloor w(\delta^+(v))/a_v \rfloor : a_v > 0 \} + 1$$

is a strict upper bound on k^* . The typical step in such a procedure would be validation for a specific choice of $k \in [0,K]$ of whether there exists in kG an integral feasible flow x with $x \leq w$. Such an x could be determined (or proved not to exist) by solving a network flow problem; since $k \leq K$, this network flow problem can be solved in time bounded by a polynomial function of the length of the description of G and w . Moreover, since this step would be executed no more than $\lceil \log_2 K \rceil$ times during binary search of the interval $[0,K]$, it follows that k^* would be determined in polynomial time by such a procedure.

Other means are available for determining k^* in polynomial time. For example, the problem $P(A,w)$ has the "integer rounding property" (see [9]), and thus if its linear programming relaxation has value r^* , then $k^* = \lfloor r^* \rfloor$. Now r^* can be computed in polynomial time, since in the dual of the linear programming relaxation of $P(A,w)$, violated constraints can be identified using a minimum cost flow algorithm. Thus the ellipsoid method (see [10]) can be applied to determine r^* without the explicit construction of matrix A . Alternatively, in [12] Padberg and Wolsey suggest a procedure, based on a general algorithm of [10], for solving the linear relaxation of $C(A,w)$; this approach should also apply to $P(A,w)$. Indeed, the exact analogue of

our situation is studied on page 13 of Padberg and Wolsey [12]. We remark, however, that Algorithm 2 of [12] is no longer strongly polynomial in this application, for essentially the same reason that Algorithm 2* of [12] is not strongly polynomial (for the definition of a strongly polynomial-time algorithm, see Cunningham [5]).

One final approach, very closely related to that in [12], is suggested by Hoffman's feasible circulation theorem, which asserts that the optimal value of $P(A,w)$ (denoted by r^* as above) is equal to

$$\min \left\{ \frac{w(U, \bar{U})}{a(U)} : U \subseteq V \text{ and } a(U) > 0 \right\}, \text{ where}$$

$$\bar{U} = V \setminus U, w(U, \bar{U}) = \sum \{w_e : e = (u,v), u \in U \text{ and } v \in \bar{U}\} \text{ and } a(U) = \sum \{a_u : u \in U\}.$$

This type of problem has been studied by Cunningham [5]. In particular, he gives a strongly polynomial-time algorithm for problems of the form

$$\min \left\{ \frac{f(U)}{x(U)} : U \subseteq V \text{ and } x(U) > 0 \right\},$$

where f is a "submodular" function on subsets of V , and x is a nonnegative vector on V (see the development on page 559 in [5]). It is easy to verify that $w(U, \bar{U})$ as defined above is submodular (as a function of U); however, a is never nonnegative (unless it is identically 0). Moreover, the latter property seems to be crucial to the proof that Cunningham's algorithm is a polynomial-time algorithm. It is thus unlikely that Cunningham's result will yield a polynomial-time (let alone strongly polynomial-time) algorithm in our case.

We now turn to algorithms for computing an optimal solution of $P(A,w)$ (as opposed to the optimal solution value of $P(A,w)$). By the

preceding discussion, we may assume that k^* and an integral feasible flow $x \leq w$ in k^*G are known. We describe below Algorithm DECOMP for decomposing such an x into k^* integral feasible flows in G , i.e., for determining integral feasible flows x^i , $1 \leq i \leq k^*$, in G such that $x = x^1 + \dots + x^{k^*}$. It is easy to see that any such algorithm will provide means for solving $P(A, w)$. Note, however, that such a solution procedure must be capable of expressing the decomposition $x = x^1 + \dots + x^{k^*}$ in time polynomial in the size of G and w . Loosely speaking, it is $\lceil \log_2 k^* \rceil$ that is known to be bounded by a polynomial function of this input size, not k^* itself; thus an explicit list of the flows $x^1, \dots, x^{k^*}$ in an arbitrary decomposition of x may be prohibitively long. Algorithm DECOMP below determines a particular decomposition of x that can be expressed in polynomial time.

A final preliminary observation is that the decomposition problem stated in the previous paragraph can be reformulated in terms of uncapacitated supply-demand networks (i.e., without $x_e \leq c_e$ restrictions) using a well-known technique due to Wagner (see [16] and page 128 of [8]). Details of this polynomial-time reformulation are given in [11]. Thus in the following we consider the uncapacitated case only.

Algorithm DECOMP:

Input: An uncapacitated supply-demand network $G = (V, E, a)$ and an integral feasible flow x in $kG = (V, E, ka)$ for some positive integer k .

Output: A decomposition of x in the form of a set $D = \{(n_z, z)\}$, where the n_z are nonnegative integers such that $\sum n_z = k$ and the z are integral feasible flows for G satisfying

$$x = \sum \{n_z \cdot z : (n_z, z) \in D\}.$$

- Step 0: Set $\ell \leftarrow k$, $D \leftarrow \emptyset$.
- Step 1: Find an integral feasible flow z in G such that $z \leq x$ and find the largest integer $p \geq 1$ such that $p \leq k$ and $x - pz \geq 0$.
- Step 2: Set $D \leftarrow D \cup \{(p, z)\}$, $x \leftarrow x - pz$, $\ell \leftarrow \ell - p$.
- Step 3: If $\ell = 0$, stop;
 else if for some $e \in E$, $x_e = 0 < z_e$, go to Step 1;
 else determine an integral feasible flow z' in G such that $z' \leq x$ and $z'_e = x_e < z_e$ for some $e \in E$;
 set $D \leftarrow D \cup \{(1, z')\}$, $x \leftarrow x - z'$, $\ell \leftarrow \ell - 1$;
 if $\ell = 0$, stop; else go to Step 1. □

Note that upon termination of Algorithm DECOMP, the set D clearly does satisfy the output requirements of the algorithm. Thus to show the validity of the algorithm, we must show that in Steps 1 and 3 the appropriate integral feasible flows z , z' can always be computed. The integral feasible flow z required in Step 1 must satisfy the relations

$$\begin{aligned} z(\delta^+(v)) - z(\delta^-(v)) &= a_v, \quad v \in V, \\ 0 \leq z_e &\leq x_e, \quad e \in E. \end{aligned}$$

Note that the constraint matrix of this system is totally unimodular (see [13]), that all a_v , x_e are integer-valued and that the system is feasible (e.g., take $z = (1/\ell)x$). It follows that there exists an integral solution to the system, thus establishing the existence of the integral feasible flow z for Step 1 of the algorithm. The determination of z by a network-flow algorithm requires only time polynomial in the length required to describe G and x ; actually, it requires time polynomial in $|V|$, provided that each arithmetic operation (such as addition, subtraction and comparison) is considered to require one unit of time.

We observe that the redefinition of x in Steps 2 and 3 of the algorithm insures that whenever Step 3 is encountered, x is an integral feasible flow in $\mathcal{L}G$ and thus the vector $y \equiv (1/\mathcal{L})x$ is a feasible flow in G . Furthermore, the execution of Step 3 must be preceded by that of Step 1, and thus whenever Step 3 is encountered, a feasible flow z in G is known for which $\hat{E} \equiv \{e \in E: x_e < z_e\} \neq \emptyset$. Define $\lambda = \max \{(z_e - x_e)/(z_e - y_e): e \in \hat{E}\}$. It is straightforward to verify that:

- (a) $0 \leq \lambda \leq 1$;
- (b) $\lambda y + (1-\lambda)z$ is a feasible flow in G ;
- (c) $\lambda y + (1-\lambda)z \leq x$;
- (d) $\lambda y_e + (1-\lambda)z_e = x_e$ for $e \in \tilde{E} \equiv \{e \in \hat{E}: \lambda = (z_e - x_e)/(z_e - y_e)\}$.

From (a)–(d) we conclude that there exists a vector z' that satisfies the relations

$$\begin{aligned} z'(\delta^+(v)) - z'(\delta^-(v)) &= a_v, \quad v \in V, \\ 0 \leq z'_e &\leq x_e, \quad e \in E, \\ z'_e &= x_e, \quad e \in \tilde{E}. \end{aligned}$$

Again the constraint matrix of this system is totally unimodular and its right-hand side is integer-valued. Thus there is an integral solution z' to this system, as required in Step 3 of the algorithm, and z' can be determined in time polynomial in $|V|$.

Thus the steps of Algorithm DECOMP are well-defined and each step can be executed in polynomial time. Furthermore, between two successive iterations of Step 1, x_e becomes 0 for at least one additional $e \in E$. Hence Step 1 is executed at most $|E|$ times, verifying the following

Theorem: Algorithm DECOMP can be used to solve $P(A, w)$ in time polynomial in $|V|$, where V is the vertex set of G . $\square$

Algorithm DECOMP can be simplified by replacing Step 3 with the following:

Step 3: If $z = 0$, stop; else go to Step 1.

This version remains polynomially time bounded, for whenever Step 1 is executed, $x_e - (p+1)z_e < 0$ for some $e \in E$. But then

$$x_e - pz_e < x_e - px_e/(p+1) \leq x_e/2,$$

which implies that between two iterations of Step 1 some x_e is reduced by a factor of at least $1/2$. Thus decomposition of the original vector x requires solving at most $\sum (\lceil \log_2 x_e \rceil + 1 : x_e > 0)$ network flow problems in this version of the algorithm, whereas it required at most $2|\{e : x_e > 0\}|$ in the earlier version. Observe that polynomial-time algorithms used to solve maximum-flow problems are in general strongly polynomial. If such an algorithm is used in Steps 1 and 3 of Algorithm DECOMP, the first version of DECOMP becomes strongly polynomial itself. The second version, however, does not.

As remarked earlier, the covering problem $C(A, w)$ can be solved similarly. From [17] it follows that the optimum value for $C(A, w)$ is given by the least integer k_* for which there exists an integral feasible flow, say x , in k_*G satisfying $x \geq w$. The value k_* and the flow x can again be computed in polynomial time by using a binary search algorithm. After conversion to a flow in the associated uncapacitated network, this flow can be decomposed using Algorithm DECOMP. The resulting decomposition, when re-expressed for the original network, yields an optimal solution of $C(A, w)$.

Fulkerson and Weinberger [9] have also studied the packing problem

$P(A,w)$ for general uncapacitated integral supply-demand networks. For such networks the rows of matrix A correspond to minimal integral feasible flows. (An integral feasible flow x is minimal provided no integral feasible flow x' satisfies $x' \leq x$, $x' \neq x$.) As in the capacitated case, one can determine in polynomial time the optimum value k^* of $P(A,w)$ and an integral feasible flow x in k^*G that satisfies $x \leq w$. Algorithm DECOMP can then be used directly to decompose x into a sum of k^* integral feasible flows for G . To force minimality of these flows and hence determine a solution for $P(A,w)$, one must eliminate positive flow around cycles (if any is present) in the flows produced by DECOMP. Since this can be done in polynomial time, Algorithm DECOMP is again the basis for a polynomial-time procedure to solve $P(A,w)$ in the general uncapacitated case.

3. Related applications

Suppose N is a totally unimodular $m \times n$ matrix, b is an integral m -vector and A is the matrix given by the minimal integral solutions of the system $\{Nx = b, x \geq 0\}$. Since "integral decomposition" holds for such systems (see [15]), the packing problem $P(A,w)$ can be solved by the method described in the previous section. First use binary search to determine a (minimal) nonnegative integral vector x such that $Nx = k^*b$ and $x \leq w$, then use Algorithm DECOMP to decompose x into a sum of k^* rows of matrix A . In the binary search procedure and in Steps 1 and 3 of DECOMP, we require a polynomial-time algorithm for finding an integral solution of $\{N'x = d, \ell \leq x \leq u\}$, where N' is totally unimodular and d, ℓ, u are integral. Such an algorithm has been reported by Bland and Edmonds [2], and can be deduced from the characterization of totally unimodular matrices given by Seymour [14]. Since this algorithm, when used as a subroutine of the binary search procedure, can be made to produce a minimal nonnegative integral x satisfying $Nx = k^*b$ and $x \leq w$, it follows

that the terms in the decomposition of x given by DECOMP will be minimal integral vectors satisfying $Nx = b$ and $x \geq 0$, i.e., rows of matrix A .

Here again the capacitated version of the problem can be reduced directly to the form above. That is, when the rows of matrix A are the integral (minimality no longer being important) solutions of $\{Nx=b, 0 \leq x \leq c\}$ for integral c , we can consider the system $\{Nx = b, x+y = c, x \geq 0, y \geq 0\}$ which is in the form treated above. Indeed, the interested reader may readily verify that this transformation is equivalent to that of Wagner [16] for supply-demand systems (see Section 2 above). We also note that our observations here apply as well to covering problems $C(A,w)$.

As a further example of a packing problem, consider $P(A,w)$ in the setting where the rows of A are the incidence vectors of bases of a matroid M defined on $E \equiv \{1,2,\dots,n\}$. Here again $P(A,w)$ has the "integer rounding property" (see [1]), and so $k^* = \lfloor r^* \rfloor$ is the optimum value of $P(A,w)$, where r^* denotes the optimum value of the linear programming relaxation. Thus in order to determine k^* , it suffices to compute r^* by the ellipsoid method [10], using an independence oracle for matroid M in conjunction with the matroid greedy algorithm to recognize violated constraints in the linear programming dual of the relaxation of $P(A,w)$. In order to find an optimal solution of $P(A,w)$, however, we need to compute a vector $x \leq w$ in k^*G and to apply Algorithm DECOMP to x . It is not clear how to apply Algorithm DECOMP in this case; in particular, there is no guarantee that the vector $x-pz$ obtained in Step 1 of Algorithm DECOMP will be a linear combination of incidence vectors of bases. As for computing x , Cunningham [4,5] has given an algorithm which may be used to compute x and an optimal solution of $P(A,w)$. This algorithm, described below, is likely to be more efficient than Algorithm DECOMP.

In [4] a (strongly) polynomial-time algorithm is presented that, for any given vector $u \geq 0$, determines a vector x that maximizes

$x(E) \equiv \sum(x_j: j \in E)$ subject to $x \leq u$ and $x \in P(M)$, where $P(M)$ (the "matroid polyhedron" of M) denotes the convex hull of the incidence vectors of independent sets in M . On the other hand, the value of r^* is equal to

$$\min \left\{ \frac{w(E \setminus T)}{r(E) - r(T)} : T \subseteq E, r(T) < r(E) \right\}$$

by Edmonds' matroid partition theorem (see Edmonds [6]). The results of [5] then show that at most n applications of the algorithm presented in [4] suffice to determine the value of r^* .

An optimum integral packing vector y achieving value k^* may be determined as follows. Suppose $b^* = \sum(\lambda_i b^i: i \in I)$, where each b^i is the incidence vector of an M -base, $b^* \leq w$ and $\sum(\lambda_i: i \in I) = k^*$; note b^* may be obtained by "scaling" the solution vector for the linear programming relaxation of $P(A, w)$ by the factor k^*/r^* . Then clearly

$$b^*/k^* = \sum(\lambda_i b^i/k^*: i \in I) \equiv z \in P(M).$$

Now Cunningham's algorithm of [4] may be used to determine a vector x maximizing $x(E)$ subject to $x \leq w/k^*$ and $x \in P(M)$; this algorithm represents x in the form $x = \sum(\beta_j f^j: j \in J)$, where $\sum(\beta_j: j \in J) = 1$, $\beta_j \geq 0$ for $j \in J$ and each f^j is the incidence vector of an independent set of M . Now since $z \in P(M)$, $z \leq w/k^*$ and $z(E) = r(E)$ (where $r(E)$ is the rank of M), it follows that the $x \in P(M)$ from Cunningham's algorithm must satisfy

$$r(E) = x(E) = \sum(\beta_j f^j(E): j \in J);$$

hence each f^j corresponds, in fact, to a base of M . Moreover, Cunningham's algorithm specifies each β_j as an integral combination of the coordinates of w/k^* . Thus $k^*x = \sum(k^*\beta_j f^j: j \in J)$ is the desired packing and, in particular, $y_j = k^*\beta_j$ for $j \in J$ and $y_j = 0$ otherwise, gives an integral solution to $P(A, w)$. We note also that Cunningham's algorithm does not

require b^* (see above) for the computation of x .

Observe that $P(A,w)$ can also be solved directly, without the need for solving any linear programming problems, by using Edmonds' matroid partitioning algorithm [6] with "scaling" (see [7]), in the same way that it can be used to determine whether any given rational vector $x \geq 0$ is in $P(M)$ (see [4]). A theorem of Schrijver, Theorem 7.1 of [4], also deals with the problem of packing matroid bases. It follows from this result that $|J| \leq 2n$ can be achieved above; in [3] this bound is reduced to $2n-2$. Finally, a similar analysis to that outlined above can be used for covering problems of the form $C(A,w)$.

References

- [1] S. Baum and L.E. Trotter, Jr., "Integer rounding for polymatroid and branching optimization problems", S.I.A.M. Journal on Algebraic and Discrete Methods 2 (1981) 416-425.
- [2] R.G. Bland and J. Edmonds, "Fast primal algorithms for totally unimodular linear programming", presented at XIth International Symposium on Mathematical Programming, Bonn, West Germany (August 1982).
- [3] W. Cook, J. Fonlupt and A. Schrijver, "An integer analogue of Carathéodory's Theorem", to appear.
- [4] W. Cunningham, "Testing membership in matroid polyhedra", Journal of Combinatorial Theory (B) 32 (1984) 161-187.
- [5] W.H. Cunningham, "Optimal attack and reinforcement of a network", Journal of the A.C.M. 32 (1985) 549-561.
- [6] J. Edmonds, "Minimum partition of a matroid into independent subsets", Journal of Research of the National Bureau of Standards 69B (1965) 67-72.
- [7] J. Edmonds and R.M. Karp, "Theoretical improvements in algorithmic efficiency for network flow problems", Journal of the A.C.M. 19 (1972) 248-264.
- [8] L.R. Ford, Jr. and D.R. Fulkerson, "Flows in Networks", Princeton University Press (Princeton, New Jersey, 1962).

- [9] D.R. Fulkerson and D.B. Weinberger, "Blocking pairs of polyhedra arising from network flows", Journal of Combinatorial Theory (B) 18 (1975) 265-283.
- [10] M. Grötschel, L. Lovász and A. Schrijver, "The ellipsoid method and its consequences in combinatorial optimization", Combinatorica 1 (1981) 169-197.
- [11] O. Marcotte, "Topics in combinatorial packing and covering", Ph. D. Dissertation, Technical Report No. 568, School of Operations Research and Industrial Engineering, Cornell University (Ithaca, N.Y., 1982).
- [12] M.W. Padberg and L.A. Wolsey, "Fractional covers for forests and matchings", Mathematical Programming 29 (1984) 1-14.
- [13] C.H. Papadimitriou and K. Steiglitz, "Combinatorial Optimization: Algorithms and Complexity", Prentice Hall (Englewood Cliffs, New Jersey, 1982).
- [14] P.D. Seymour, "Decomposition of regular matroids", Journal of Combinatorial Theory (B) 28 (1980) 305-359.
- [15] L.E. Trotter, Jr. and D.B. Weinberger, "Symmetric blocking and anti-blocking relations for generalized circulations", Mathematical Programming Study 8 (1978) 141-158.
- [16] H.M. Wagner, "On a class of capacitated transportation problems", Management Science 5 (1959) 304-318.
- [17] D.B. Weinberger, "Network flows, minimum coverings, and the Four-Color Conjecture", Operations Research 24 (1976) 272-290.