Control
and Cybernetics
VOL. 13 (1984) No. 1–2

An efficient algorithm for partitioning a network into minimally interconnected subnetworks

by

WIESŁAW STAŃCZAK

Institute of Computer Science
Polish Academy of Sciences
PKiN, P.O.Box 22
00-901 Warszawa, Poland

This paper presents a new algorithm for determining minimal sets (sometimes called minimal groups or minimally interconnected subnetworks) in a graph. The upper bound on the number of iterations and/or simple arithmetic operations in this algorithm is derived, and is shown to compare favourably with the upper bounds on the number of iterations required by some earlier algorithms.

First, the paper states the idea of a minimal set, and points out the most important properties of some class of algorithms useful for determining minimal sets. Then, the idea of the earlier computational realization of this class is described. It occurs that in the worst case the previous algorithms need 2^n—2 iterations, and a single iteration is of type $0(n^2)$, where n is the number of vertices in a graph considered. The idea of the interdependence between a minimal set and a terminal capacity matrix, first derived by Nieminen, gives a possibility to improve the efficiency of the algorithm. This concept is developed and it leads to a faster algorithm with two kinds of iterations. It is shown that the new algorithm needs $0(n^5)$ simple arithmetic operations.

1. Introduction

The idea of minimal sets was introduced by Luccio and Sami [11] for solving a problem of dividing a given electrical network into subnetworks. Such a decomposition results in minimizing the total number of interconnections (leads) between these subnetworks under some specific conditions. Then, Kacprzyk and Stańczak [7, 9] generalized the concept of minimal sets to an arbitrary weighted graph, specifically to unigraphs with nonnegative real edge weights. The method of minimal sets proves to be a relatively useful technique for solving some specific graph partitioning problems. These problems consist in dividing the set of vertices into subsets, such that the mutual connections between vertices (called similarities) in a subset are stronger than those between the vertices in the subset and the vertices not belonging to the subset. As it was shown in several recent papers of the author et al., we can use the technique of minimal sets for preliminary partitioning a design problem

into subproblems in many practical fields, e.g. in designing a telephone interexchange network [10], in determining hierarchical structure of a computer network [8, 16], in seeking the optimal division of a group of enterprises into interacting subgroups [6], in the hierarchization of data structures [15], etc. Hence, the problem of determining an efficient algorithm for enumerating all minimal sets in a given graph is of a great interest. The first algorithm was proposed by Luccio and Sami in [11]. Kacprzyk and Stańczak have derived a number of additional properties of minimal sets and also developed an improved algorithm in [7, 9]. Some extensions introduced in [16] have led to a new form of the algorithm (or a class of algorithms) and its implementation as a computer procedure. It can be proved that the class of algorithms defined in [9, 16] provides a complete enumeration of minimal sets existing in a specified graph (see e.g. [9, 16]). The revised proof of this feature for this class of algorithms is shown in [14]. Unfortunately, all the previous realizations have not the polynomial complexity. The purpose of the paper is to propose and describe in detail a polynomial-type algorithm.

Nieminen [13] derived some relations between minimal sets and minimal cut-sets in a graph. These properties make possible to obtain a polynomial-type algorithm for finding minimal sets.

2. Preliminaries

Let $G=(X, E)$ be a finite complete undirected graph without loops and multiple edges. X is its vertex-set and E is its edge-set. $E=\{\{x, y\}: x, y \in X,\ x \neq y\}$, i.e. we refer to an edge as to an unordered pair of vertices. We assign a nonnegative weight $w(x, y)$ to all the edges of G, and it results in an ordered pair $\langle G, w\rangle$. In applications (see e.g. [6, 8, 10, 15, 16]) $w(x, y)$'s are of similarity type, and thus $\langle G, w\rangle$ is called the edge-weighted graph of similarity or simply a graph of similarity. For brevity by $f(A, B)$ we denote the following double sum

$$f(A, B)=\sum_{x \in A} \sum_{y \in B} w(x, y), \tag{1}$$

where A and B are disjoint subsets of X. Obviously, $f(A, B)=f(B, A)$. Moreover, $f(A, \emptyset)=0$, by definition.

A nonempty subset S of X is called minimal in $\langle G, w\rangle$ (further, we assume that our discussion concerns a fixed and specified ordered pair $\langle G, w\rangle$, hence the remark about $\langle G, w\rangle$ will be omitted), if for each nonempty proper subset R of S the inequality

$$f(R, X-R)>f(S, X-S) \tag{2}$$

holds.

In the introduction we mentioned a class of algorithms generating all minimal sets in $\langle G, w\rangle$. To describe any algorithm from this class we need the following properties of minimal sets (see [16]).

THEOREM 1A. 1. *$\{x\}$ is a minimal set, for each $x \in X$.*

2. *Let K be a nonempty set of indices and $\{Z_i : i \in K\}$ be a collection of pairwise disjoint minimal sets. By S_J, $J \subset K$, we denote*

$$S_J = \bigcup_{i \in J} Z_i \tag{3}$$

If the following inequality

$$f(S_J, X - S_J) < \min \{f(Z_i, X - Z_i) : i \in J\} \tag{4}$$

holds for $J = K$ and is not satisfied for any nonempty proper subset J of K, then S_K is a minimal set.

THEOREM 1B. *A set S, $|S| > 1$, is a minimal set which does not include another minimal set consisting of more than a single element if and only if*

$$f(D, X - D) < f(\{x\}, X - \{x\}) \tag{5}$$

is fulfilled for each $x \in D$ where $D = S$, and does not hold for any $x \in D$, where D is a nonempty proper subset of S.

In the sequel, we refer to Theorems 1A and 1B as to Theorem 1, for brevity.

Since G is a finite graph, then there exists a finite number of nonempty subsets of its vertices. Thus the number of minimal sets is also finite. The idea of algorithms creating the class mentioned in the introduction consists in merging some minimal sets into a new one. Then each of these algorithms terminates after performing some finite number of iterations, say $i_{\max}$ iterations, at worst. Let $L = \{0, 1, 2, \ldots, i_{\max} - 1\}$ be the set of indices of consecutive iterations. Moreover, we denote $L_1 = L - \{i_{\max} - 1\}$, for short. Each algorithm belonging to the class consists of two alternating phases which are described below.

Aggregation phase (AP).

For $j = 0$ we simply have $\langle G^0, w^0 \rangle = \langle G, w \rangle$.

Let $j \in L - \{0\}$ and let $B_j \neq \emptyset$ be a collection of all minimal sets determined in the $(j-1)$ th iteration of the algorithm, and X^j be an arbitrary set consisting of $|B_j|$ elements. We assume that d_j is a bijective function, $d_j : B_j \to X^j$. Now, a new weighted graph $\langle G^j, w^j \rangle$ is obtained, where G^j is a complete undirected graph without loops and multiple edges with a vertex-set X^j. The values of w^j are computed by using the following formula

$$w^j(x, y) = \sum_{s \in d_j^{-1}(x)} \sum_{t \in d_j^{-1}(y)} w^{j-1}(s, t), \tag{6}$$

where d_j^{-1} denotes the inverse function of d_j. In other words, each minimal set obtained in the $(j-1)$ th step is substituted by a single vertex in G^j and then the individual similarities $w^{j-1}(s, t)$ are replaced by joined similarities $f^{j-1}(A, B)$ calculated with the aid of (1), since the elements of B_j are pairwise disjoint (see SP below). If the relation $w(x, y) = 0$ holds for each edge of the graph G^j, then the algorithm terminates. Otherwise, we pass to the searching phase.

Searching phase (SP).

The $(j+1)$ th iteration, $j \in L_1$, of this phase is defined recursively as follows. In the very beginning of SP we have $J=\emptyset$ and $S_J^j=\emptyset$. We check the minimallity of subsets of $R_J^j=X^j-S_J^j$ by using Theorem 1. If we find a new minimal set Z, $|Z|>1$, then the set J is augmented with the index of Z. Moreover, S_J^j is redefined due to a formula similar to (3). Then R_J^j is updated which preserves that the elements of B_{j+1} are pairwise disjoint, etc. We terminate the current SP and the $(j+1)$ th step when there is no subset $Z\,|z|>1$, of R_J^j, which can be a minimal set. Thus, we define $K=J$ and the partial solution is obtained in the form $B_{j+1}=\{Z_i^j: i \in K\} \cup \{\{x\}: x \in R_K^j\}$ Now we check whether the equality $|B_{j+1}|=1$ holds. If so, then the algorithm terminates. Otherwise, we pass to AP of the next iteration.

Thus, the algorithms of the class considered differ one from another in the family of functions $\{d_j: j \in L_1\}$ used in AP and in the method of choosing subsets for checking their minimallity in the current R_J^j in SP. The computer implementation described in [16] is based on a modification of a fast procedure proposed by Even [2], which generates all the subsets of a given set in succession. It means that in the worst case one has to examine all the $2^{|X|}-2$ nonempty proper subsets of X.

Now, we derive the upper bound on the number of minimal sets for a given $\langle G, w\rangle$, $G=(X, E)$. Let $a_j^{(i)}$ denote the number of minimal sets having exactly j elements, where these minimal sets were obtained by merging some minimal sets with cardinality at most i. Since two minimal sets can be either disjoint or one of them is included in the other (see e.g. [7, 11, 16]), then

$$|X| \geqslant (i+1)\, a_{i+1}^{(i)}+(i+2)\, a_{i+2}^{(i)}+\ldots+(|X|-1)\, a_{|X|-1}^{(i)} \tag{7}$$

for $i=1, 2, \ldots, |X|-2$. Maximizing the value of the sum $W(i)=a_{i+1}^{(i)}+a_{i+2}^{(i)}+\ldots$ $\ldots+a_{|X|-1}^{(i)}$ subject to (7) we obtain $a_k^{(i)}=0$ for $k=i+2, i+3, \ldots, |X|-1$, and $a_{i+1}^{(i)}=$ $=\text{entier}\left(\frac{|X|}{i+1}\right)$, for each $i=1, 2, \ldots, |X|-2$. Therefore $|X|, \frac{|X|}{2}, \frac{|X|}{3}, \ldots, \frac{|X|}{|X|-1}$ is a majorant sequence for $|X|, W(1), W(2), \ldots, W(|X|-2)$. Thus we have no more than $|X|\,[\gamma+\ln(|X|-1)+1]$ minimal sets for a given $\langle G, w\rangle$, $G=(X, E)$, where $\gamma=0.577\ldots$ is the Euler constant. Hence the implementations based on Even's procedure as described above are of very low efficiency. Our task is to construct an algorithm which examines a substantially lower number of subsets than $2^{|X|}-2$.

3. On some specific sets

Let us consider a graph of similarity $\langle G, w\rangle$, and $S \subset X$ be a nonempty set of vertices. For short we denote $|S|$ by s. We assume that the elements of S are ordered in a sequence

$$\sigma(S)=(x_i: i=0, 1, 2, \ldots, s-1), \tag{8}$$

where if $k \neq r$, then $x_k \neq x_r$ and that the following condition

$$f(\{x_i\}, X-\{x_i\})=w(x_{(i-1) \bmod (s)}, x_i)+w(x_i, x_{(i+1) \bmod (s)}) \tag{9}$$

holds for each $i=0, 1, 2, ..., s-1$. Since each weight in $\langle G, w\rangle$ is nonnegative, the equality (9) means that each edge joining $x \in S$ with $y \in X-S$ is weighted by zero. Moreover, we have $w(x_i, x_{i+k})=0$ for every $i=0, 1, 2, ..., s-k-1$, $k>1$, for each $x_i, x_{i+k} \in S$. Such ordering of elements of S is called a circuit structure of S (generated by $\sigma(S)$ as in (8)) and denoted by $c(S)$. Furthermore, if $w(x_{s-1}, x_0)=0$, then we call $c(S)$ a path structure and denote it by $p(S)$. The parameter s is called the length of S. By a path substructure of $c(S)$ we understand some path structure R, where $R \subset S$, and the sequence $\sigma(R)=(x_{i(R)+i}: i=0, 1, 2, ..., |R|-1)$ generating $p(R)$ is a subsequence of that generating $p(S)$, i.e. it begins with the $i(R)$ th element of $\sigma(S)$, where $0 \leqslant i(R) \leqslant s-|R|$. Now, we can state the following lemma.

LEMMA 1. *Let us consider a circuit structure $c(S)$ and a collection of its q path substructures $\{p(H_j): j=1, 2, ..., q\}$, $q>1$.* If the path substructures are pairwise disjoint, then the set

$$Q=\bigcup_{j=1}^{q} H_j \tag{10}$$

is not minimal.

Proof. We can write

$$H_j=\{x_{i(H_j)+i}: i=0, 1, 2, ..., |H_j|-1\}. \tag{11}$$

Since these path substructures are pairwise disjoint, then the inequality $k<r$ implies

$$i(H_k)+|H_k|<i(H_r) \tag{12}$$

for each $k, r=1, 2, ..., q$. Due to (10), (11) and (12), if $k<r$, then there exists an index t, such that $x_t \in S$, $x_t \notin Q$ and $i(H_k)+|H_k|-1 <t<i(H_r)$. Therefore, we have

$$f(Q, X-Q)=\sum_{j=1}^{q} f(H_j, X-H_j), \tag{13}$$

because for each $k=1, 2, ..., q$

$$f(H_k, X-H_k)=w(x_{i(H_k)-1}, x_{i(H_k)})+w(x_{i(H_k)+|H_k|-1}, x_{i(H_k)+|H_k|}), \tag{14}$$

where the indices are taken modulo s. According to (13) the inequality

$$f(Q, X-Q) \geqslant f(H_k, X-H_k), \tag{15}$$

is satisfied for each $k=1, 2, ..., q$, and hence Q is not a minimal set. Q.E.D. ■

Applying the rule of contraposition we obtain

COROLLARY 1. *Let us consider $c(S)$ and a nonempty subset R of S. If R is a minimal set, then there exist indices r and q, $0 \leqslant r \leqslant q \leqslant s-1$, such that*

$$R=\{x_i: i=r, r+1, ..., q\}, \tag{16}$$

i.e. $p(R)$ is a path substructure of $c(S)$.

This corollary plays a very important role in the proof of validity of our new algorithm which is derived in the next section.

4. A new algorithm

First, we briefly alrec frthat a cut-set C in a graph $G=(X, E)$ consists of a nonempty set of edges $C \subset E$ln om the original graph G. By deleting C from E we obtain a new graph $(X, E-C)$ w ch the vertex-set X can be divided into two disjoint nonempty sets, say X_1 and X_2, such that there is no edge in $E-C$, connecting any $x \in X_1$ with any $y \in X_2$, while in E such edges exist. The deletion of any proper subset E' of C does not result in such partition. The partition of X into two disjoint nonempty subsets X_1 and X_2 generated by a cut-set C is denoted here by $\{X_1, X_2\} \mid C$. After removing all the edges of a cut-set from the original graph there exists no path joining any $x \in X_1$ with $y \in X_2$. To emphasize this fact we denote the cut-set by $C(x; y)$ and call it a cut-set separating x and y. In a weighted graph the value $V[C]$ of a cut-set is defined as the sum of weights of edges from C. In general, there exist many cut-sets $C(x; y)$ in a given graph $G=(X, E)$. The cut-set $C^*=C(x; y)$ separating fixed and specified vertices x and y is called minimal when $V[C^*]$ attains the smallest possible value over all the $V[C(x; y)]$.

Moreover, we recall (see e.g. [12]) that the terminal capacity matrix of $\langle G, w\rangle$ is a symmetric square matrix, in which the (i, j) th entry, $i \neq j$, is equal to the smallest possible value of $C(i; j)$, $i, j \in X$. At the main diagonal, it usually has a dummy value, say d.

Now, we consider a symmetric square matrix M with d on its main diagonal. Another entries are assumed to be real and nonegative. We assume that by a simultaneous permutation of rows and columns of M, the following representation of this matrix

$$M=\left[\begin{array}{c|c} M_a & M_c \\ \hline M_c^T & M_b \end{array}\right]$$

is obtained (the superscript T denotes the matrix transposition), where each entry of M_c has the smallest possible value over the (i, j) th elements, $i \neq j$, and M_a, M_b are square matrices with d at their main diagonals. We call M_a and M_b the resultant main submatrices. This representation is known as a principal partition of M [12]. The coincident permutation of rows and columns resulting in a principal partition of M is said to be the principal partitioning process. The following theorem holds.

THEOREM 2 [12]. *A symmetric square matrix M is the terminal capacity matrix of some $\langle G, w\rangle$ if and only if its principal partition exists and the process of principal partitioning can be continued till all the resultant main submatrices have no more than a single entry d.*

Theorem 2 implies that by successive rearrangements of rows and columns in the terminal capacity matrix M for $\langle G, w\rangle$ we obtain consecutive principal partitions of M and its resultant main submatrices. Evidently, each principal partition generates the same division of X into two disjoint subsets as some minimal cut-set. The final ordering of rows and columns in M corresponds to an arrangement of vertices of X in the so-called path realizing the terminal capacity matrix of $\langle G, w\rangle$. The

most important feature of this path is that each minimal cut-set separating its two vertices, say x and y, has the same value as the minimal $C(x; y)$ in $\langle G, w\rangle$. This property of path realizing the terminal capacity matrix is used here for increasing the efficiency of the algorithm determining minimal sets. Now, we describe a procedure constructing the path realizing the terminal capacity matrix.

First, we define an auxiliary weighted graph $\langle G^A, w^A\rangle$ on the basis of $\langle G, w\rangle$. Let X^a and X^b be two disjoint subsets of X (X^a and/or X^b may be empty), such that $X^B=X-(X^a \cup X^b)$ is nonempty. We merge X^a and X^b into single vertices, a and b, respectively and define $X^A=X^B \cup \{a, b\}$, $E^A=\{\{x, y\}: x, y \in X^A, x \neq y\}$, $G^A=(X^A, E^A)$ If $x, y \in X^B$, $x \neq y$, then $w^A(x, y)=w^A(y, x)=w(x, y)$. Otherwise, $w^A(a, y)=$ $=w^A(y, a)=f(X^a, \{y\})$, and similarly for b. Furthermore, $w^A(a, b)=w^A(b, a)=$ $=f(X^a, X^b)$. Evidently, if e.g. $X^a=\emptyset$, then $w^A(a, b)=w^A(a, y)=0$ for each $y \in X^B$.

We use the graph $\langle G^A, w^A\rangle$ in the procedure generating some path on the basis of $\langle G, w\rangle$. An idea of this procedure is the following. In the first iteration we have $X^a=X^b=\emptyset$, $X^B=X$, and, obviously, $w^A(x, a)=w^A(x, b)=w^A(a, b)=0$, for each $x \in X$. We seek a cut-set C_1 which minimizes the value of minimal cut-sets $C(x; y)$ over all pairs of distinct $x, y \in X$. Obviously, there exists such C_1, but not necessarily unique. By choosing some C_1 we attain a partition $\{X_1, X_2\}|C_1$ of the original X. If $|X_1|=|X_2|=1$, then the procedure terminates. Let us now assume that $|X_1|>1$. We merge X_2 into a single vertex b and we have $X^a=\varnothing$, $X^B=X_1$. We find a cut-set C_2 which minimizes the value of $C(y, x)$ over all pairs of distinct $x, y \in X^B$, and obtain a new partition $\{X_1', X_2'\}|C_2$. We relabel the subsets of X, such that X_1' is now denoted by X_1, X_3' — by X_2, and the old X_2 — by X_3, etc. Let us consider the general case in which $X_1, X_2, X_3, \ldots, X_q$, $q \geqslant 2$, and $|X_1|=1$. If $|X_2|=1$, for each $i=1, 2, \ldots, q$, then the procedure terminates. Otherwise, there exists the smallest subscript, say r, for which $|X_r|>1$. We have here $X^a=X_1 \cup X_2 \cup \ldots \cup X_{r-1}$ and either $X^b=X_{r+1} \cup X_{r+2} \cup \ldots \cup X_q$ for $q>r$, or $X^b=\emptyset$ for $q=r$. We seek a cut-set C_r minimizing the value of $C(x; y)$ over all pairs of distinct $x, y \in X^B=X_r$, etc. The consecutive steps of this procedure are as follows.

Construction of a path strurture (CPS).

1. Set $X_1: =X$. Merge X_1 into a single vertex $\{X\}_1$.
2. $r: =1$; $q: =1$.
3. Take the set X_r corresponding to $\{X\}_r$ and set $t: =1$.
4. If $|X_r|=1$, then $r: =r+1$ and $t: =0$.
5. If $r=|X|$, then STOP.
6. If $t=0$, then return to Step 3. Otherwise pass to Step 7.
7. Define $\langle G^A, w^A\rangle$, where $X^B=X_r$, $X^a=\bigcup_{i=1}^{r-1} X_i$, and $X^b=\bigcup_{i=r+1}^{q} X_i$.
8. Construct a minimal cut-set $C(x; y)$ in $\langle G^A, w^A\rangle$, $x, y \in X^B$, such that $V[C(x; y)]=\min\{V[C(x; y)]: x, y \in X^B, x \neq y\}$ and determine $\{X_r^1, X_r^2\}|$ $|C(x; y)$.
9. $A: =X^B \cap X_r^1$; $B: =X^B \cap X_r^2$.

10. If $r=q$, then go to Step 12.
11. For each $i=r+1, r+2, ..., q$ do: $X_{i+1}: =X_i$ and $\{X\}_{i+1}: =\{X\}_i$ (relabelling the subsets and the vertices).
12. If $a \in A$, then $X_r: =A$; $X_{r+1}: =B$ and go to Step 14.
13. $X_r: =B$; $X_{r+1}: =A$.
14. Split the old vertex $\{X\}_r$ into two new vertices $\{X\}_r$, $\{X\}_{r+1}$ and join them with the edge weighted by $V[C(x; y)]$.
15. $q: =q+1$.
16. If $q<|X|$, then go to Step 3. Otherwise, STOP.

PROPOSITION 1. The algorithm CPS generates a path realizing the terminal capacity matrix for $\langle G, w\rangle$.

P r o o f. Evidently, CPS generates a path consisting of all vertices of X. It remains to prove that the values of minimal cut-sets separating distinct vertices x and y of X are the same for the path P determined by CPS and for $\langle G, w\rangle$.

For brevity, we denote by $C^P(x; y)$ the minimal cut-set separating x and y in P and by $C^A(x; y)$ that in the current $\langle G^A, w^A\rangle$. Let $\mathfrak{C}(x; y)$ denote the set of all $C(x; y)$ in $\langle G, w\rangle$ for a fixed pair of distinct $x, y \in X$.

In the first iteration $X=X_r$ and

$$V[C^A(x; y)]=\min\{V[C(x; y)]: x, y \in X_r,\ x\neq y\}, \tag{17}$$

due to Step 8. According to $x \in X_1$ and $y \in X_2$ (or $y \in X_1$, $x \in X_2$), and (17) we have

$$V[C^P(x; y)]=\min\{V[C(x; y)]: C(x; y) \in \mathfrak{C}(x; y)\}, \tag{18}$$

because $\{X\}_1$ and $\{X\}_2$ are joined with the edge weighted by $V[C^A(x; y)]$ in Step 14. If for some $s, t \in X$, $s\neq t$,

$$\min\{V[C(s; t)]: C(s; t) \in \mathfrak{C}(s; t)\} \geqslant V[C^P(x; y)], \tag{19}$$

then either $s, t \in X_1$ or $s, t \in X_2$.

In the ith iteration, $i>1$, we have $X^B=X_r$, $|X_r|>1$. If $X^a, X^b=\emptyset$, then by the same arguments as used in [3, 5] to prove the validity of procedures useful for constructing a graph realizing a given terminal capacity matrix we obtain (17). We therefore assume that X^a and X^b are nonempty. Let C be a minimal cut-set separating x and y in $\langle G, w\rangle$ and $\{X_1, X_2\}|C=\{A \cup X_1^a \cup X_1^b,\ B \cup X_2^a \cup X_2^b\}$, where A, B are defined by Step 9 and $X^a=X_1^a \cup X_2^a$, $X^b=X_1^b \cup X_2^b$, $X_1^a, X_1^a\neq\emptyset$ and $X_2^a \cup X_2^b\neq\emptyset$. We can write $V[C]=f(A, B)+f(A, X_2^a)+f(A, X_2^b)+f(X_1^a, B)+f(X_1^a, X_2^a)+f(X_1^a, X_2^b)+f(X_1^b, B)+f(X_1^b, X_2^a)+f(X_1^b, X_2^b)$. Due to the construction of X^a and X^b and due to (19) we easily obtain

$$f(X_1^a, X_2^a)+f(X_1^b, X_2^b) \geqslant f(A, X_2^a \cup X_2^b)+f(B, X_2^a \cup X_2^b)+ \\ +f(X_2^a, X^b)+f(X^a, X_2^b) \tag{20}$$

Thus, combining (20) and the expression for $V[C]$ we get

$$V[C] \geqslant f(A \cup X^a \cup X^b, B)+\varepsilon, \tag{21}$$

where $\varepsilon=2\,[f(A, X_2^a\cup X_2^b)+f(X_2^a, X_1^b)+f(X_1^a, X_2^b)+f(X_2^a, X_2^b)]>0$. Hence, C is not a minimal cut-set separating x and y in $\langle G, w\rangle$, i.e. a contradiction. Thus, the condition (17) is fulfilled, and we obtain (18) again. Q.E.D. ■

The path $P=(X, E_P)$ realizing the terminal capacity matrix of $\langle G, w\rangle$ generates a graph of similarity $\langle G, w^*\rangle$, where the weights w^* are determined by

$$w^*(x, y)=\begin{cases} V[C^P(x; y)] & \text{if } x, y\in E_P,\\ 0 & \text{if } x, y\notin E_P.\end{cases}$$

It is evident that P is a path structure of $\langle G, w^*\rangle$. Nieminen [13] proved the following theorem.

THEOREM 3. *If S is a minimal set in $\langle G, w\rangle$, then S is also a minimal set in each path structure realizing the terminal capacity matrix of $\langle G, w\rangle$.*

As an immediate consequence of Theorem 3 we obtain the following corollary.

COROLLARY 2. *If S is not a minimal set in some path structure realizing the terminal capacity matrix of $\langle G, w\rangle$, then S is not a minimal set in $\langle G, w\rangle$.*

Due to Corollaries 1 and 2 we have the following proposition.

PROPOSITION 2. R is a minimal set in $\langle G, w\rangle$ only if it is a minimal set in a path structure realizing the terminal capacity matrix of $\langle G, w\rangle$ and there exist indices r and q, $0\leqslant r\leqslant q\leqslant|X|-1$, such that $R=\{x_i: i=r, r+1, ..., q\}$.

Hence, we have a simple and efficient method for generating subsets for checking their minimallity in SP. It can be done in the following way. Let us consider the $(j+1)$ th iteration, $j\in L_1$, where we handle $\langle G^j, w^j\rangle$, $G^j=(X^j, E^j)$. We assume that $\sigma(X^j)=(x_i: i=0, 1, 2, ..., |X^j|-1)$ is the sequence of vertices obtained in CPS, i.e. the sequence generating the path structure $p(X^j)$ (see Section 3). In the very beginning we check all the subsets Z, for which we have $|Z|=2$. According to Proposition 2 it is sufficient to take $Z_1^{(2)}=\{x_0, x_1\}$, $Z_2^{(2)}=\{x_1, x_2\}$, ..., $Z_r^{(2)}=\{x_{r-1}, x_r\}$, ..., $Z_{|X^j|-1}^{(2)}=\{x_{|X^j|-2}, x_{|X^j|-1}\}$. Now let us assume, that $S_j^j\neq\emptyset$, and we test t-tuples. If k, $0\leqslant k\leqslant|X^j|-t-1$, is the smallest index beginning a path substructure of length t consisting of the elements belonging to R_j^j, then we take $Z_1^{(t)}=\{x_k, x_{k+1}, ..., x_{k+t-1}\}$, $Z_2^t=\{x_{k+1}, x_{k+2}, ..., x_{k+t}\}$, etc. If for some t, such k does not exist, then there is no subset Z, $|Z|\geqslant t$, in the current R_j^j (and thus in the $(j+1)$ th iteration) which can be a minimal set. The correctness of the method described above directly follows from Proposition 2. Let us denote this realization of the searching phase by SP_1. Hence, we can formulate the following revised algorithm.

Algorithm.

1. Execute the AP.
2. Perform the CPS.
3. Go to Step 5.

4. Execute the AP.
5. Realize the SP_1.
6. Update the path structure being the realization of the terminal capacity matrix by merging the vertices belonging to the same minimal set and then deleting loops.
7. Return to Step 4.

5. Some properties of the revised algorithm

Now, we consider the relation between the revised algorithm and the class of algorithms defined in Section 2. We note that CPS can be included in AP for $j=0$, SP_1 is some modification of SP, and updating the path structure can be included in AP for $j \in L-\{0\}$. Thus, the revised algorithm belongs to the general class described in Section 2. All the general properties of this class, which are introduced in [9, 14, 16] hold in the case of our new algorithm. More specifically, we can formulate the following important theorems [9, 14, 16].

THEOREM 4. *The revised algorithm generates a partition of X in each iteration, i.e. the minimal sets determined in the jth step are pairwise disjoint and their union is equal to X^j. Furthermore, the algorithm gives all minimal sets in $\langle G, w\rangle$.*

THEOREM 5. *The results of the revised algorithm do not depend upon the choice of the family of functions $\{d_j: j \in L_1\}$.*

The fact that the algorithm generates a partition of X in each iteration is important from the practical point of view (see, e.g. [6, 8, 10, 11, 15, 16]). Theorem 5 has a real meaning for constructing a computer implementation. It allows to choose a family $\{d_j: j \in L_1\}$ in the way which is more convenient for us (see e.g. [9, 16]), e.g. in the simplest way from the programmer point of view.

Summarizing, the main idea of the revised algorithm is the same as in the previous algorithms generating minimal sets. The only, but also very important refinement lies in a considerable increase of the efficiency, which is shown in more detail in the next section.

6. Efficiency of the algorithm

Now, we direct our efforts to the estimation of the complexity of our new algorithm. It is evident that we can make the calculations in two separate phases. The first one concerns the initial construction in CPS. The second phase refers to the rest of the algorithm, mainly to SP_1. In CPS we seek minimal cut-sets. In the computer implementation programmed in the Institute of Computer Science of PAS we generated minimal cut-sets using the algorithm proposed by Edmonds

and Karp [1]. Now, we prepare a new version of CPS, which bases on the procedure described by Galil [4]. Then, it is convenient to make the further analysis in general, i.e. independently on the procedure used for generating minimal cut-sets.

In CPS we need to evaluate $|X|-1$ terminal capacitieres rath than one minimal cut-set and we have no way to forecast the size of sets A and B (see Steps 9, 12 and 13 of CPS), which are consecutively generated. Moreover, we do not seek in CPS a minimal cut-set separating some fixed and specified pair of distinct vertices, but we look for a minimal cut-set which has the smallest possible value among all the cut-sets separating all distinct pairs of vertices in $X^B=X_r$ (see Steps 7 and 8 of CPS). Hence, the method of Gomory and Hu [5], described also in [3], in its original form is not sufficient for our purposes, because we should obtain a path realizing the terminal capacity matrix instand of a tree.

In view of the above remarks we have to describe Step 8 of CPS in detail, then evaluate its efficiency and finally determine the numerical complexity of the whole CPS. The realization of Step 8 of CPS can be as follows. Let the vertices of X^B be arranged in any sequence $(x_i: i=1, 2, ..., |X^B|)$. We determine a minimal cut-set $C(x_2'; x_2)$, where $x_1'=x_1$, merge x_1' and x_2 into a single vertex, say x_2', determine a minimal cut-set $C(x_2'; x_3)$, merge x_2' and x_3 into x_3', etc. Further, we can use the following theorem.

THEOREM 6. *Let* $\langle G^A, w^A\rangle$ *be given as described in Section* 4. *If* $C(x_i'; x_{i+1})$, $1\leqslant i\leqslant |X^B|-1$, *is a minimal cut-set with the smallest value obtained by the procedure defined above, then*

$$V[C(x_i'; x_{i+1})]=\min\{V[C(x; y)]: x, y\in X^B,\ x\neq y\} \tag{22}$$

Proof. Let a minimal cut-set with the smallest value which separates two vertices of X^B be denoted by C^0, its value — by V^0, and $\{X_1^0, X_2^0\}=\{X_1, X_2\}|C^0$. We assume that $V^0<V[C(x_i'; x_{i+1})]$ for each i, $1\leqslant i\leqslant |X^B|-1$. We also assume that $x_1\in X_1^0$. Hence, there exists an index t, $1+t\leqslant |X^B|-2$, such that

$$A_t=\{x_i: i=1, 2, ..., t\}\subset X_1^0 \tag{23}$$

holds and $x_{t+1}\in X_2^0$. Thus, we have $V[C(x_t'; x_{t+1})]=f(A_t\cup D, E)\leqslant f(A_t\cup(X_1^0-A_t), X^A-X_1^0)=V^0$, for some D, E and $x_{t+1}\in E$ (D may be empty). Therefore, we obtain $V^0<V^0$, i.e. a contradiction. Q.E.D. ∎

Now, we note again that there is no way to forecast the size of sets A and B consecutively generated in CPS. To estimate the computational effort needed for execution of CPS we have to define a vertex weighted arborescence (i.e. a rooted directed tree) of the partitioning. We proceed in a recurrent way based on the description of CPS, as follows.

The root is the vertex $\{X\}_1$ defined in Step 1 and its weight equals $|X|$. Initia..y, t has no label. From the root there are two arcs directed to vertices $\{X\}_1$ and $\{X\}_2$ obtained in the first iteration by splitting the original $\{X\}_1$. Initially, they are also unlabelled and their weights are $|A|$ and $|B|$ (see Steps 9, 11 and 13), respectively.

We assume that some vertex $\{X\}_r$ was reached in Step 4 of the ith iteration, $i>1$.
CASE 1. $|X_r|=1$. There exists a single arc incident in $\{X\}_r$ and there is no arc incident out. We backtrack the arc to the nearest vertex and increment r by 1. It follows from CPS that a new vertex, we reached, has the current number r, and $|X_r|>1$. If $\{X\}_r$ is labelled, then we check whether it is the root. If so, the procedure terminates. Otherwise, we backtrack once more, increment r by 1, and again, until we reach an unlabelled vertex or the labelled root. The latter situation was described before. In the former one we label the vertex and the arc incoming to the current $\{X\}_r$ and proceed in the direction of the unlabelled arc outgoing from $\{X\}_r$.
CASE 2. $|X_r|>1$. We introduce two arcs outgoing from the old $\{X\}_r$ to new vertices $\{X\}_r$ and $\{X\}_{r+1}$ obtained by splitting the original $\{X\}_r$ (see Step 14). Neither the old vertex $\{X\}_r$ nor the new vertices $\{X\}_r$ and $\{X\}_{r+1}$ are labelled. We assign to the new $\{X\}_r$ and $\{X\}_{r+1}$ the values of $|A|$ and $|B|$, respectively, and so on.

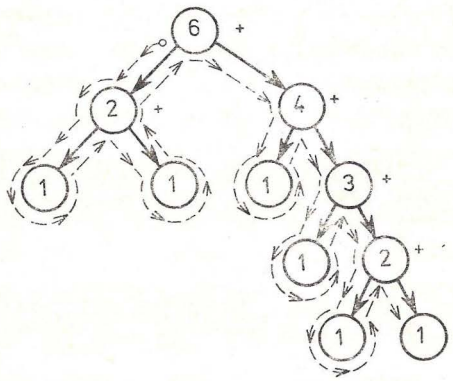

Fig. 1. An example of construction of a weighted arborescence.

An example of the walk described above is shown in Fig. 1. It can be easily stated that each vertex weighted by one was passed only once. The other vertices are passed three times. It is obvious, because to the vertex weighted by one there is only one incoming arc and no arc is outgoing. To the other vertices, excluding the root, there are incident three arcs. Two of them are outgoing and a single is incoming. In fact, each vertex with the weight greater than 1 is split into two parts and each vertex results from splitting only one vertex. The root is not obtained by the splitting mechanism, and there are only two arcs outgoing from it. Since we start our walk from the root, then we are in the root three times, too. From the above discussion the next proposition follows.

PROPOSITION 3. For each $\langle G, w\rangle$ and each realization of CPS the above walk terminates in a finite number of steps. It gives an arborescence which has the vertices weighted by cardinal numbers of sets X_r obtained during the execution of CPS. Each arc is directed from the vertex corresponding to some X_r to another vertex corresponding to a subset of X_r. All the vertices weighted by more than one are labelled.

The arborescence gives an idea how CPS works. We need it for the estimation of the efficiency of CPS. The unlabelled vertices correspond to the sets which are not partitioned. Thus, we can remove from the arborescence each vertex without label and arcs incident to it. Further we can also delete every vertex weighted by two and then the arcs incident to it. Evidently, the minimal cut-set in $\langle G^A, w^A\rangle$ with $|X^B|=2$ can be obtained at once. It is obvious that now we also get an arborescence A of the partitioning. We define the value $F(A)$ of A, as follows

$$F(A)=\sum_{x\in X_A} h\,(v\,(x)), \tag{24}$$

where X_A is the vertex-set of A, $v\,(x)$ denotes the weight assigned to $x\in X_A$, and $h\,(y)$ is the complexity of procedure generating minimal cut-sets in a graph with y vertices. To estimate the upper bound for the computational complexity of the whole CPS we have to maximize F over all possible arborescences of partitioning.

THEOREM 7. *If h is an increasing and nonnegative valued function for arguments not less than 2, then F reaches its maximal value for $A=A^*$, where A^* denotes a path with vertices weighted by consecutive natural numbers from $n=|X|$ to 3.*

P r o o f. First, we introduce some additional notations. Let A_n denote any arborescence of partitioning defined for a graph G with n vertices. We have $F(A_i)=0$, for $i=1, 2$. Hence, we notice that the following formula

$$F(A_n)=h\,(n)+F(\hat{A}_{n-r})+F(\hat{A}_r) \tag{25}$$

holds for every A_n, $n>1$, where $\hat{A}_{n-r}$ and $\hat{A}_r$ are the subarborescences of A_n obtained by removing the root and arcs incident to it from A_n.

The proof will be accomplished by induction on n. There is only one A_3. For $n=4$ we have two arborescences. They are shown in Fig. 2. We calculate that $F(A_4')=$ $=h\,(4)<F(A_4^*)=h\,(4)+h\,(3)$, due to the properties of h.

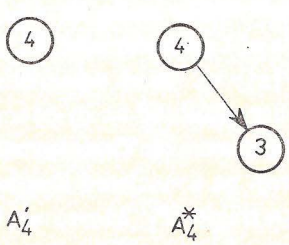

Fig. 2. Two arborescences A_4.

Now, let us assume that for each m, $3\leqslant m\leqslant n-1$, and for every A_m the following inequality

$$F(A_m)\leqslant F(A_m^*) \tag{26}$$

is satisfied. According to (25) and (26) we have $F(A_n)-h(n)=F(\hat{A}_{n-r})+F(\hat{A}_r)\leqslant$ $\leqslant F(A^*_{n-r})+F(A^*_r)$, where obviously

$$F(A^*_k)=\sum_{i=1}^{i} h(i) \tag{27}$$

According to the properties of h and due to (27) $F(A^*_{n-r})+F(A^*_r)\leqslant F(A^*_{n-1})$ for $r\geqslant 3$ and $n-r\geqslant 3$. If $r=1$ or $r=2$, then $F(A^*_r)=0$ and we have $F(A^*_{n-r})\leqslant F(A^*_{n-1})$ again. Analogously, for $n-r=1$ or $n-r=2$ we obtain $F(A^*_r)\leqslant F(A^*_{n-1})$. Hence, obviously, $F(A^*_n)\leqslant F(A^*_n)$. Q.E.D. ∎

It means that in the whole CPS we have to perform no more than $F(A^*_n)$ simple arithmetic operations such as additions, subtractions and comparisons. When we use the procedure described by Edmonds and Karp we need $\frac{1}{4}(k^3-k)\,k^2$ these operations for finding a minimal cut-set separating a given pair of vertices in a graph with k vertices [1]. Thus, the procedure described above Theorem 6 is of type $0(n^6)$. Therefore, due to Theorem 7 the whole CPS needs $0(n^7)$ simple arithmetic operations. The procedure of Galil is of type $0(n^{5/3}e^{2/3})$ [4], where e denotes the cardinality of edge-set, i.e. of type $0(n^3)$ in the worst case in which the square matrix consisting of entries being the weights $w(x, y)$ is not sparse. Thus, the new form of CPS now realized in the Institute of Computer Science needs $0(n^5)$ simple arithmetic operations.

It can be easily shown that for testing a subset of X^j consisting of i vertices we need $\frac{i^2-i}{2}+i+1$ additions, 1 subtraction and i comparisons, if all the values of $f^j(\{x\}, X^j-\{x\})$, $j\in L$, are known (f^j in $\langle G^j, w^j\rangle$ is defined by the same formula as f in $\langle G, w\rangle$. In the other cases, the number of these operations is greater. Thus, the first case is preferable and we consider it now. Therefore, we need at most

$$\sum_{i=2}^{k-1}(k+1-i)\,\frac{i^2+3i+4}{2}=\frac{(k-2)\,(k^3+12k^2+59k+24)}{24},$$

i.e. $0(k^4)$ simple arithmetic operations in the jth, $j\in L$, iteration, because we handle no more than $k+1-i$ subsets of i elements (see SP_1), where $k=|X^j|$. For the initial evaluation of $f^0(\{x\}, X^0-\{x\})$ we need $n(n-1)$ additions. For $j\in L-\{0\}$ the values of $f^j(\{x\}, X^j-\{x\})$ are obtained as by-products of testing the minimallity of the subsets from X^{j-1} (see Theorem 1). Thus, the evaluation of $f^j(\{x\}, X^j-\{x\})$'s, being in fact a part of AP, can be considered simultaneously with SP_1 and has at most the same complexity as a single SP_1. Moreover, the rest of AP can be executed by simple changing the indices of two working lists (see [9]) which extremly speeds up the algorithm and saves the computer memory. Therefore, since $i_{\max}\leqslant n$ (which can be easily proved by a construction similar to the arborescence of partitioning), then AP and SP_1 in the whole algorithm need $0(n^5)$ simple arithmetic operations, which is also the complexity for the new algorithm as a whole. For a comparison we recall that the upper bound of the old version is $0(2^n n^2)$ (see Section 2).

References

[1] EDMONDS J., KARP R. M. Theoretical improvements in the algorithmic efficiency for network flow problems. *Journal of the ACM*, **19** (1972) 2, 248–264.

[2] EVEN S. Algorithmic combinatorics. New York, Macmillan, 1973.

[3] FORD L. R., FULKERSON D. R. Flows in networks. Princeton, N. J., Princeton Univ. Press,1962.

[4] GALIL Z. An O ($V^{5/3}E^{2/3}$) algorithm for the maximal flow problem. *Acta Informatica*, **14**, 3 (1980), 221–242.

[5] GOMORY R. E., HU T. C. Multi-terminal network flows, *SIAM J. Appl. Math.*, **9**, (1961), 551–570.

[6] KACPRZYK J., STAŃCZAK W. Application of the method of minimally interconnected network for solving the problem of partitioning a group of enterprises into subgroups (in Polish). *Archiwum Autom. i Telemech.*, **20**, 4 (1975), 513–526.

[7] KACPRZYK J., STAŃCZAK W. On an extension of the method of minimally interconnected subnetworks. *Control and Cybernetics*, **5**,4 (1976), 61–77.

[8] KACPRZYK J., STAŃCZAK W. Partitioning a computer network into subnetworks and allocation of distributed data bases. Proc. 8th IFIP Conf. Optimization Techn., Würzburg, 1976. Springer, 1977, 464–472.

[9] KACPRZYK J., STAŃCZAK W. On a further extension of the method of minimally interconnected subnetworks. *Control and Cybernetics* **7**, (1978) 2, 17–31.

[10] KALISZEWSKI I., NOWICKI T., STAŃCZAK W. On the telephone interexchange network structure decomposition using the method of minimally interconnected subgraphs (in Polish). *Rozprawy Elektrotechn.*, **21**, (1975) 2, 573–580.

[11] BUCCIO F., SAMI M. On the decomposition of networks in minimally interconnected subnetworks. *IEEE Trans. Circuit Theory*, **CT-16**, (1969) 2, 184–188.

[12] MAYEDA W. Graph theory. New York, John Wiley, 1972.

[13] NIEMINEN J. On minimally interconnected subnetworks of a network. *Control and Cybernetics*, **9**, (1980) 1–2, 47–52.

[14] NOWICKI T., STAŃCZAK W. On some class of algorithms using for determination of minimal groups in cliques. Proc. International Symp. on Applications of Math. in System Theory, Braşov, 1978, 275–280.

[15] NOWICKI T., STAŃCZAK W. Partitioning a set ot elements into subsets due to their similarity. Proc. 2nd International Symp. Data Analysis and Informatics, Versailles, 1979. (in) Data Analysis and Informatics, Diday E. et al. (eds). Amsterdam, North Holland, 1980, 583–591.

[16] STAŃCZAK W. Application of the method of minimally interconnected subnetworks to computerized designing of topological structure and configuration for teleprocessing network (in Polish). Ph. D. Thesis, Inst. of Computer Sci., Polish Academy of Sci., Warsaw, 1978.

Received, March 1982.

Efektywny algorytm podziału sieci na zespoły minimalne

W artykule przedstawiono nowy algorytm wyznaczania zespołów minimalnych w grafie. Podano także górne ograniczenie liczby iteracji wykonywanych w trakcie działania algorytmu. Okazało się, że nowy algorytm jest znaczenie lepszy od poprzednich.

Na wstępie przedstawiono pojęcie zespołu minimalnego i wymieniono najważniejsze właściwości pewnej klasy algorytmów służących do wyszukiwania zespołów minimalnych. Następnie opisano dotychczasową realizację wspomnianej klasy algorytmów. Okazało się, że w najgorszym przypadku poprzedni algorytm wymagał wykonania 2^n—2 iteracji (każda z nich wymaga co najwyżej 0 (n^2) operacji), gdzie n jest liczbą wierzchołków w rozpatrywanym grafie. Nieminen był

pierwszym, który odkrył zależność między zespołami minimalnymi a macierzą pojemności granicznych grafu. Wykorzystanie tego wyniku pozwala zwiększyć efektywność algorytmu wyznaczającego zespoły minimalne. Rozwinięcie koncepcji Nieminena doprowadziło do uzyskania szybkiego algorytmu o dwóch rodzajach iteracji. Górne ograniczenie czasu działania nowego algorytmu wynosi $0(n^5)$ prostych operacji arytmetycznych.

Эффективный алгоритм декомпозиции сети на минимально связные графы

В статье предложен новый алгоритм определения минимально связных подграфов. Дается также верхний предел числа итераций алгоритма. Новый алгоритм является лучше предыдущего.

Вначале предлагается понятие минимально связного подграфа и приводятся наиболее важные свойства некоторого класса алгоритмов, определяющих эти подграфы. Затем представлена более ранняя реализация выше упомянутого класса алгоритмов. В худшем случае она требует $2^n - 2$ итераций (каждая из них требует не более чем $0(n^2)$ операций), где n — число вершин рассматриваемого графа. Впервые Неминен открыл зависимость минимально связных подграфов и матриц конечных пропускных способностей. Применение этого результата дает возможность повысить эффективность алгоритма определения минимально связных подграфов. Развитие концепции Неминена привело к полученою более скоростного алгоритма, в котором существуют итерации двух типов. Верхний предел времени действия нового алгоритма равен $0(n^5)$ элементарных арифметических операций.