Network Decomposition for the Optimization of Connection Structures

Alfred Iwainsky,* Enrico Canuto,† Oleg Taraszow,* and Agostino Villa†
**Akademie der Wissenschaften der DDR; Zentralinstitut für Kybernetik und Informationsprozesse; GDR 1086 Berlin, Kurstr.33 and †Politecnico di Torino, Dipartimento di Automatica e Informatica; Corso Duca degli Abruzzi 24—10129 Torino, Italy*

In many practical situations, connection structures have to be laid out in an environment with a strong inner structure. This article presents a new general mathematical formulation of corresponding design problems in which the layout possibilities are represented by a network. Because in practice such networks are often very large and sparse, there is great interest in the utilization of decomposition techniques for the optimization of connection structures. New decomposition methods for the determination of optimal paths without interference (independent decomposition), of k node-disjoint paths with minimal total costs and of optimal Steiner trees are presented. The new methods are compared with other techniques under different aspects of practical importance.

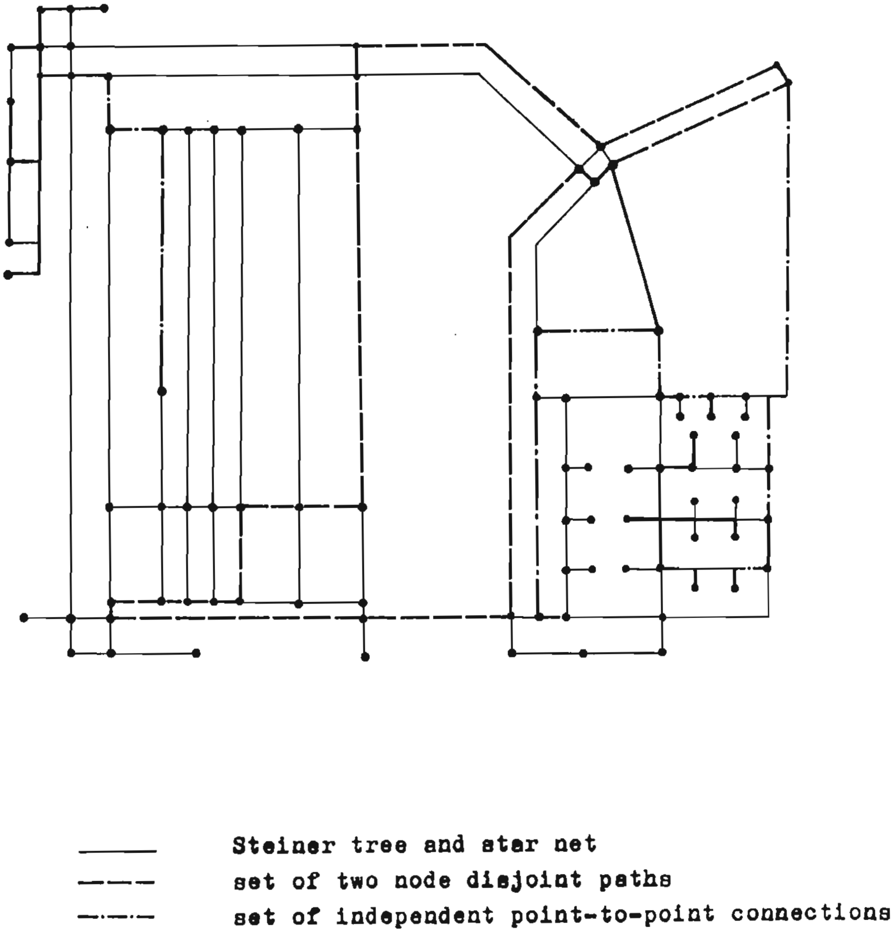
1. INTRODUCTION

Many practical design problems regard the optimization of structures connecting a set of given points in the two- or three-dimensional Euclidean Space. This article considers those cases in which only a finite number of essentially different admissible paths between any two given points exists. Then it is reasonable to embed a network as a model of the routing, branching, and placement possibilities in the region in which connection structures have to be laid out. Such situations arise in transport optimization, routing of pipelines and cable connections in environments with intense inherent structure and in other fields (see also [1, 2, 3]).

The entire network that represents routing, placement, and branching possibilities for the layout of connection structures may be very large. In an industrial site, for

This research has been supported by the Italian National Research Council–Progetto finalizzato Informatica and the Academy of Sciences of the German Democratic Republic and carried out in the frame of the cooperation between these institutions.

CCC 0028-3045/86/020205-31$04.00



Steiner tree and star net
set of two node disjoint paths
set of independent point-to-point connections

FIG. 1. Different connection structures in a fictive network representing layout possibilities in an industrial site.

instance, there are networks with hundreds to thousands of edges each representing a point-to-point connection already prepared or at least suitable for the installation of cables [3]. These networks are always sparse.

In many practical situations, connection structures of different kind must be laid out in one and the same sparse large-scale network. Often at least some of these structures have to connect nodes being located within relatively small regions of the entire network. Figure 1 illustrates a simple example of such a situation in which a set of noninteracting point-to-point connections, a star net, a Steiner tree, and two node-disjoint paths are marked as design results.

In the situations described above the computational effort for the optimization of connection structures can be generally significantly reduced by network decomposition techniques. This has already been utilized for the determination of all optimal paths in a graph ([4–9], e.g.). Based on the correspondence between this problem and the solution of linear algebraic systems [10] sparse matrix techniques have been applied [8, 9].

Section 2 of this article regards decomposition techniques for the determination of optimal paths. In Subsection 2.2, a new method is described.

More complex structures than paths represent an even stronger challenge for the application of network decomposition. In Section 3 we describe methods for the decomposition of the entire network into two parts, one being sufficient for the solution of the layout-optimization problem, the other being absolutely irrelevant for that purpose. Such techniques are especially important in those cases in which the structures to be laid out have to connect nodes located in relatively small regions of large-scale networks. The proposed algorithms answer the question which parts of the network can be eliminated without loss of the optimal solution of the original problem.

In Section 4, the different methods described in the article are compared with respect to such characteristics as suitable applications, implementation effort, storage requirements, computational effort, and typical man–machine interaction possibilities within the entire problem-solving process.

As an outlook on future research work the last section contains a formal generalization of optimization problems regarding the layout of connection structures in environments with pronounced inner structure.

In the more formal parts of this paper the network representing the layout possibilities in such environments is called graph (or digraph) G.

2. DECOMPOSITION METHODS FOR THE DETERMINATION OF OPTIMAL PATHS IN GRAPHS

2.1. Decomposition without Optimality Properties

In the practical applications characterized in Section 1 the occurring graphs are far from being completely connected. Decomposition techniques can take advantage of this fact. At first we consider a very simple situation:

Let Y_1 be a subset of the nodes of a given digraph $G = (Y,W)$ with the set of vertices and arcs $Y = V(G)$ and $W = E(G)$, respectively:

$$Y_1 \subseteq Y = V(G).$$

By X we denote the set of all nodes of G which are neighbors of nodes of Y_1, but do not belong to Y_1. Finally we define

$$Y_2 = (Y \backslash Y_1) \backslash X.$$

Obviously X is a cut set: If the nodes of X and the arcs directed from or to elements of X to or from elements of $Y \backslash X$ are removed from G, G is decomposed into the two nonoverlapping parts G_1 and G_2 with $V(G_1) = Y_1$ and $V(G_2) = Y_2$, respectively (see Fig. 2 for the case of undirected graphs).

Now we define two overlapping digraphs $\overline{G}_1 = (\overline{Y}_1,\overline{W}_1)$ and $\overline{G}_2 = (\overline{Y}_2,\overline{W}_2)$ which result if the node sets of G_1 and G_2 are both extended by X and if $\overline{W}_1$ and $\overline{W}_2$ contain all arcs of G connecting elements of $Y_1 \cup X$ and $Y_2 \cup X$, respectively.

Let us look at the optimal path problem on G for a given node cut set X. If no further properties than the above described are imposed on the decomposition, for the determination of an optimal path between two vertices of $\overline{G}_1$ we have to consider also $\overline{G}_2$ and vice versa. The fundamental idea of the decomposition described in this subsection lies in a successive execution of path-finding algorithms in subgraphs of G.

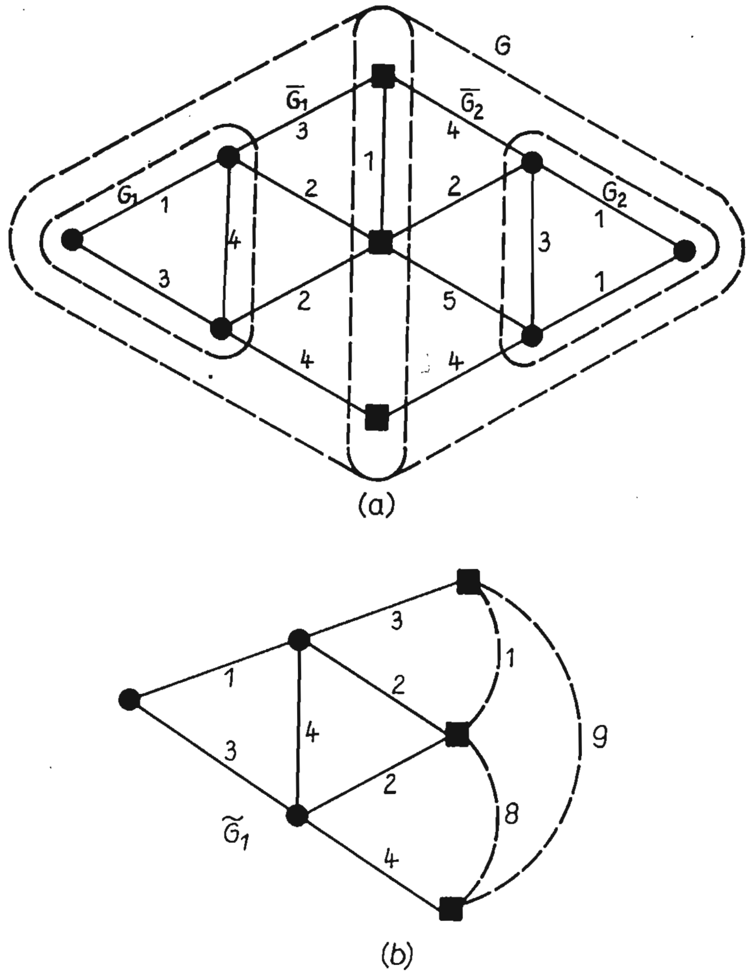


FIG. 2. (a) A decomposition of G by the cut set X (elements of X are represented by squares). (b) Graph $\tilde{G}_1$ with edges (dotted lines) representing optimal paths between all elements of X in $\overline{G}_2$.

Such an organization of the optimization can be utilized for a reduction of both the necessary computation time and memory space. Consider, for instance, the problem of finding an optimal path between $y_i \in \overline{Y}_1$ and $y_j \in \overline{Y}_1$. At first within $\overline{G}_2$ the optimal paths from every vertex of X to all other vertices of this set are determined. Then $\overline{G}_1$ is modified in the following way: All arcs connecting vertices of X with each other are eliminated from $\overline{G}_1$. Then for every ordered pair (x_i,x_j) of vertices of X for which a path from x_i to x_j exists in $\overline{G}_2$ we add to $\overline{G}_1$ one arc representing an optimal path from x_i to x_j in $\overline{G}_2$ and being weighted by its cost (see Fig. 2 for the case of an undirected graph). The resulting graph is called $\tilde{G}_1$. Now we are able to solve the optimal path problem with respect to the entire digraph G considering only graph $\tilde{G}_1$.

The described procedure provides optimal paths between elements of $\overline{Y}_1$ and between elements of $\overline{Y}_2$. If we look for an optimal path from $y_i \in Y_1$ and $y_j \in Y_2$ we obviously have to perform the following minimization

$$d(y_i,y_j) = \min_{x \in X} [d(y_i,x) + d(x,y_j)] \quad (1)$$

TABLE I. Short characterizations of some well-known optimal path algorithms not using decomposition methods.

Author	Purpose of the Algorithm	Approximate Number of Operations	Reference
Dijkstra	Determination of the optimal paths from one source node to all other nodes in a network with nonnegative weights	$O(N^2)$	[13]
Yen	as Dijkstra's algorithm	$O(N^2)$ (less than Dijkstra's algorithm)	[11]
Bellmann	as Dijkstra's algorithm, but negative weights admissible	$O(M_1 \cdot N^2)^a$	[14]
Yen	as Bellman's algorithm	$O(M_2 \cdot N^2)^a$ (less than Bellman's algorithm)	[15]
Floyd	Simultaneous determination of all optimal paths in a network with negative weights admissible	$O(N^3)$	[16]
Spira	as Floyd's algorithm, but negative weights are not admissible	$O(N^2(\log N)^2)$	[17]

[a] M_1 and M_2 lie between 1 and $N - 1$, depending on the particular network.

where $d(y_i,y_j)$ with y_i, $y_j \in Y$ are the costs of an optimal path from y_i to y_j with respect to the entire digraph G.

Thus, it is sufficient to solve optimal path problems successively only on both of the subnetworks and to carry out minimization (1). In the case of a proper choice of the cut set and sufficiently large sparse networks already this simple decomposition leads to a significant reduction of necessary memory space and computation time. The reason for the potential reduction of computer time lies in the fact that the computational effort for the determination of optimal paths in a network with N nodes generally increases faster than linearly with rising N. Table I illustrates the computational efficiency of some well-known powerful optimal path algorithms that do not use decomposition. More detailed comparisons can be found, e.g., in [11, 12].

Of course, regarding computer time a decomposition is most beneficial if all or at least many optimal paths have to be determined.

The described algorithm that goes back to Hu [4] can be extended in the direction of a more general decomposition. In [4] a cascade and a hierarchical cascade decomposition are proposed. By a cascade decomposition a network is decomposed into a number of subnetworks in such a way that the "inner" subnetworks overlap only with two others and the two "outer" subnetworks overlap only with one subnetwork each (Fig. 3).

In the case of a hierarchical cascade decomposition at least one of the resulting subnetworks is further decomposed into a cascade structure.

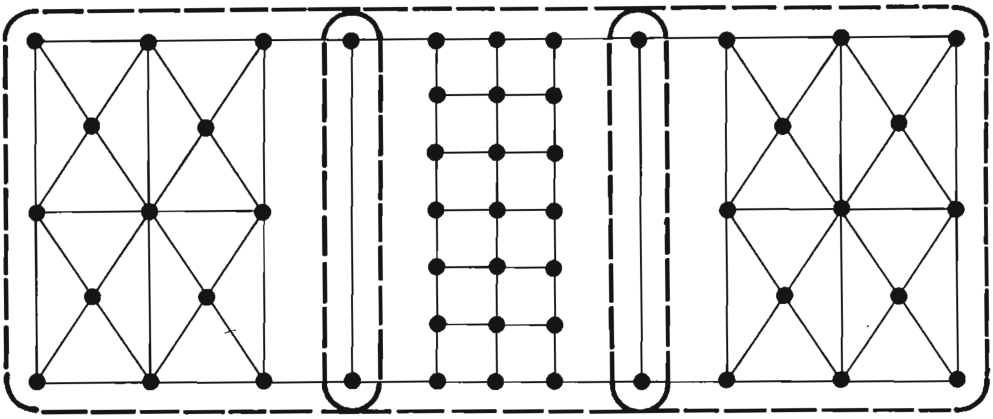


FIG. 3. Example of a cascade decomposition.

2.2. Decomposition by Optimal Paths

The method for the determination of optimal paths in a network described in Section 2.1 does not answer the essential question how to choose the cut sets. In fact, in [4] the effort for the decomposition of the entire network into a proper set of subnetworks was not considered in the efficiency analysis. In certain practical situations, for instance, if we investigate large planar networks and have a graphical representation of them at hand, it may be relatively easy to choose proper cut sets manually. In such situations it is an advantage of the method of Hu, to permit, in principle, an *arbitrary* cascade decomposition.

On the other hand, especially in more complex applications we are interested in a fully computerized method without interaction by man defining the cut sets. In [6] and [7] a Dynamic Programming procedure for an optimal decomposition is described which supplies for appropriately chosen integers m (number of subnetworks) and K (maximum number of nodes in each subnet) a minimal number of elements in the union of all node cut sets. The term optimal is used there in a similar sense as in control theory: The solution is optimal with respect to an initial state given by an initial guess of m vertices which represent ''crystal nuclei'' for the *m* subnets. Thus, the determination of an optimal solution with respect to all possible initializations can be rather time consuming.

In this section we present another algorithm for the calculation of optimal paths in which certain optimal paths themselves are used for an algorithmic decomposition. Thus, the computational effort for the decomposition is fairly moderate in comparison with the method described in [6, 7]. Furthermore, the algorithm eliminates the necessity to determine optimal paths additionally in $\overline{G}_2$ if we look for such routes in $\overline{G}_1$, to speak in terms of a decomposition into two subnetworks as illustrated in Figure 2. The algorithm is effectively applicable to the determination of all or at least many optimal paths in large planar networks, and in special other cases.

In the following we assume that graph G in which we look for optimal paths is undirected. A generalization to digraphs is possible only in very special cases. At first we introduce some terms regarding a sequence of special optimal paths.

Definition 1. A first level optimal border is an optimal path between two nodes of graph G decomposing this graph into at least two nonoverlapping subgraphs if it is eliminated from G. The corresponding subgraphs overlapping on the optimal border are called first-level subnets.

Definition 2. A kth level optimal border ($k > 1$) is an optimal path between two nodes of a $(k - 1)$-th level subnet representing a first-level optimal border with respect to this subnet. The resulting subgraphs overlapping on the optimal border are called kth-level subnets.

Definition 3. An independent decomposition of an undirected graph G is a successive partition of G and the resulting subnets by optimal borders.

Figure 4 shows a simple typical example of an independent decomposition. At first the entire network is decomposed into two subnets by a first level optimal border stretching from one boundary node to another. Then one of the resulting first-level subnets is further decomposed by a second-level optimal border connecting a boundary node with a node on the first-level optimal border.

The level of an optimal border may depend on the sequence in which a network is decomposed by such borders.

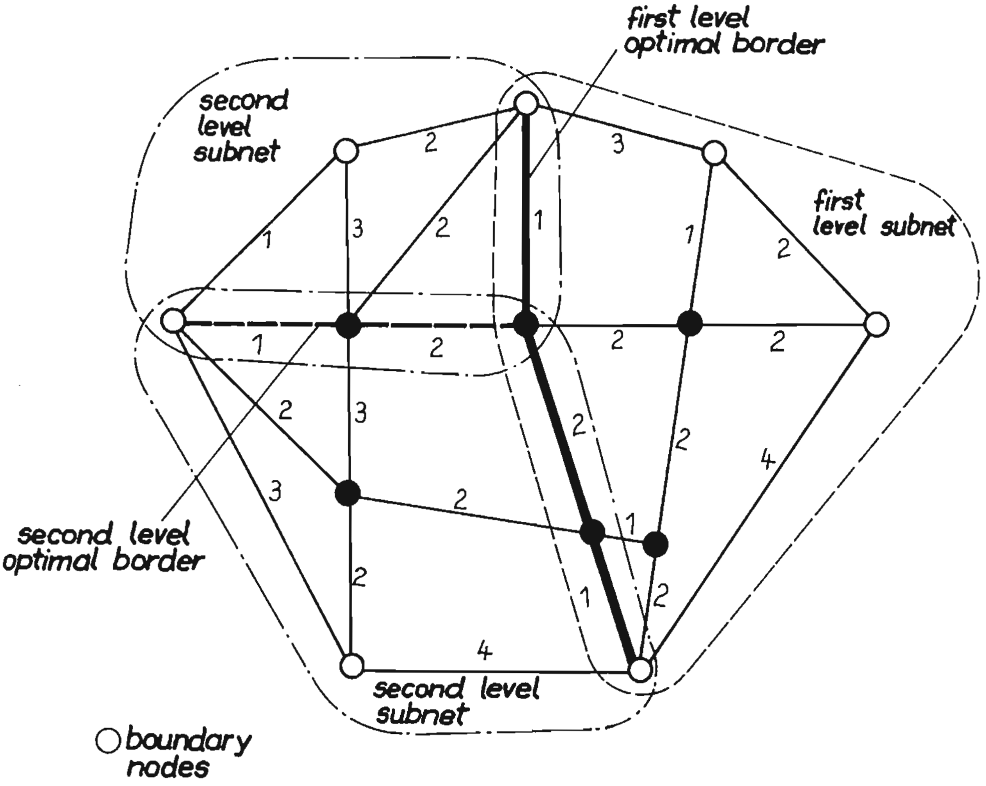


FIG. 4. Independent decomposition by a first- and a second-level optimal border sharing one node of the network.

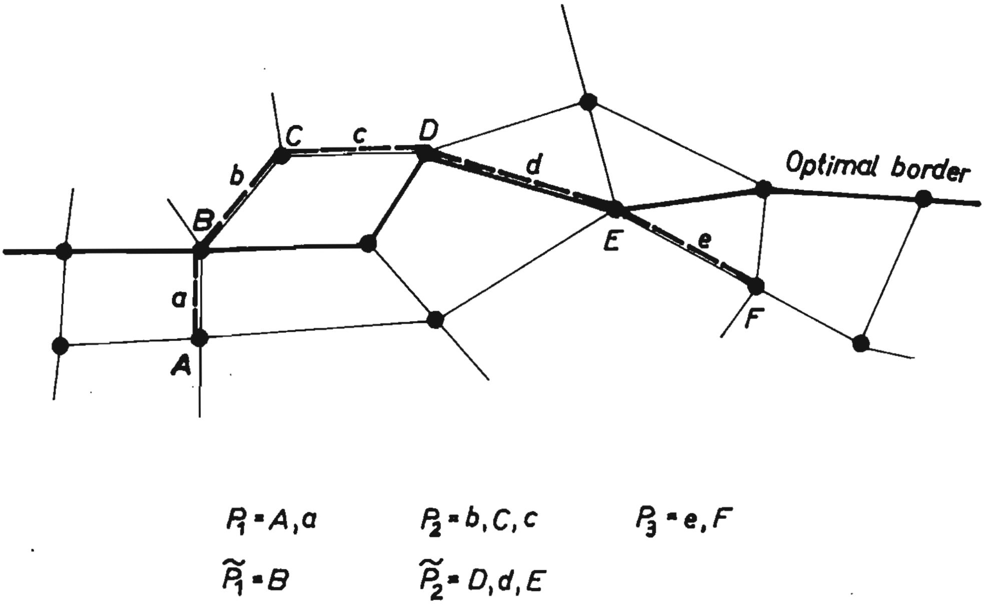


FIG. 5. Example of a path with degree 2 with respect to an optimal border.

Definition 4. A path $p(v_{i_1},v_{i_r})$ in $G = (Y,W)$, represented by the sequence of contiguous nodes and edges

$$v_{i_1},\ e_{j_1},\ v_{i_2},\ e_{j_2},\ \ldots,\ v_{i_{r-1}},\ e_{j_{r-1}},\ v_{i_r} \qquad (2)$$

$$v_{i_s} \in Y \qquad (s = 1, \ldots, r)$$

$$e_{j_t} \in W \qquad (t = 1, \ldots, r - 1),$$

has degree $n \geqslant 1$ with respect to an optimal border B (also represented by a sequence of contiguous nodes and edges), if sequence (2) can be divided into a sequence of segments

$$P_1,\ \tilde{P}_1,\ P_2,\ \tilde{P}_2,\ \ldots,\ P_n,\ \tilde{P}_n,\ P_{n+1},$$

where each $\tilde{P}_i$ $(i = 1, \ldots, n)$ coincides with some segment of B, no element of P_i $(i = 1, \ldots, n + 1)$ belongs to B, and only P_1 and P_{n+1} may be empty (Fig. 5).

Lemma 1. If a path between two nodes of one and the same subnet resulting from an independent decomposition does not remain entirely inside this subnet it has a degree $n > 1$ with respect to at least one optimal border.

Proof. We prove this lemma by complete induction.

Consider a first level optimal border $B^{(1)}$ and one of the resulting first-level subnets $G_j^{(1)}$ in which the starting and destination node of a path may be located. Because the optimal border by definition decomposes the entire network, a path leaving $G_j^{(1)}$ must obviously have a degree $n > 1$ with respect to $B^{(1)}$.

Assume that the lemma is right for a kth-level subnet. Now we consider paths between two nodes of one and the same $(k + 1)$-th level subnet $G_j^{(k+1)}$ $(k \geqslant 1)$. By Definition

2 this subnet is part of a kth-level subnet, say, $G_i^{(k)}$, which is decomposed by a $(k + 1)$-th level optimal border, say, $B^{(k+1)}$. According to the induction assumption, each path between two nodes of $G_j^{(k+1)}$ leaving $G_i^{(k)}$ has a degree $n > 1$ with at least one optimal border. If a path between two nodes of $G_j^{(k+1)}$ leaves $G_j^{(k+1)}$ but remains inside $G_i^{(k)}$ it has a degree $n > 1$ with respect to $B^{(k+1)}$. ■

Lemma 2. There is no path between two nodes v_i and v_j of a network independently decomposed which has a degree higher than one with respect to any optimal border and lower costs than all other paths between v_i and v_j.

Proof. We prove Lemma 2 by contradiction. Assume that a path $p(v_i,v_j)$ exists which has degree $n > 1$ with respect to an optimal border B and lower costs than any other path between these nodes. By Definition 4 this path has at least two nodes (say, v_r and v_s) in common with B which are separated by at least one segment of p not coinciding with a segment of B. By the definition of optimal borders (Definitions 1 and 2) there is no path between v_r and v_s having lower costs than the path along B. Thus, if the segment between v_r and v_s in p is substituted by the path between these nodes along the optimal border B, another path $p^+(v_i,v_j)$ results whose costs are lower than or equal to those of p. This contradicts the assumption. ■

Lemma 3. If a graph G is independently decomposed and two vertices (v_i and v_j) belong to one and the same of the resulting subnets overlapping exclusively on optimal borders then at least one of all optimal paths between v_i and v_j lies entirely in this subnet.

Proof. According to Lemma 1 each path between v_i and v_j leaving the subnet has a degree $n > 1$ with respect to at least one optimal border. Thus, Lemma 3 follows immediately from Lemma 2. ■

Lemma 3 reveals the reason for the choice of the term independent decomposition: In contrast to the approach of Hu (see Section 2.1) the optimization of paths between two nodes of one and the same subnet can be carried out independently of the other subnets.

Optimal paths between nodes of different subnets can be determined on the base of optimal paths lying entirely within one subnet. This is described by Eq. (1) for the case of two subnets. If the independent decomposition results in a cascade of overlapping subnets as presented in Figure 3 the determination of optimal borders provides a sequence of decision stages for a multistage optimization process, to speak in terms of Dynamic Programming.

Figures 6 and 7 illustrate the determination of optimal paths on the base of independent decomposition. The graph whose structure is shown in Figure 6 was generated by a random graph generator described in [18] and developed for the rationalization of our research process. In this example integer numbers equally distributed in [0, 10] were assigned as costs to the edges of G. Only for the purpose of illustration some few nodes that have to be connected in pairs by optimal paths are marked. With respect to the location of these nodes and to the structure of the graph it is reasonable to decompose the entire graph by optimal borders in the region of its natural bottlenecks.

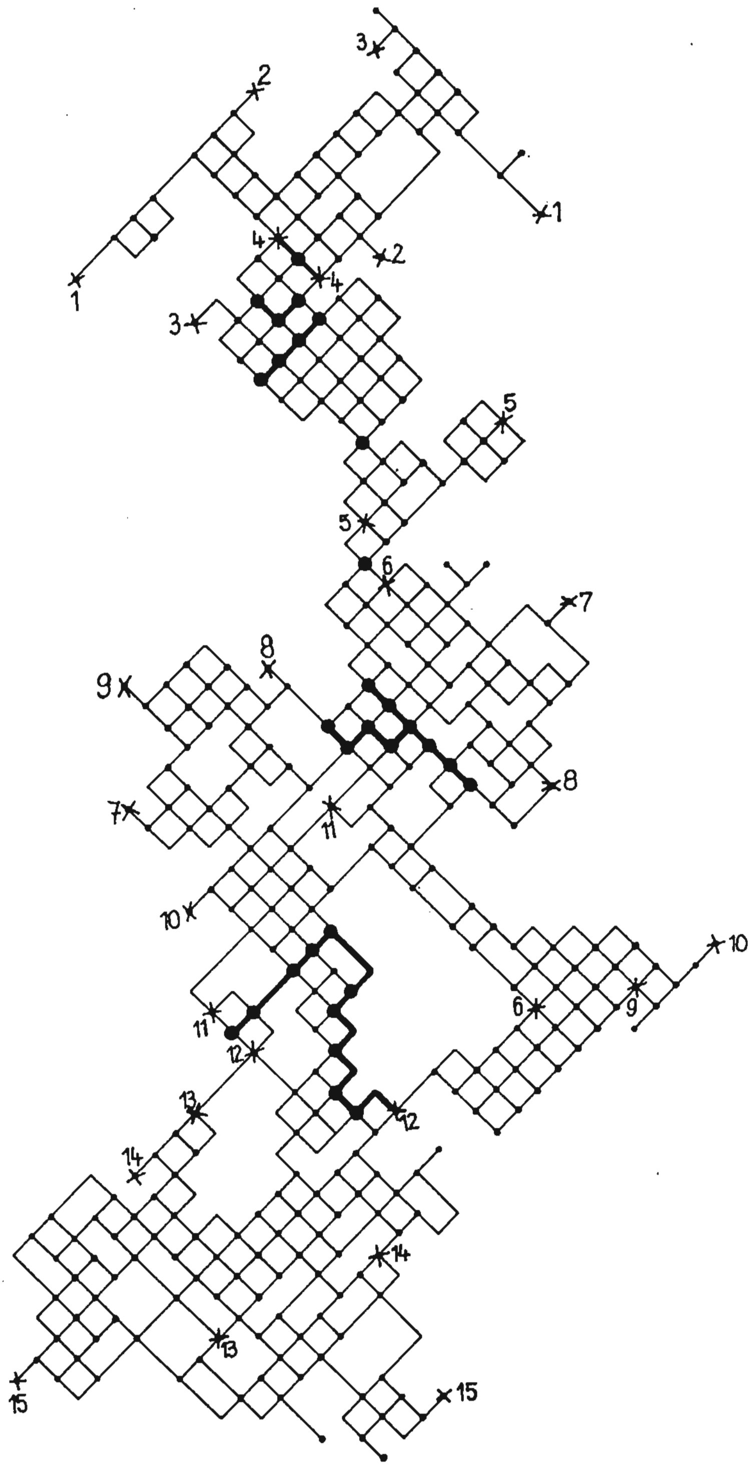


FIG. 6. Network with optimal borders (heavy lines).

In Figure 6 such optimal borders are shown, two of them represent already optimal paths to be determined. Figure 7 shows the individual subnets resulting by a corresponding choice of optimal borders. The optimal paths between nodes of one and the same subnet and between nodes of different subnets are marked by heavy-typed and dotted lines, respectively.

As already mentioned in special cases an extension of the procedure to directed graphs is possible. An independent decomposition of a digraph G can be achieved if we are able to find two paths $p(v_i,v_j)$ and p^+ (v_j,v_i) with reverse node order, whose elimination decomposes G into G_1 and G_2. In the case of digraphs we call such a pair of paths an optimal border. Analogously to the terms used in Section 2.1 we define the subnets $\overline{G}_1$ and $\overline{G}_2$ which result, if G_i $(i = 1, 2)$ is extended by all nodes of $p(v_i,v_j)$, by all arcs of $p(v_i,v_j)$ and p^+ (v_j,v_i) and by all arcs between nodes of G_i and nodes of $p(v_i,v_j)$. Then the optimization of paths between nodes of one and the same subnet $\overline{G}_i$ $(i = 1, 2)$ can be carried out in $\overline{G}_i$ independently of $\overline{G}_j$ $(i, j = 1, 2; i \neq j)$.

In conclusion of this subsection we point out that the method of independent decomposition is principally applicable to all graphs (or digraphs) which can be decomposed into two parts by the elimination of one optimal border. If the resulting subnets have this property, too, the original graph can be further decomposed by optimal borders. The most important cases in which an independent decomposition is possible are planar graphs. However, even if the networks representing layout possibilities in spatial regions with pronounced inner structure (in industrial sites, e.g.) are not planar, they often contain at least quiet large planar parts in which optimal borders can be determined.

For a further discussion of the efficiency of independent decomposition with respect to typical applications we refer to Section 4.

2.3. Analogy to Linear Algebraic Equations

The definition of a particular algebra revealed an analogy between the optimal path problem and the problem of solving linear algebraic equations [10]. A set $S = \mathbb{R} \cup \{\infty\}$ and two operations for all x, $y \in S$ were introduced:

$$\begin{array}{lll} x \oplus y = \min \{x,y\} & \text{(generalized addition)} & \\ x \otimes y = x + y & \text{(generalized multiplication)} & \end{array} \qquad (3)$$

The unit and null element in this algebra are 0 and ∞, respectively.

Obviously, the costs $d(x_i,x_j)$ (with x_i, $x_j \in Y(G)$ $(i, j = 1, \ldots, N)$) of an optimal path from x_i to x_j in digraph G, in the following called distance d_{ij}, satisfy the relation

$$d_{ij} = \begin{array}{ll} \min\, [d_{ik} + c_{kj}]; & i \neq j \\ 0^{k} \qquad\qquad ; & i = j \end{array} \qquad (4)$$

where c_{ij} are the costs of the arc from x_i to x_j and c_{ij} is infinite, if no arc from x_i to x_j exists.

By the extension of (3) to matrix operations, Eq. (4) can be written in the corresponding matrix form, that means in the form of the linear algebraic system

$$D = C \otimes D \oplus I \qquad \text{with } C = \{c_{ij}\} \text{ and } D = \{d_{ij}\} \qquad (5)$$

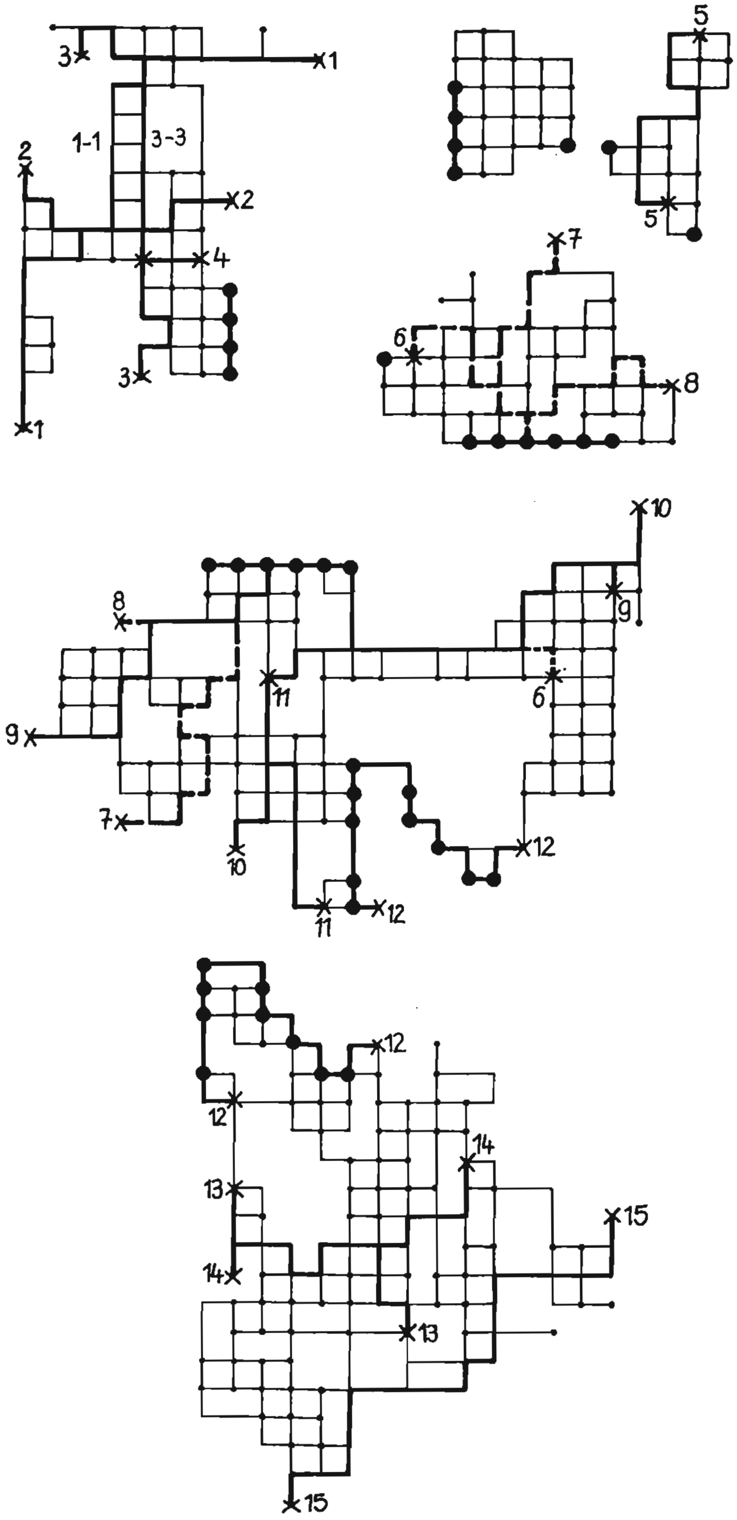


FIG. 7. Independent decomposition of the network shown in Figure 6; some optimal paths are marked.

where all matrices are square matrices of order N (number of nodes in G), and the main diagonal and off-diagonal elements of I are 0 and ∞, respectively.

As already mentioned in Sections 1 and 2, many practical path-design problems are associated with sparse networks. Considering the analogy of the optimal path problem to the solution of a linear algebraic system it is possible to utilize sparse matrix techniques in such cases.

In [8] two methods for the solution of the shortest path problem are presented. A computer program generates another program or an address table (set-up stage) which represents an optimal path algorithm that executes only nontrivial operations. The procedure has an algebraic analogy with the Crout elimination. Because the execution of the generated optimal path algorithm is very fast (for examples see [8]), the relatively high computational effort of the set-up stage can be accepted, if a network of fixed structure must be repeatedly treated with different arc costs.

In [9] the matrix of arc costs is assumed to be given in Block Triangular Form (BBTF). This matrix is decomposed into a lower triangular matrix and a matrix with nontrivial elements exclusively in the right border column. Then a modification algorithm is applied to the solution of a linear algebraic system corresponding to (5) with the mentioned triangular matrix as coefficient matrix.

3. GRAPH REDUCTION FOR THE OPTIMIZATION OF MORE COMPLEX CONNECTION STRUCTURES

In this section we consider the practical situation in which a connection structure of a certain class must be optimized within a large-scale network, whereas the optimized structure itself is located in a relatively small (of course a priori unknown) region of the entire net. In this case it is reasonable and can be highly efficient to eliminate in a first reduction step by some appropriate simple procedure as many nodes and edges from the graph as possible without loss of optimal solutions. Then some known method can be applied to the reduced graph in the optimization step.

3.1. Node-Disjoint Paths

At first we consider the determination of k node-disjoint paths ($k > 1$) with minimal total costs between two given vertices v_i and v_j of an undirected graph G (see also [19]). For the present we assume that an admissible solution, i.e., a set of k node-disjoint paths is already known. We denote the costs of the k given node-disjoint paths in such a way by $\tilde{d}(v_i,v_j,s)$ ($1 \leq s \leq k$) that the following inequality holds:

$$\tilde{d}(v_i,v_j,s) > \tilde{d}(v_i,v_j,r) \qquad \text{for } s > r \ (1 \leq s,\ r \leq k).$$

Now the following reduction steps can be carried out (only for simplicity of the description we consider undirected graphs):

1. All nodes with valence 1 and the incident edges are eliminated from G.
2. All nodes $v_s \in V(G)\backslash(\{v_i\} \cup \{v_j\})$ with valence 2 can be eliminated, if the incident edges are combined to one edge.

3. All edges $w \in E(G)$ with edge costs

$$c(w) > S \triangleq \tilde{d}(v_i,v_j,k) + \sum_{m=1}^{k-1} [\tilde{d}(v_i,v_j,m) - d(v_i,v_j)] \tag{6}$$

($d(v_i,v_j)$ – as above costs of an optimal path between v_i and v_j in G),

4. all nodes

$$v_s \in V(G)\backslash(\{v_i\} \cup \{v_j\}) \qquad \text{with } d(v_i,v_s) + d(v_s,v_j) > S, \tag{7}$$

and all incident edges are eliminated from G.

Steps 1 and 2 are independent of the selected admissible initial solution and can be applied additionally to the graph resulting after steps 3 and 4.

The efficiency of the method depends very much on the computational effort necessary for the determination of an admissible initial solution. In many practical applications it is reasonable to try to determine such a solution by a successive single path optimization where after each optimization step one of the calculated optimal paths is eliminated from the network. It should be noted that for $k = 2$, which is the most interesting case from the practical viewpoint, such a procedure—if successful—supplies the smallest bound in (6), namely, $S = \tilde{d}(v_i,v_j,2)$. Another reasonable method consists of the following steps:

—Definition of some more or less arbitrary S on the base of single optimal path costs.

—Graph reduction according to steps 1 to 4 above.

—Optimization of the k-fold node-disjoint connection on the reduced graph.

—If a solution exists, usage of its costs as a new bound in the reduction of G, which delivers a subnet that certainly contains the optimal solution of the original problem; otherwise new trial with increased S.

An analogous graph reduction procedure is proposed in [19] for the NP-complete problem of finding k node-disjoint paths with minimal total costs such that for a given B additionally

$$\tilde{d}(v_i,v_j,s) \leqslant B \qquad (1 \leqslant s \leqslant k)$$

holds.

3.2. Steiner Trees in Graphs

A similar methodology can be applied to the Steiner-tree problem in graphs (see also [1, 20]). Given an undirected graph $G = (Y,W)$ with positive edge weights (costs) and a subset $\bar{Y} \subset Y = V(G)$ of $M = |\bar{Y}| > 2$ nodes that must be contained in the Steiner tree.

At first we introduce a special sum $\tilde{S}$ of costs associated with $M - 2$ of the nodes of $\bar{Y}$. Let for all $v_i \in \bar{Y}$

$$c_{v_i}^{\min} = \min_{v_j \in F(v_i)} c(v_i,v_j) \qquad (v_i \in \bar{Y}), \tag{8}$$

where $c(v_i,v_j)$ and $F(v_i)$ are the weight of edge (v_i,v_j) and the set of direct neighbors of v_i in G, respectively. Using (8) we define

$$\tilde{S} = \min_{\substack{\bar{Y}_2 \subset \bar{Y} \\ |\bar{Y}_2| = 2}} \left(\sum_{v_i \in \bar{Y} \setminus \bar{Y}_2} c_{v_i}^{\min} \right). \tag{9}$$

Furthermore C_{opt} denotes the entire costs of an optimal solution of the Steiner tree problem.

Lemma 4. Let $\tilde{C}$ denote any upper bound of C_{opt}, i.e.,

$$\tilde{C} \geqslant C_{\text{opt}}. \tag{10}$$

Then holds: Each node $v_s \in Y \setminus \bar{Y}$, for which the distances (costs of optimal paths) $d(v_s,v_k)$ to all $v_k \in \bar{Y}$ obey the inequality

$$d(v_s,v_k) > \tfrac{1}{2}(\tilde{C} - \tilde{S}) \triangleq S_1 \tag{11}$$

cannot be contained in an optimal Steiner tree.

Proof. Assume v_s is contained in an optimal Steiner tree $T = (Y_T,W_T)$ and meets inequality (11) for all $v_k \in \bar{Y}$. Because $v_s \in Y \setminus \bar{Y}$ and positive edge costs are assumed, there must be at least two paths $p(v_s,v_m)$ and $p(v_s,v_n)$ $(v_m \neq v_n)$ in T which join exclusively v_s and connect v_s with some $v_m \in \bar{Y}$ and some $v_n \in \bar{Y}$, respectively, but do not contain any element of $\bar{Y}' = \bar{Y} \setminus (\{v_m\} \cup \{v_n\})$. Let $d_T(v_i,v_j)$ with v_i, $v_j \in Y_T$ denote the distance between v_i and v_j in T. Because this distance is never smaller than the distance between v_i and v_j in G, inequality (11) yields

$$\begin{aligned} d_T(v_s,v_m) &> S_1, \\ d_T(v_s,v_n) &> S_1. \end{aligned} \tag{12}$$

To construct a tree containing all elements of $\bar{Y}$ the nodes of $\bar{Y}'$ must be linked to the union of the node sets of $p(v_s,v_m)$ and $p(v_s,v_n)$. Obviously the contribution of these connections to the total costs of T is at least $\tilde{S}$ (see (9)).

Thus, according to (12) the costs C_T of T meet the inequality

$$C_T > 2 \cdot S_1 + \tilde{S} = \tilde{C}.$$

Because of (10) we get

$$C_T > C_{\text{opt}}$$

which is a contradiction to the assumption that T is an optimal Steiner tree. ■

In the following we propose procedures for the determination of a cost bound $\tilde{C}$ meeting inequality (10). These procedures compute costs of trees containing all elements of $\bar{Y}$ on the base of path optimization.

1. Determination of costs $C_{\bar{Y}}^{\text{MST}}$ of a minimal spanning tree in the complete graph on the vertices of $\bar{Y}$, for which the edge costs are given by the lengths of the corresponding shortest paths in G.

In contrast to the Steiner tree problem the minimal spanning tree problem can be solved in polynomial time.

At most $O(M \cdot N^2)$ with $M = |\bar{Y}|$, $N = |Y|$ operations are required. A well-known minimal spanning tree algorithm is proposed in [21].

2. Determination of costs $C_H^{(i)}$ of an approximate solution of the Steiner tree problem by one of the heuristic variants described in [1, 20, 22–24].

All these variants of a basic heuristic approach are mainly characterized by a stepwise extension of a tree: In each step one node of $\bar{Y}$ not yet in the tree is connected by an optimal path to the "nearest" vertex in the tree. The most time-consuming variant requires at most $O(M^2 \cdot N^2)$ operations (for more details see the above quoted references, especially [24]).

Because the costs determined by each of these procedures are not lower than the costs of the optimal solution of the Steiner tree problem, according to Lemma 4, $C_{\bar{Y}}^{\mathrm{MST}}$ or costs $C_H^{(i)}$ derived by variant i of the heuristic approach can be used as bound $\tilde{C}$ by (11) for network reduction.

Note that only $C_{\bar{Y}}^{\mathrm{MST}}$ or $C_H^{(i)}$, not the trees themselves are necessary for the decomposition of G into two parts.

Because for all of the mentioned heuristic variants

$$C_H^{(i)} \leqslant C_{\bar{Y}}^{\mathrm{MST}} \tag{13}$$

(see [23, 24]), the heuristics never yield a smaller degree of network reduction on the base of Lemma 4 than the minimal spanning tree (MST-) procedure. However, with respect to both computational and implementation effort for the determination of a cost bound $\tilde{C}$, the latter, too, has its right to exist. If, for instance, both the MST- and the heuristic Steiner tree algorithm are implemented in such a way that all optimal paths and corresponding costs potentially necessary for the tree construction are determined in a first stage, the computational and storage requirements of the MST-procedure are generally lower than those of the heuristics. Because in the case of the minimal spanning tree branchings are admissible exclusively in nodes of $\bar{Y}$, only $M(M - 1)/2$ optimal paths in G and corresponding costs must be computed and stored in this stage for the construction of the tree and the determination of its costs $C_{\bar{Y}}^{\mathrm{MST}}$, whereas $M(M - 1)/2 + M(N - M)$ paths and costs are necessary for the calculation of $C_H^{(i)}$. Furthermore using special purpose software for the determination of $C_{\bar{Y}}^{\mathrm{MST}}$ even in the course of the MST-procedure only the costs of optimal paths, not the paths themselves, are required, whereas in the case of the Steiner tree heuristics certain optimal paths must be calculated and stored in the course of the tree determination.

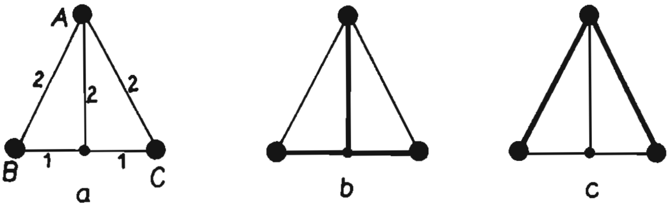


FIG. 8. (a) Graph G and $\bar{Y} = \{A,B,C\}$; (b) optimal Steiner tree with costs $C_{\mathrm{opt}} = 4$; (c) a solution of the MST-problem (determined in the complete graph on $\bar{Y}$) with $C_{\bar{Y}}^{\mathrm{MST}} = 4$.

To achieve a high degree of network reduction according to Lemma 4 we are interested in a cost bound $\tilde{C} \geqslant C_{\text{opt}}$ being as small as possible. Therefore the following question arises:

How much larger than C_{opt} can be $\tilde{C}$, if its value is delivered by the procedures proposed above?

The answer to this question is given by the following Theorem.

Theorem 1. If the value of $\tilde{C} \geqslant C_{\text{opt}}$ is given by $C_{\bar{Y}}^{\text{MST}}$ or by $C_H^{(i)}$ (where the upper index i specifies some variant of the class of Steiner tree heuristics described in [24], then

(1) $$\frac{\tilde{C}}{C_{\text{opt}}} \leqslant 2\left(1 - \frac{1}{M}\right)$$

and (14)

(2) there is no $B > 1$ such that always
$$\frac{\tilde{C}}{C_{\text{opt}}} \geqslant B.$$

Proof. Because of

$$\frac{C_{\bar{Y}}^{\text{MST}}}{C_{\text{opt}}} \leqslant 2 \cdot \left(1 - \frac{1}{M}\right)$$

(see [23, 24]) and inequality (13) the first statement holds. The existence of Steiner tree problems for which $C_{\text{opt}} = C_{\bar{Y}}^{\text{MST}}$ (see Fig. 8) proves the second statement. ■

Figures 9 and 10 show an example for the Steiner tree optimization using the proposed method for the reduction of G with $\tilde{C} = C_{\bar{Y}}^{\text{MST}}$.

In Figure 9 the result of the MST-procedure is marked. The costs of this tree connecting all nodes of $\bar{Y}$ are $C_{\bar{Y}}^{\text{MST}} = 13$. Because according to (9), $\tilde{S} = 2$, we have in this example $S_1 = 5, 5$.

Figure 10 shows the reduced graph certainly containing the optimal Steiner tree. The exact solution of the problem is marked.

3.3. An Extension of the Steiner Tree Problem in Graphs

Now we treat network reduction for an extension of the Steiner tree problem in graphs in which additional costs for branchings arise which are dependent on the degree of branching. Formally this optimization problem can be stated in the following way.

Steiner tree problem with branching-degree dependent costs (STBC-problem):

Given

—an undirected graph $G = (Y, W)$ with positive edge weights (costs),

—a subset $\bar{Y} \subset Y = V(G)$ of $M = |\bar{Y}| > 2$ nodes,

—for any subgraph T of G, a function $\deg_T(v)$ assigning to each node $v \in V(T)$ the number of its incident edges in T, and a cost function $c_B(v,b)$, with $c_B(v,b') \geqslant c_B(v,b)$

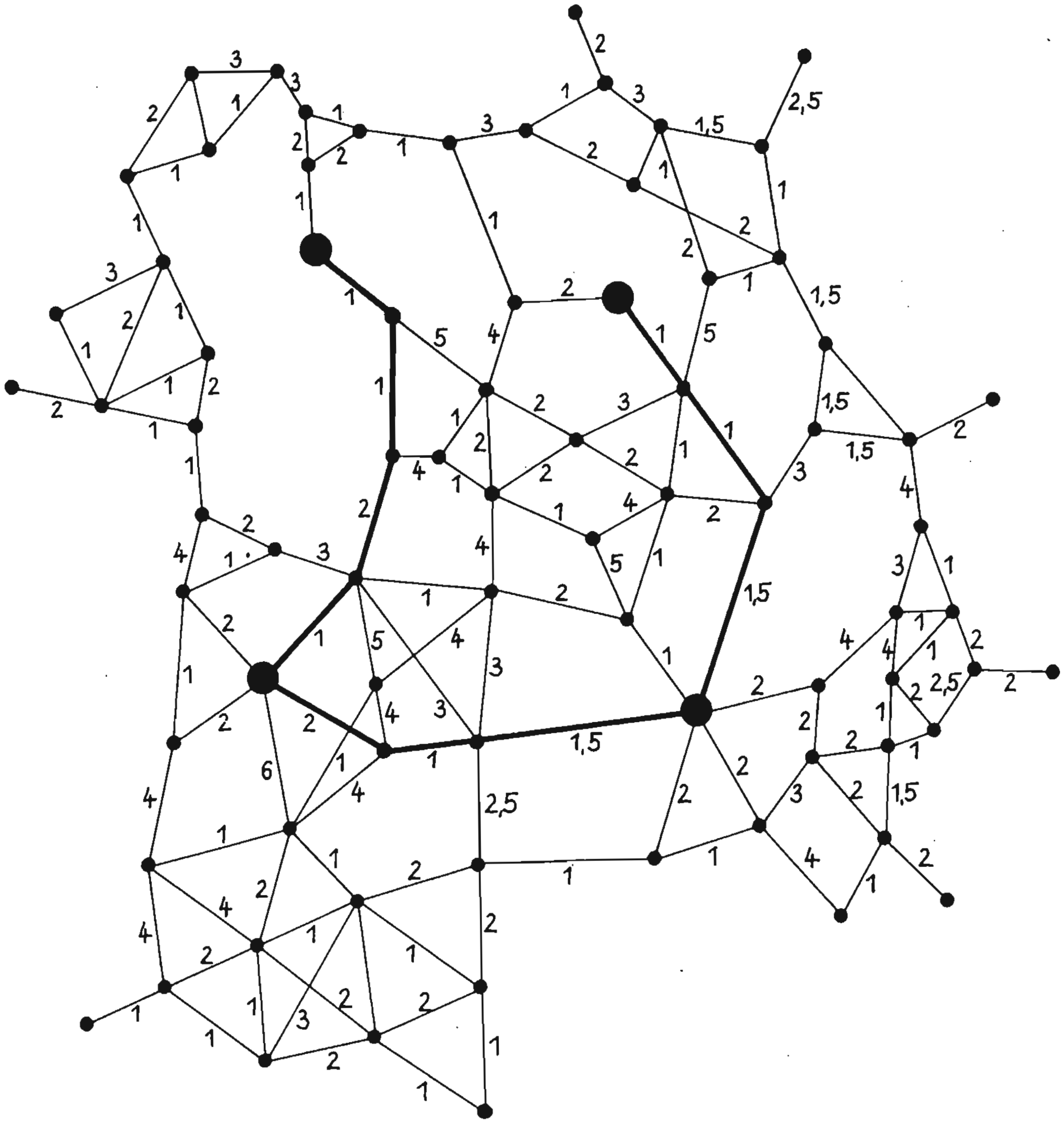


FIG. 9. Graph $G = (Y, W)$ with four marked nodes (composing $\overline{Y}$) that have to be connected by an optimal Steiner tree.

for $b' \geqslant b$, assigning to each $v \in V(T)$ and each $b \in \mathbb{N}$ with $2 < b \leqslant \deg_G(v)$ a nonnegative real number.

Find a subgraph of G with the minimum costs among all connected subgraphs that contain $\overline{Y}$, where the costs of a subgraph T are the sum of the costs of edges in T increased by the branching-degree dependent term

$$\sum_{v \in V_B(T)} c_B(v, \deg_T(v)),$$

where $V_B(T) = \{v | v \in V(T) \wedge \deg_T(v) > 2\}$.

This minimum cost subgraph will be called Steiner tree with branching-degree dependent costs (STBC) and the minimum total costs are denoted by $C_{\text{opt}}^{\text{STBC}}$ in the following.

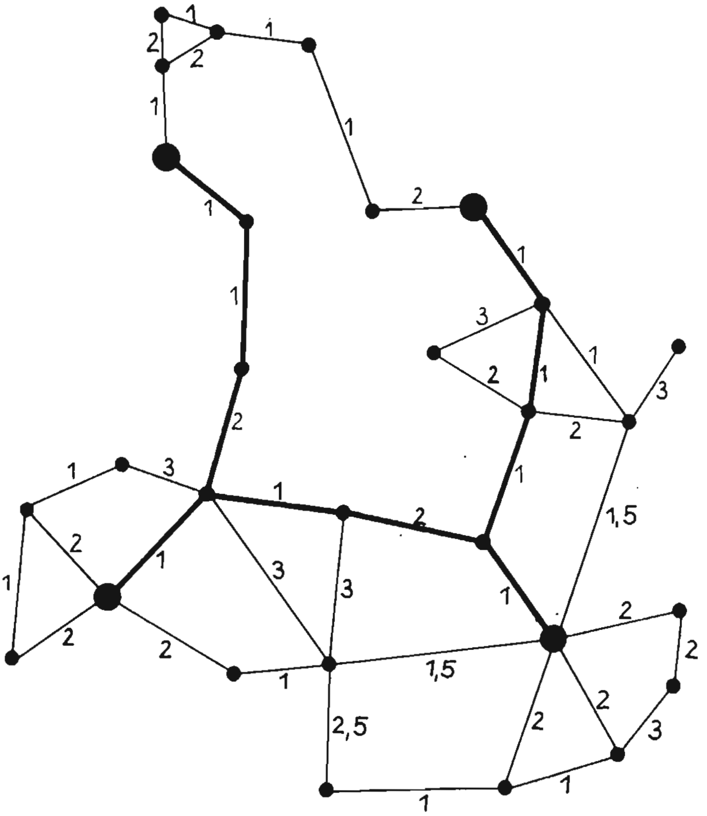


FIG. 10. Result of the graph reduction using Lemma 4 in the case of the example presented in Figure 9.

Again network reduction can simplify the solution of the problem. A Lemma similar to Lemma 4 will be used for this purpose.

Lemma 4a. Let $\tilde{C}^{\text{STBC}}$ denote any upper bound of $C^{\text{STBC}}_{\text{opt}}$, i.e.,

$$\tilde{C}^{\text{STBC}} \geqslant C^{\text{STBC}}_{\text{opt}} \tag{10a}$$

Then holds: Each node $v_s \in Y \backslash \overline{Y}$, for which the distances $d(v_s, v_k)$ to all $y_k \in \overline{Y}$ obey the inequality

$$d(v_s, v_k) > \tfrac{1}{2}(\tilde{C}^{\text{STBC}} - \tilde{S}) \triangleq S_1^{\text{STBC}*} \tag{11a}$$

cannot be contained in the STBC.

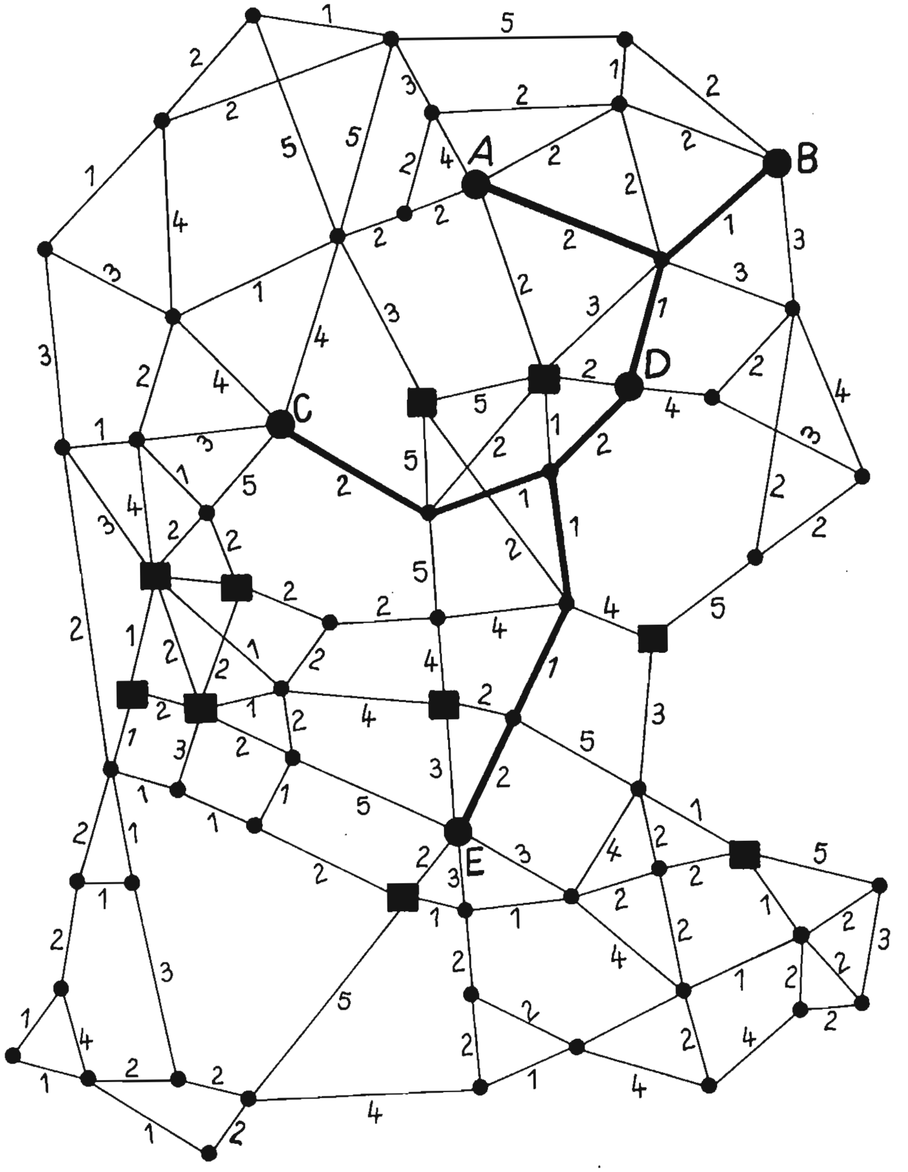
The proof is similar to that of Lemma 4 and therefore left to the reader.

In the following we propose simple procedures for the derivation of a cost bound $\tilde{C}^{\text{STBC}}$.

1. Determination of the minimum among the costs of all stars connecting the nodes of $\overline{Y}$ and with their centers placed in an element of this set (at most $O(M \cdot N^2)$ operations required).

2. Determination of $C_H^{(i)}$ and a corresponding tree T_i as before, and calculation of

*$\tilde{S}$ is given by (9).



● nodes $v \in \overline{Y}$; $(\forall v \in \overline{Y} \quad c_B(v,d) = 4 \text{ for } 2 < d \leq \deg_G(v))$

• nodes $v \in Y \setminus \overline{Y}$, for which $c_B(v,d) = 2$ for $2 < d \leq \deg_G(v)$

■ nodes $v \in Y \setminus \overline{Y}$, for which $c_B(v,d) = 0$ for $2 < d \leq \deg_G(v)$

FIG. 11. Example of a graph $G = (Y,W)$ with set $\overline{Y} = \{A,B,C,D,E\}$ whose nodes have to be connected by a Steiner tree with branching-degree dependent costs (STBC). An admissible solution of the STBC-problem is marked.

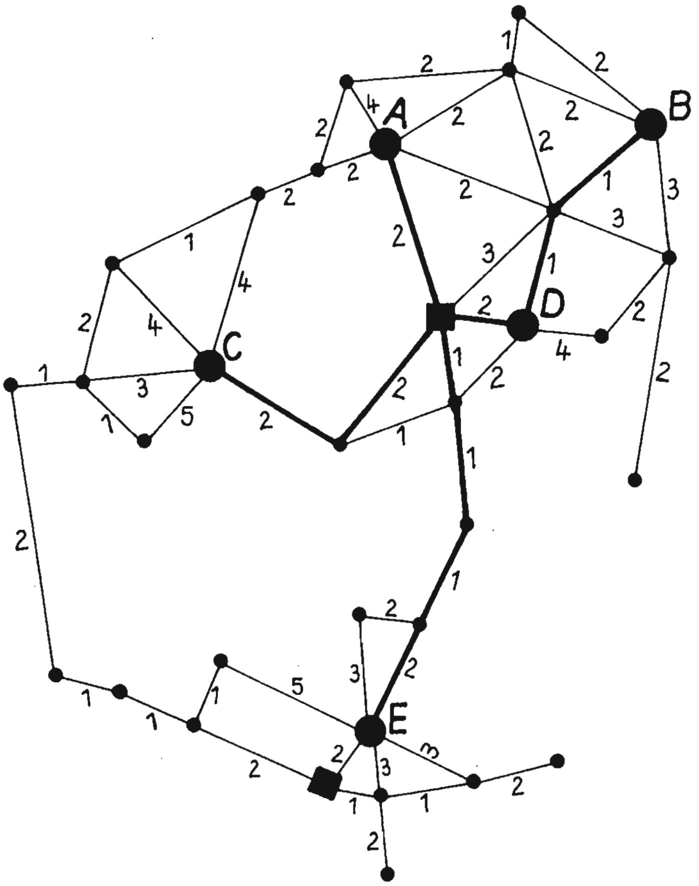


FIG. 12. Result of the network reduction by usage of Lemma 4a for the example of Figure 11 ($S_1^{\text{STBC}} = \frac{1}{2}(17 - 4) = 6, 5$). The optimal solution of the STBC-problem specified in Figure 11 is marked.

the sum

$$C_H^{(i)} + \sum_{v \in V_B(T_i)} c_B(v, \deg_{T_i}(v)).$$

The first procedure yields the minimum among the costs of all subgraphs connecting the vertices of $\bar{Y}$ and containing only one branching node with minimal degree, the others emphasize the minimization of branching independent costs. Because all mentioned procedures determine costs of admissible solutions of the STBC-problem, these costs can be used as bound $\tilde{C}^{\text{STBC}}$ in Lemma 4a for network reduction.

Figures 11 and 12 illustrate the procedure by an example. Graph G, node set $\bar{Y}$ and an approximate solution of the ordinary Steiner tree problem are shown in Figure 11. The marked approximate solution with edge costs of 13 results by that variant of the basic heuristic approach which was presented in [1]. The additional costs for branchings in this tree are 4 (see Fig. 11). Thus we can use $\tilde{C}^{\text{STBC}} = 17$ as cost bound in Lemma 4a. Because $\tilde{S} = 4$ for the example, the right side of (11a) is $S_1^{\text{STBC}} = 6, 5$. Application of Lemma 4a to G yields the reduced graph presented in Figure 12. Of course this graph could be further simplified, for instance by a successive elimination of nodes with valence 1. The exact solution with $C_{\text{opt}}^{\text{STBC}} = 15$ is marked in Figure 12. This solution can be determined by a simple extension of the exact optimization approach presented in [1] for the ordinary Steiner tree problem.

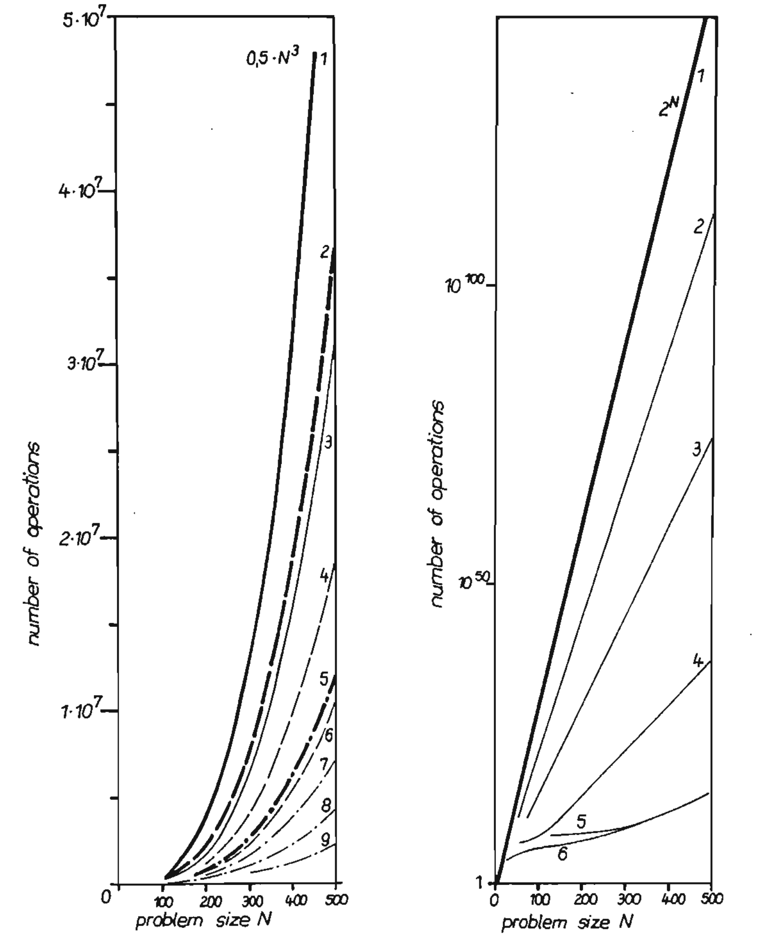
4. COMPARISONS AND CONCLUSIONS

It is a common practice to characterize algorithms by the number of necessary operations. In complex problem-solving processes as considered in this paper such a single parameter characterization can be rather uninteresting or even misleading. In [4] and [9], for instance, the computational effort for the determination of appropriate cut sets and for the transformation of the matrix of arc costs to Bordered Block Triangular Form (BBTF), respectively, is not taken into account. Furthermore in [4] the efficiency of the algorithm is discussed for a cascade decomposition with an equal number of nodes in all cut sets and an equal number of nodes in all subnets, which is generally no typical situation. The methods described in [8] need relatively high computational effort in the set-up stage, but after the evaluation of the structural properties the execution is very fast. In the case of independent decomposition it is impossible to fix the optimal borders arbitrarily. Therefore also the subnets resulting by this decomposition method cannot be given in advance. The same is true for the graph reduction approach presented in Section 3. The computational effort for these decomposition techniques is low because only single paths or other simple connection structures which can be optimized in polynomial time are necessary.

The choice of a method depends very much on the practical application. In some situations (as those mentioned in Sections 1 and 2) the entire network is already described in parts. Therefore the algorithms presented in [4, 5] can be used immediately. On the other hand, the BBTF is generally not given a priori.

Independent decomposition is very suitable in applications in which a high degree of independence between subnets is desired. This is especially important in those practical situations which lead to the optimization of paths between nodes that are clustered in relatively small parts of an entire large-scale network. By independent decomposition it is possible to eliminate network parts that are irrelevant for the optimization from further considerations. This also holds for the graph reduction methods described in Section 3. But in contrast to independent decomposition the relevant network part is not determined by a direct construction of its borders but by the usage of cost bounds.

The computational advantages of these methods are quantitatively illustrated by Figure 13. In Figure 13a the three marked curves represent approximatively the computational effort for the determination of $R \cdot N$ optimal paths in a graph with N nodes by some algorithm requiring $O(N^2)$ operations (in the worst case) and not utilizing decomposition methods. The other curves in Figure 13a stand for the corresponding computational effort for those cases in which by independent decomposition, i.e., by the determination of some optimal paths, a subnet with $r \cdot N$ $(R \leqslant r \leqslant 1)$ nodes results, to which the search for the $R \cdot N$ optimal paths can be restricted. This computational effort consists of two additive terms. The first $(\alpha \cdot N^2)$ regards the effort for the determination of the necessary optimal borders by some shortest path algorithm. The second term $(R \cdot N \cdot (r \cdot N)^2)$ represents approximatively the computational effort for the determination of the $R \cdot N$ optimal paths in the subnet. Note that for sufficiently large N the curves are practically independent of α.



reference number	function type	parameters
1	$R \cdot N^3$	$R = 0,5$
2	—"—	$R = 0,3$
3	$\alpha N^2 + RN(rN)^2$	$R = 0,5; r = 0,7; \alpha = 5$
4	—"—	$R = 0,3; r = 0,7; \alpha = 5$
5	$R \cdot N^3$	$R = 0,1$
6	$\alpha N^2 + RN(rN)^2$	$R = 0,3; r = 0,5; \alpha = 5$
7	—"—	$R = 0,1; r = 0,7; \alpha = 5$
8	—"—	$R = 0,1; r = 0,5; \alpha = 5$
9	—"—	$R = 0,1; r = 0,3; \alpha = 5$

◄ (a)

reference number	function type	parameters
1	2^N	–
2	$\alpha N^3 + 2^{(r \cdot N)}$	$r = 0,75$
3	—"—	$r = 0,5$
4	—"—	$\alpha = 50; r = 0,25$
5	—"—	$\alpha = 50, r = 0,1$
6	—"—	$\alpha = 1; r = 0,1$

▲ (b)

FIG. 13. Reduction of the computational effort by independent decomposition (a) and graph reduction (b).

As already mentioned the reduction of the computational effort by decomposition methods can be much higher, if the optimization regards more complex structures than shortest paths. Only as one demonstration of this fact Figure 13b illustrates the situation for a typical case in which the computational effort for the determination of an exact solution without utilizing decomposition possibilities is proportional to 2^N (see marked curve in Figure 13b). The other curves correspond to $\alpha \cdot N^3 + 2^{(r \cdot N)}$, where the first term $\alpha \cdot N^3 = \alpha' \cdot (R \cdot N) \cdot N^2$ stands for the number of operations required by a graph reduction method based on the determination of $R' \cdot N$ shortest paths, and the second term represents the computational effort for the optimization in the reduced graph with $r \cdot N$ nodes. Curves 2 and 3 are practically independent of α in the region in which they are drawn. Note that the scales of the ordinates in Figures 13a and b are different.

Besides its computational efficiency the main advantage of these decomposition techniques lies in the fact that we achieve independence of optimization processes in certain subnets. As a consequence of this the results of independent decomposition and graph reduction are dependent on edge costs. This is a disadvantage in such situations in which optimization has to be carried out many times in a graph with changing numerical values of costs, but invariable structure. Such applications occur in traffic control problems. If the execution of each optimization after a change of costs is so fast as described in [8], in these situations one may accept high implementation and computational effort for a set-up stage and use decomposition techniques that utilize only structural properties of the entire network.

Finally Table II presents in an overview main characteristics of the methods described or mentioned in this paper, their integration in the overall problem solving process including man–machine interaction, and typical applications.

5. FORMULATION OF A GENERAL OPTIMIZATION PROBLEM

In this final section a new, relatively general problem formalization is presented which regards the optimal layout of networks in environments with pronounced inner structure. This formulation (see also [1]) is based on two models, one for the inner structure of a two- or three-dimensional region, in which a netlike system has to be laid out, and one for the system itself. We model this system as a graph P. In the case of a material system the vertices represent its components and the edges connections between them (e.g., cables). In the context of transport optimization problems, for instance, P may represent an abstract structure of activities, where the vertices may correspond to maintenance, repair, and parking of a vehicle at generally not yet localized depots, as well as to starting, intermediate stopping, and terminating in (generally not yet fully specified) points. In this case P is a digraph and the arcs represent vehicle movements between the activities corresponding to the adjacent vertices.

According to the arguments of Section 1 we model the inner structure of the region, in which the layout has to be performed, also by a graph (generally a digraph). The edges of this graph—in the following called G—represent possibilities of connection line routing, the vertices may correspond to possible crossings and branchings of route segments, to points of possible placement of system components, and to spatially fixed connections to other systems already existing.

Furthermore, we introduce a set Q of admissible system structures. In the most simple case this set contains only one element whose characteristics are explicitly given. However, in many practical applications only a global structure for the system or parts of it is defined. This is the case, for instance, if a part of the system has to be a ring, connecting some components in a sequence not specified a priori. An admissible net layout solution is given by an admissible assignment of vertices and arcs of $P \in Q$, respectively, to vertices and paths in G, respectively. By this mapping (called ψ in the following) costs result. We look for that pair (P,ψ) with $P \in Q$ and admissible ψ for which the objective function is optimal.

TABLE II. Main characteristics of decomposition methods described or mentioned in this paper.

Reference number	Method	References	Main characerization	Demanded properties of the entire graph	Characteristics of the subnets	Applications	Implementation effort	Computational effort and torage requirements	Influence of the problem solving by man
M1	Determination of optimal paths in networks already decomposed by arbitrary choice of cut sets	Section 2.1 of this paper; [4, 5]	Successive computation of conditional opt. paths on subnets resulting by some choice of node cut sets	• Large and loosely connected; • Easy choice of cut sets by inspection	• Not possible to determine opt. paths between two nodes of one and the same subnet independently of the other subnets • Any specification of subnets possible	Determination of all (or at least many) optimal paths in the network	Very low, because the method doesn't consider an algorithmic determination of cut sets	Low; possible to treat only one subnet at a time	Choice of cut sets
M2	Optimal decomposition	Section 2.2 of this paper; [6, 7]	For given number of subnets and given size restriction the algorithm based on dynamic programming determines node cut sets in which union the overall number of elements is minimal	Large and loosely connected	• As first point for M1 • Number of nodes in each of the m subnets $\leq K$ • Number of elements in the union of all cutsets is minimal with respect to given m and K • Independence of the edge costs	As M1	Additionally to the programs used for M1 the optimal decomposition algorithm must be implemented	• High additional computational effort (with respect to M1) for the optimal decomposition which reduces on the other side the computational work associated with equ. (1) • The decomposition alg. treats the entire graph	Selection of nodes as "crystal nuclei" for the subnets; in this case the decomposition is optimal only with respect to this initial guess

TABLE II. *(Continued from previous page.)*

Reference number	Method	References	Main characterization	Demanded properties of the entire graph	Characteristics of the subnets	Applications	Implementation effort	Computational effort and torage requirements	Influence of the problem solving by man
M3	Independent decomposition	Section 2.2 of this paper	The entire graph is decomposed by optimal paths	• Large and loosely connected • Easy choice of the extremes of opt. borders possible (this is ensured for planar graphs)	• Independent in the sense of lemma 3 • Dependent on edge costs • No complete control of subnet and cut set sizes	• as M1 • determination of opt. paths between nodes clustered in a relatively small region	Very low; the overall procedure is based only on some standard optimal path algorithm	• Low computational effort • Opt. paths between nodes of one subnet can be determined exclusively on this subnet • The decomposition alg. treats the entire graph	• Selection of extreme nodes of optimal borders • Selection of opt. borders from a set offered by the computer
M4	Sparse matrix techniques	Section 2.3 of this paper [8, 9]	The analogy between the optimal paths problem and the solution of a set of linear algebraic equations is utilized	As M2	• Independent of edge costs • As first point for M1	as M1	Relatively high; not based on standard optimal paths algorithms	• High computational effort for the code generation technique [8] applied to Crout elimination (setup stage); execution stage extremely fast in this case	

								• Very fast calculation of distance matrix for cost matrices already given in Bordered Block Triangular Form
M5	Graph reduction	Section 3 of this paper; [19]	Elimination of such nodes and edges that cannot be contained in the optimal solution	Large	Only two subnets result; one of them is completely irrelevant for the optimization	• Optimization of Steiner trees • Optimization of a set of disjoint paths, if the optimal structure is located in a relatively small region of the network	For the mere graph reduction low, because optimization of only simple connection structures is required	• For the mere graph reduction low computational effort • The graph reduction alg. treats the entire network • Optimization of the desired structure on a subnet

More formally this general problem can be stated in the following way:

Given a digraph $G = (Y, W)$ with the sets of vertices and arcs $Y = V(G)$ and $W = E(G)$, respectively. In the following we consider only digraphs in which the total costs of any circuit are nonnegative.

Furthermore a set

$$U = \{A_i | A_i \subseteq Y\}_{i \in J} \text{ with the index set } J \subset \mathbb{N} \tag{15}$$

and

$$\bigcup_{i \in J} A_i = Y,$$

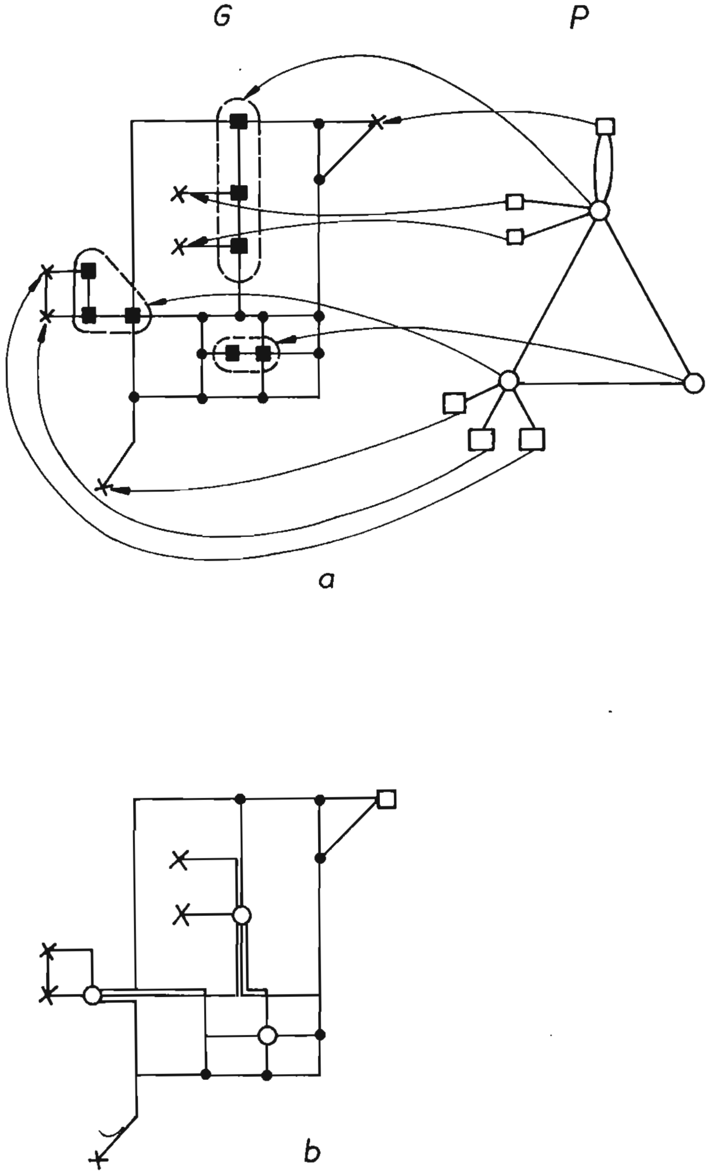


FIG. 14. (a) A simple example of graphs G and P and the relation $\mathfrak{R}$, (b) realization of an admissible mapping.

where the sets Ω, ω and the functions α and β are given. With these definitions the optimization problem can be formulated in the following way:

Given a finite digraph G, a set U according to (15), a set Q of digraphs (admissible system structures). Find a pair (P,ψ) with $P \in Q$ and admissible ψ,· for which the (generally multidimensional) objective function

$$Z(P,\psi) = Z(\{\Phi_r\}_{r\in W}, \{\varphi^{-1}(y)\}_{y\in Y}) \tag{21}$$

becomes optimal.

Figure 14 illustrates the problem. The arrows represent the relation $\mathfrak{R}$ between X_1 (consisting of all elements of X in this case) and U.

Obviously some of the classical optimization problems in graphs, as the determination of optimal paths, belong to the just described class of network layout problems.

References

[1] O. Taraszow and A. Iwainsky, Trassierung und Plazierung auf Graphenmodellen. Vortrags-Sammelband der 4. Fachtagung Numerische Realisierung Mathematischer Modelle; Zingst, 21.–25. 9. 1981; ZfR-81.16, pp. 215–231.

[2] O. Taraszow, Optimal layout of system structures-problem formalization and directions of research. In preparation.

[3] S. Döring, A. Iwainsky, and P. Richter, Network layout in an industrial site—a case-study. III. Bilateral Meeting GDR–Italy on Advances in Informational Aspects of Industrial Automation; Berlin, 19th–21st February 1984, pp. 145–166.

[4] T. C. Hu, A decomposition algorithm for shortest paths in a network. *Operations Res.* **16** (1968) 91–102.

[5] J. Y. Yen, On Hu's decomposition algorithm for shortest paths in a network. *Operations Res.* **19** (1971) 983–985.

[6] J. G. Lee, W. G. Vogt, and M. H. Mickle, Optimal decomposition of large-scale networks. *IEEE Trans. Systems, Man Cybernetics* **SMC-9** (1979) 369–375.

[7] J. G. Lee, W. G. Vogt, and M. H. Mickle, Calculation of the shortest paths by optimal decomposition. *IEEE Trans. Systems, Man, Cybernetics* **SMC-12** (1982) 410–415.

[8] S. Goto, T. Ohtsuki, and T. Yoshimura, Sparse matrix techniques for the shortest path problem. *IEEE Trans. Circuits Systems* **CAS-23** (1976) 752–758.

[9] S. Goto, and A. Sangiovanni-Vincentelli, A new decomposition algorithm for the shortest path problem. Proceedings of 1979 ISCAS.

[10] B. A. Carre, An algebra for network routing problems. *J. Inst. Math. Appl.* **7** (1971) 653–656.

[11] J. Y. Yen, Shortest path network problems. *Mathematical Systems in Economics 18;* Verlag Anton Hain, Meisenheim am Glan (1975) 273–294.

[12] I. M. Soi and K. K. Aggrarwal, On shortest-path algorithms in the topological design of computer networks; a comparative study. *Int. J. Systems Sci.* **12** (1981) 1379–1387.

[13] E. W. Dijkstra, A note on two problems in connection with graphs. *Numerische Math.* **1** (1959) 269–271.

[14] R. Bellmann, On a routing problem. *Q. Appl. Math.* **16** (1958) 87–90.

[15] J. Y. Yen, An algorithm for finding shortest routes from all source nodes to a given destination in general networks. *Q. Appl. Math.* **27** (1970) 526–530.

[16] R. W. Floyd, Algorithm 97—Shortest path. *Commun. ACM* **5** (1962) 345.

[17] P. M. Spira, A new algorithm for finding all shortest paths in a graph of positive arcs in average time $O(n^2 \cdot \log^2 n)$. *SIAM J. Comput.* **2** (1973) 28–32.

[18] P. Richter and O. Taraszow, Random graph generation. In A. Iwainsky (Ed.), *Optimization of Connection Structures in Graphs.* Preprints of the Centr. Inst. of Cyb. a. Inform. Processes, Berlin (1985) 213–225.

[19] S. Döring and A. Iwainsky, Graph reduction for the optimization of node-disjoint paths in networks. In A. Iwainsky (Ed.), *Optimization of Connection Structures in Graphs.* Preprints of the Centr. Inst. of Cyb. a. Inform. Processes, Berlin (1985) 177–201.

[20] O. Taraszow and P. Richter, New heuristic algorithms for solving the Steiner tree problem in graphs. In preparation.

[21] R. C. Prim, Shortest connection networks and some generalizations. *Bell System Tech. J.* **36** (1957) 1389–1401.

[22] H. Takahashi and A. Matsuyama, An approximate solution for the Steiner problem in graphs. *Math. Japonica* **24** (1980) 573–577.

[23] A. Iwainsky, Some notes on the Steiner tree problem in graphs. In A. Iwainsky (Ed.), *Optimization of Connection Structures in Graphs*. Preprints of the Centr. Inst. of Cyb. a. Inform. Processes, Berlin (1985) 57–73.

[24] A. Iwainsky, and A. Neumann, Some variants of a heuristic approach to the solution of the Steiner tree problem in graphs. In preparation.

Received July, 1983
Accepted October, 1985