論文 84-21-6-13

# A Network Partitioning Using the Concept of Connection Index-Algorithm and Implementation

# (連結指數의 概念을 使用한 回路網分割 - 알고리즘 및 實施)

朴 鎭 燮*, 朴 松 培**

(Chin Sup Park and Song Bae Park)

要 約

하중 그래프(weighted graph)의 연결지수(connection index)라는 새로운 개념에 기초를 둔 회로망분할에 대한 O(v.e) -v, e는 각각 마디 및 가지의 수-의 새로운 효과적 휴리스틱(heuristic) 알고리즘을 제안하였다.

실험적 결과는 이 알고리즘이 매우 효과적이고 시험한 다수의 분할문제에 대하여 최적의 또는 거의 최적의 해를 준다는 것을 실증하고 있다. 제안된 알고리즘의 몇 가지 응용을 제안하였고 그 컴퓨터 프로그램을 상세히 기술하였다.

**Abstract**

Based on a new concept of connection index of a weighted graph, a new efficient heuristic algorithm of $O(v.e)$ for network partitioning is presented, where v and e are the number of nodes and edges, respectively. Experimental results show that our algorithm is very efficient and yields an optimal or near optimal solution for a number of partitioning problems tested. Some applications of the proposed algorithm are suggested and its computer implementation is described in detail.

## I. Introduction

Partitioning and piecewise approaches are extensively used in computer aided design (CAD) of large-scale networks and systems. In circuit theory these approaches care called variously – diakoptic analysis, generalized hybrid analysis and node tearing analysis[2, 13, 14, 26], which are, however, essentially the same from the modified model analysis[15] point of view.

In the automatic design of printed circuit board (PCB) and integrated circuit (IC) layout, the procedure can be divided into three steps: partitioning, placement and routing. The first and most important step, partitioning, implies tearing apart a given network into many subnetworks, and optimal partitioning generally means minimization of the coupling between the subnetworks. Partitioning of an undirected graph belongs to the NP-complete class.[16]

*正會員, 韓國電子技術研究所

(Korea Inst. of Electronics Tech.)

**正會員, 韓國科學技術院 電氣 및 電子工學科

(Dept. of Electrical Engineering, KAIST)

接受日字 : 1984年 3月 28日

For NP-complete problems there does not exist an algorithm of polynomial bound and all of them except for isomorphism problems are polynomially equivalent.[17]

In partitioning of graphs, after drawing the associated graph of the problem, human may exercise the ability of determining a fairly good partitioning by laborous visual inspection. However, for large networks a rigorous algorithm must be developed to systematically tear apart the associated graph into optimal or near optimal partitions.

A number of partitioning algorithms have been proposed in the past in various fields. The hitherto published algorithms may be classified as follows from various points of view.

*1. Utility*

1) Network and system decomposition [1,3,5, 6,9,11]
2) PCB and IC layout [4]
3) Computer program and logic [19,20]
4) Statistical data grouping [20]

*2. Objective of minimization*

1) Number of interconnection nodes [1,5,9]
2) Number of interconnection branches[3, 11]
3) Total cost of interconnection branches [4]
4) Distance between the centroids of partitions [20]

*3. Methodology*

1) Growing clusters of subnetwork minimal group [3]
   cliques [5]
   strong component in digraph [6]
2) Finding the contour of a graph
   contour approach [1]
   SBP approach [11]
3) Interchanging nodes until some local optimallity condition is satisfied [4]
4) Transforming the problem into some associated mathematical equation [20] or finite automata [7]
5) Maximum flow minimum cut [22]

Most of the above methods do not yield an efficient heuristic algorithm[23] and some of them are of little practical use. One common difficulty is the seed selection effect. The partitioning algorithm to be presented in this paper is based on a new concept of connection index and almost elimimates the seed selection effect, although a greedy strategy similar to the contour approach[1] is used.

In section II, connection cost and connection index of a section graph are first defined. A condition for the contractable section graph (CSG) is then suggested and, using it the CSG's are found by breath first search and collapsed to supernodes. This process is repeated until the final contracted graph with N nodes, each with a weight not exceeding a given partition size, is obtained, where N is a given number of partitions. In Section III is introduced the new partitioning algorithm, while a sufficiently detailed description of the program is given in Appendix. In Section IV two examples are given to illustrate the proposed algorithm. The computational complexity of the above algorithm is analyzed in Section V. The algorithm is tested for a number of network partitioning problems and the results, as given in Section VI, show that it is very efficient and yields an optimal or near optimal solution to all of the tested problems. Finally, application of the proposed partitioning algorithm to two network problems is suggested.

## II. Definitions and Theorems

Before describing the new algorithm we give definitions of some terminologies and notations. We will also introduce some useful theorems.

Geneally a graph is denoted as G(V,E), where V is a finite non-empty set of nodes and E is a set of edges. A graph in which weight is assigned to both or either of its nodes and edges is called a *weighted graph,* and a graph in which a direction is assigned to every edge is called a *digraph* (directed graph). In this paper we are concerned with weighted *undigraphs* (undirected graphs). (However, an undigraph will be converted to a symmetric digraph in storing the graph data for the convenience of computer implementation of the algorithm). A graph which contains neither a self loop nor a parallel edge, is called a *simple graph.*

*Defintion 1. Section Graph*

A section graph $G_s(V_P,E_P)$ associated with a node subset $V_P$ of $V$ is a subgraph of $G(V,E)$ consisting of $V_P$ and all edges interconnecting the nodes in $V_P$. Sometimes, $G_s(V_P,E_P)$ will be denoted simply as $G_s(V_P)$.

*Defintion 2. Contractable Section Graph*

A section graph whose components are not parted during some stage of partitioning operation is called a contractable section graph (CSG) in that partitioning stage.

*Definiton 3. Weighted Degree of a Node*

For a weighted graph, let

$$w(v_i) = \text{weight of node } v_i$$
$$w(e_i) = \text{weight of edge } e_i$$
$$w(V_P) \triangleq \sum_{v_i \in V_P} w(v_i) \qquad (1)$$

The weighted degree of a node $v_i \in V$ or a node set $V_P \in V$ is defined as

$$D(v_i) \triangleq \sum_{\substack{v_j \in V \\ v_i \neq v_j}} w(e(v_i,v_j)) \qquad (2)$$

$$D(V_P) \triangleq \sum_{v_i \in V_P} D(v_i) \qquad (3)$$

In the above $e(v_i,v_j)$ denotes the edge connecting node $v_i$ and node $v_j$.

*Definition 4. Connection Cost*

The connection cost $f(V_P,V_Q)$ from a node set $V_P$ to a node set $V_Q$, is the sum of weights of edges connecting the nodes in $V_P$ to those in $V_Q$, that is,

$$f(V_P,V_Q) \triangleq \sum_{\substack{v_i \in V_P \\ v_j \in V_Q}} w(e(v_i,v_j)) \qquad (4)$$

In particular, $f_i(V_P)$ defined by

$$f_i(V_P) \triangleq f(V_P,V_P) \qquad (5)$$

is twice the sum of weights of edges in $G_s$ $(V_P)$, and may be called the internal connection cost of $G_s(V_P)$. Also, $f_e(V_P)$ defined by

$$f_e(V_P) \triangleq f(V_P, V\text{-}V_P)$$

where V is the node set of given a weighted graph, is the sum of weights of the cutset edges incident at a section graph $G_s(V_P)$, and may be called the external connection cost of $G_s(V_P)$.

*Lemma 1.*

For an undigraph,

$$f(V_P,V_Q) = f(V_Q,V_P). \qquad (7)$$

*Lemma 2.*

Let $V_P$ be the union of disjoint node sets $V_1$, $V_2$, ..., $V_m$. Then,

$$f(V_P,V_Q) = \sum_{K=1}^{m} f(V_k,V_Q) \qquad (8)$$

*Definition 5. Contracted Graph*

A contracted graph (C-graph) is a graph resulting from callapsing each CSG to a (super) node with thinning operation in an iterative partitioning stage, with the connection cost and the total node weight associated with each CSG being preserved.

*Lemma 3.*

The following operations are equivalent and will be called optimal partitioning.

(i) Partitioning a given weighted graph with a total node weight $w(V)$ to N partitions $p_1,p_2$, ..., $p_N$, such that $w(V_k) \leqslant w(V)/N$ (where $V_k$ is the node set of $p_k$, k = 1,2, ..., N) and the total connection cost among the N partitions is minimum.

(ii) Converting the same graph to a C-graph with N (super) nodes $V'_1$, $V'_2$, ..., $V'_N$ such that $w(V'_k) \leqslant w(V)/N$ (k=1,2, ..., N), and the total weighted degree of the N (super) nodes is minimum.

Lemma 3(ii) suggests that optimal partitioning may be obtained iteratively by finding a sequence of CSG's, each (super) node of the resulting C-graph satisfying the conditions stated above.

*Theorem 1.*

For an undigraph G(V,E),

$$D(V_P) = f_e(V_P) + f_i(V_P) \qquad (9)$$

where $V_P \subseteq V$.

*Proof:* The theorem can easily be proved by mathematical induction. Let $N(V_P)$ be the cardinality of a node set $V_P$. For $N(V_P) = 1$, (9) is true by the definitions of respective terms. Now, assume that (9) is true for $N(V_P) = n$. Then, for $V_Q = V_P \cup v_i (v_i \not\in V_P)$, i.e. for $N(V_Q) = n+1$, (9) also holds true since

$$\begin{aligned} D(V_Q) &= D(V_P) + D(v_i) \\ &= f(V_P, V-V_P) + f(V_P,V_P) + f(v_i,V_P) \\ &= + f(v_i,V-V_P-v_i) \\ &= f(V_P, V-V_P-v_i) + f(V_P,v_i) + f(V_P,V_P) \\ &\quad + f(v_i,V_P) + f(v_i,v_i) = f(v_i, V-V_P-v_i) \\ &= [f(V_P,V-V_P-v_i) + f(v_i,V-V_P-v_i)] \\ &\quad + [f(V_P,v_i) + f(V_P,V_P)] + [f(v_i,V_P) \\ &\quad + f(v_i,v_i)] \\ &= f(V_Q, V-V_Q) + f(V_P,V_Q) + f(v_i,V_Q) \\ &= f_e(V_Q) + f_i(V_Q) \end{aligned}$$

Therefore, (9) is true for any $V_P \subseteq V$.

*Corollary*

In partitioning a given node set V into N partitions such that $V = \{V_1, V_2, ..., V_n\}$ and $w(V_k) \leq w_{max}$ for $k = 1,2, ..., N$, the following two expressions are equivalent:

(i) minimize $\sum_{K=1}^{N} f_e(V_k)$

(ii) maximize $\sum_{K=1}^{N} f_i(V_k)$

*Proof*: $$f(V) = \sum_{K=1}^{N} (D(V_k) f_i(V_k)) = D(V) - \sum_{k=1}^{N} f_i(V_k)$$

where D(V) is constant for a given graph. Therefore, the statement to be proved is true.

*Difinition 6. Connection Index*

The connection index $C(V_P)$ of a section graph $G_s(V_P)$ associated with a node set $V_P$ is defined as the ratio of the internal connection cost of $V_P$ to the external connection cost of $V_P$, that is,

$$C(V_P) \triangleq f_i(V_P)/f_e(V_P) \qquad (10)$$

As will become clear in the next section, the introduction of the concept of connection index is motivated by an effort, in network partitioning, to find a section graph for which the total weight of internal edges is as large as possible, within constraint for the total internal node weight, and, at the same time, the total weight of outgoing edges is as small as possible.

*Theorem 2.*

For any node $v_i$ in a graph G(V,E), there is a node set $V_P$ such that $v_i \in V_P$ and $C(V_P)$ is maximum or infinite.

*Proof:* If $V_P = \{v_i\}$, then $C(V_P) = 0$ and when $V_P = V$ then $C(V_P)$ is infinite. From this fact the theorem follows.

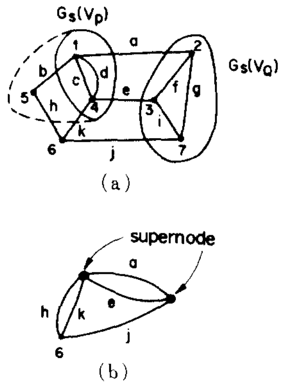


Fig. 1. A graph for illustration of definitions.

To illustrate some definitions given thus far we consider Fig. 1. Initially the node subset $V_P$ consists of nodes 1 and 4 and the associated section graph $G_s(V_P)$ consists of nodes 1 and 4 and edges c and d. The weighted degree of node $v_1$, $D(v_1)$, is $w_a + w_b + w_c + w_d$ and that of node set $V_P$, $D(V_P)$, is $w_a + w_b + 2(w_c+w_d) + w_e + w_k$. The total node weight of $G_s(V_P)$ is $w_1 + w_4$ and the connection cost $f(V_P,V_Q) = w_a + w_e$. The internal connection cost $f_i(V_P)$ is $2 \cdot (w_c+w_d)$ and the external connection cost $f_e(V_P)$ is $w_a+w_e+w_k+w_b$. The connection index $C(V_P)$ is the ratio of these two, and hence $2 \cdot 2/4$ if all edges are equally weighted. It is easily seen that (9) holds true for $V_P$. If $G_s(V_P)$ does not part during a partitioning iteration stage, it may be regarded as a CSG. $G_s(V_P)$ may grow during a partitioning stage to include node 5 and hence edge b and then it may be collapsed to a supernode. If $G_s(V_Q)$ is likewise collapsed to a supernode and no further operation is performed in this iteration stage, we will end up with a C-graph as shown in Fig. 1(b), where two supernodes have weights $w_1+w_4+w_5$ and $w_2+w_3+w_7$, respectively.

## III. Algorithm

Our objective is to partition a given weighted graph into a given number N of CSG's such that the sum of weights of the nodes in each CSG does not exceed a given number $w_{max}$ and the number of torns $N_t$, i.e., the sum of weights of the edges interconnecting the CSG's, is minimized.

Essentially, the strategy of the new algorithm can be stated as follows:

Step 1. Selecting a seed (initial CSG, the node set of which is denoted as $V_K$), find one of its adjacent nodes $v_a$ that gives the largest connection index together with the original CSG; if the new index is greater than the previous one, that is, if

$$C(V_K + v_a) > C(V_K) \tag{11}$$

then include that node to the original CSG. Grow the CSG in this way until either the connection index begins to decrease or the node weight of the CSG exceeds $w_{max}$. Collapse the resulting CSG to one supernode.

Step. 2 Repeat Step 1, selecting another seed from the remaining nodes, until every node is included in some CSG and thus collapsed to some supernode to obtain a C-graph.

Step. 3 If the number of supernode in the resulting C-graph is greater than N, repeat Steps 1 and 2 with the C-graph successively; otherwise, stop.

Although the above approach seems to be reasonable conceptually, it was found that criterion (11), being a local searching strategy, has a weak point such as the seed effect. Hence, instead of (11), we propose the following criterion for $v_a$, the adjacent node which gives the largest $C(V_K+v_a)$, to be included in the current CSG:

$$C'(V_K + v_a) > C(V_K) + \delta \tag{12}$$

$$\text{where } C'(V_K + v_a) \triangleq \frac{f_i(V_K) + F(V_K, v_a)}{f_e(V_K + v_a)} \tag{13}$$

$$\delta = \begin{cases} \dfrac{1}{2f_e(V_K)}, & \text{for } 0 < C(V_K) < 1 \\ 0 \quad , & \text{otherwise.} \end{cases} \tag{14}$$

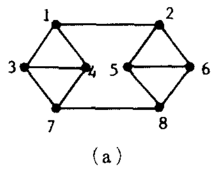
Note that C $(V_K+v_a)$ of (13) becomes $C(V_K+v_a)$ if $f(V_K,v_a)$ in the numerator is multiplied by 2. Hence, criterion (12) is more strict than (11).

Even with criterion (12), a CSG with only two nodes may occur. If this bad sitution happens during the course of finding a C-graph, we neglect such a CSG and treat the related two nodes in the same way as the remaining nodes to include them in the subsequent CSG's but they will not be selected as a seed in the current iteration stage, unless there remain no other nodes to be explored in that particular iteration stage of finding a C-graph.

## IV. Examples

To illustrate the algorithm described in the above we give two examples. In the following we assume that all nodes are equally weighted and so are all edges.

*Example* 1. For the graph of Fig. 2(a) we assume that N=2, $w_{max}$=5 and $v_1$ is arbitrarily selected as the seed (initial CSG-1). Thus, $V_1 = v_1$ initially. Hence, $C(V_1) = 0/3$ and $\delta = 0$. Now, $V_1$ has three adjacent nodes, namely, $v_2$, $v_3$ and $v_4$. $C(V_1+va) = 2\cdot 1/4$ for $v_a$ = any of $v_2$, $v_3$ and $v_4$. Therefore, we arbitrarily pick up $v_2$ (when the scores are even, we pick up the node in the order of data stored) and calculate $C'(V_1 + v_2) = (0 + 1)/4$ according to (13). Therefore, the test criterion (12) is satisfied. All this is indicated in the first row of Table 1. ("O" or "X" in the colum "Test" indicates whether the picked-up adjacent node $v_a$ meets criterion (12) or not, respectively.) CSG-1 now includes $v_1$ and $v_2$ and $V_1 = v_1 \cup v_2$. Thus we obtain the second row of Table 1, which shows that the candicate node $v_3$ does not meet the criterion and hence we do not try to grow CSG-1 any further. Furthermore, since CSG-1 contains only two nodes (a bad situation), we actually discard it and start all over again, with a new seed $v_3$ (this choice is again arbitrary but nodes 1 and 2 can not be a seed candidate any more). The CSG starting with $v_3$ is grown to include $v_4$, $v_7$ and $v_1$ sequentially but not $v_8$, as indicated in rows 3 through 6 in Table 1. We collapse this CSG to a supernode. Then, from the remaining nodes we arbitrarily pick up $v_5$ as the next seed and grow the new CSG to include all the remaining nodes. Thus we end up with a C-graph with two supernodes (implying two partitions), as shown in Fig. 2(b) each satisfying the condition for the maximum node weight.



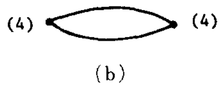


**Fig. 2.** (a) Graph of Example 1.
(b) The C-graph; the number in ( ) indicates the weight of the supernode.

In other words, in one iteration we have obtained the desired partitioning with the number of torn $N_t = 2$, which is optimum in this problem. This optimal result can not be obtained in the contour approach[11] as applied to branch tearing, if we start with node $v_1$—which minifests the seed effect of the contour approach.

**Table 1.**

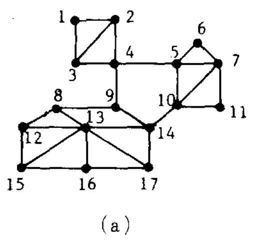
| $V_K$ | $C(V_K)$ | $\delta$ | $v_a$ | $C(V_K+v_a)$ | Test |
|---|---|---|---|---|---|
| [1] | 0 | 0 | 2 | 1/4 | O |
| [1,2] | 2/4 | 1/8 | 3 | 3/5 | X |
| [3] | 0 | 0 | 4 | 1/4 | O |
| [3,4] | 2/4 | 1/8 | 7 | 4/3 | O |
| [3,4,7] | 6/3 | 0 | 1 | 8/2 | O |
| [3,4,7,1] | 10/2 | 0 | 8 | 11/3 | X |
| [5] | 0 | 0 | 2 | 1/4 | O |
| [5,2] | 2/4 | 1/8 | 6 | 4/3 | O |
| [5,2,6] | 6/3 | 0 | 8 | 8/2 | O |
| [5,2,6,8] | 10/2 | 0 | none | | |
| CSG-1 | [1,2], neglected | | | | |
| CSG-1 | [3,4,7,1] | | | | |
| CSG-2 | [5,2,6,8] | | | | |

*Example 2.* For the graph of Fig. 3(a) we assume that N=2, $w_{max}=9$ and $v_1$ is selected as the seed. The first C-graph obtained by

**Table 2.**

| $V_K$ | $C(V_k)$ | $\delta$ | $v_a$ | $C'(V_K+v_a)$ | Test |
|---|---|---|---|---|---|
| [1] | 0 | 0 | 2 | 1/4 | O |
| [1,2] | 2/3 | 1/6 | 3 | 4/2 | O |
| [1,2,3] | 6/2 | 0 | 4 | 8/2 | O |
| [1,2,3,4] | 10/2 | 0 | 6 | 11/3 | X |
| [5] | 0 | 0 | 6 | 1/4 | O |
| [5,6] | 2/4 | 1/8 | 7 | 4/4 | O |
| [5,6,7] | 6/4 | 0 | 10 | 8/4 | O |
| [ ..., 10] | 10/4 | 0 | 11 | 12/2 | O |
| [ ..., 11] | 14/2 | 0 | 14 | 15/4 | X |
| [8] | 0 | 0 | 9 | 1/4 | O |
| [8,9] | 2/4 | 1/8 | 12 | 3/5 | X |
| [12] | 0 | 0 | 8 | 1/4 | O |
| [12,8] | 2/4 | 1/8 | 13 | 4/6 | O |
| [12,8,13] | 6/6 | 0 | 15 | 8/5 | O |
| [ ..., 15] | 10/5 | 0 | 16 | 12/4 | O |
| [ ..., 16] | 14/4 | 0 | 17 | 16/3 | O |
| [ ..., 17] | 18/3 | 0 | 14 | 20/3 | O |
| [ ..., 14] | 22/3 | 0 | 9 | 24/2 | O |
| [ ..., 9] | 26/2 | 0 | none | | |
| CSG-1 | [1,2,3,4] | | | | |
| CSG-2 | [5,6,7,10,11] | | | | |
| CSG-3 | [8,9], neglected | | | | |
| CSG-4 | [12,8,13,15,16,17,14,9] | | | | |

following the procedure indicated in Table 2 is shown in Fig. 3(b), which is, in turn, further reduced to the second (and final) C-graph as shown in Fig. 3(c), which is the optimal result with $N_t = 2$.



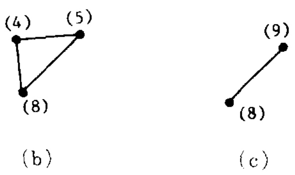


**Fig. 3.** (a) Graph of example 2.
(b) First C-graph of (a).
(c) Final C-graph. The number in ( ) in (b) and (c) indicates the weight of the supernode.

## V. Computational Complexity

The most time-consuming process in the proposed algorithm is the step of growing a CSG. Therefore, the computational complexity may be considered to be proportional to the total number of steps taken in growing all CSG's in the whole iteration stages, namely, proportional to the number of rows in Table 1 or 2. Suppose at the beginning of the k-th iteration stage we have $v^{(k)}$ nodes and $e^{(k)}$ edges. In the worst case each node is chosen as a seed (once at most) and the adjacent nodes to be explored for growing the CSG is of the order of $e^{(k)}$. Thus, when the iteration stops at the m-th C-graph (m is usually small, perhaps less than 5, in most of practical problems), the order of complexity is $0(\sum_{i=1}^{m} v^{(i)}.e^{(i)})$ or $O(v.e)$, where $v=v^{(1)}$ and $e=e^{(1)}$ are the number of nodes and edges, respectively, in the original associated graph, and $v^{(1)} > v^{(2)} > v^{(3)} > \dots, e^{(1)} > e^{(2)} > e^{(3)} \dots$ .

## VI. Experimental Results

The proposed algorithm has been implemented on the HP3000 computer. The details of the program is given in Appendix. It was tested for a number of graphs. Table 3 shows the results for those graphs given in Fig. 12 of [1] and reproduced in Fig. 4 below. In Table 3, v is the number of nodes, e is the number of edges, $N_{t,e}$ is the experimental $N_t$, $N_{t,o}$ is the optimal $N_t$ and Time is CPU time in seconds on HP3000. As can be seen, in most cases the algorithm gives optimal partitioning,

**Table 3.** Summary of experimental results for the graphs of Fig. 5.

| Fig. | v | e | N | $w_{max}$ | $N_{t,e}$ | $N_{t,o}$ | Time |
|---|---|---|---|---|---|---|---|
| Fig.5(a) | 46 | 69 | 3 | 18 | 6 | 5 | 0.086 |
| Fig.5(b) | 94 | 176 | 4 | 27 | 4 | 4 | 0.377 |
| Fig.5(c) | 51 | 126 | 4 | 15 | 18 | 18 | 0.132 |
| Fig.5(d) | 14 | 25 | 2 | 8 | 6 | 6 | 0.026 |
| Fig.5(e) | 37 | 68 | 4 | 12 | 8 | 8 | 0.052 |
| Fig.5(f) | 50 | 100 | 3 | 20 | 14 | 14 | 0.144 |
| Fig.5(g) | 77 | 180 | 3 | 30 | 22 | 20 | 0.358 |

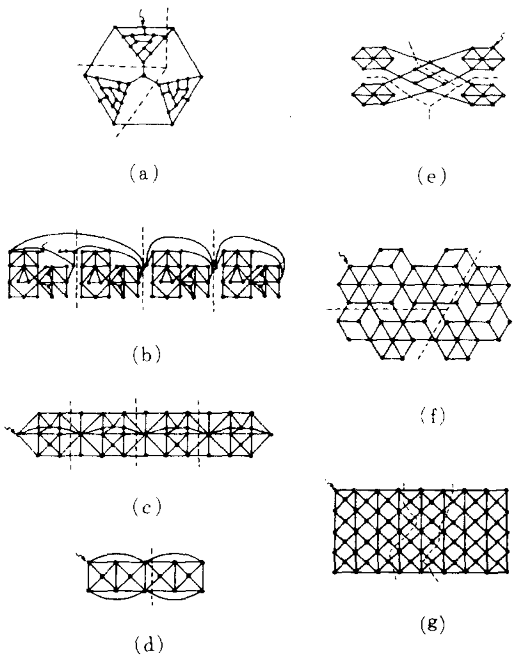


**Fig. 4.** Graphs for test of the proposed algorithm. The arrow indicates the seed.

while in other cases it gives near optimal partitioning with only one or two additional torns as compared with the optimal partitioning. Note that the last statement is true in general even for those graphs without heavily clustered components (that is, for those graphs in which most of the circuit lengths are equal and the minimum circuit length is relatively large) such as Fig. 4a. Incidentally, in the above experiments all nodes and all edges are again equally weighted for simplicity.

Fig. 5 plots the computation time versus the product of ve from Table 3. It is seen that 0(ve) is an upper bound for the computational complexity since all the data points are bounded by a straight line. It is interesting to note that actually the processing time is 0(v), as can be seen from Fig. 6(b), where not only the data of Table 3, but also the computation times for the test pattern of Fig. 6(a) are plotted.

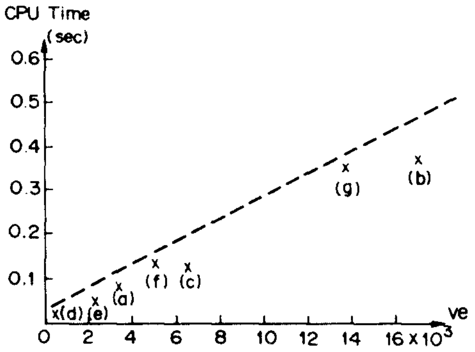
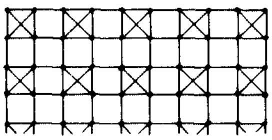
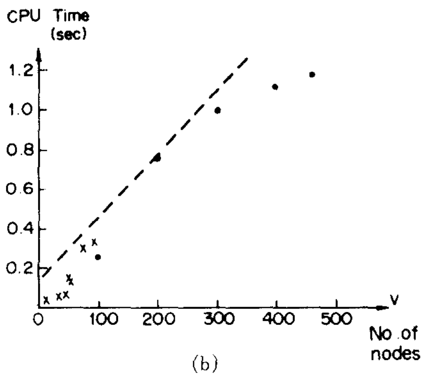


Fig. 5. Processing time vs. the product vs for the graphs of Fig. 4.



(a)



(b)

Fig. 6. (a) Test pattern. (b) Processing time vs. v(total number of nodes) for the graphs of Fig. 4 and Fig. 6(a).

## VII. Applications to Network Problems

The algorithm presented in this paper can be applied to many practical problems in network analysis and design, among which we will consider the VLSI layout and the diakoptic analysis problems. We will give only some suggestions without going into detailed development.

In the VLSI design, in particular, in the gate array approach the current trend is to use cells as the basic circuit blocks and to find an efficient way of placing and interconnecting the cells compatible to the circuit configuration. Therefore, in the VLSI layout design, partitioning plays an important role, in which the objective is usually to minimize the sum of the weights of torn edges. In order to apply the partitioning algorithm to this layout problem, we have to first construct the associated graph, which will be done in the following way. Each circuit element or cell is converted to a node whose weight represents the geometric size of the corresponding element or cell. On the other hand, the edges are weighted equally as unity in general; a larger weight may be assigned, however, to those edges which are desired to be short in length for low ohmic voltage drop or for low electric/magnetic coupling. Sometimes edges with a negative weight may be added between those elements which are not desired to be in the same cluster. The partitioning algorithm may now be applied to the associated graph constructed in this way.

Next, consider the diakoptic analysis problem. When the modified nodal approach[26] is employed for circuit analysis, node tearing is preferable since branch tearing increases the number of variables in general. The idea and algorithm presented in this paper may still be applied to the case of node tearing except that connection cost (Definition 5) should be interpreted now as the number of edges whose terminal nodes are not the same, in any expression involving this term (in particular, in the test condition (12) and (13) ). (In actual programming, an edge table, instead of a node table, may be needed to store the terminal nodes.) The matrix of diakoptic analysis is of the bordered block diagonal form and the number of torn nodes and edges ,is equal to the width of the border. With the same partition size and number of partitions, the thinner the border is, the less the computation time is required. One important restriction is that there must not be coupling between different partitions except the coupling at torns. Thus, for example, the controlling and controlled branch pair should not be torn, but treated as a supernode from the biginning. Transistors belong to this category. Such a supernode should be weighted by the number of nodes within it that are not represented in the associated graph.

## VIII. Conclusion

Considering the concept of the newly introduced connection index, the ratio of twice the sum of weights of edges in a CSG to the sum of weights of edges incident at the CSG, it is reasonable to grow a CSG, starting with a seed, in such a way that the new CSG with one adjacent node added has the largest possible connection index. This local searching technique, as found to have some weak points, was modified for global searching and to cover broader graph configurations. The modified criterion (12) can not be unique by nature, but it worked well for most of partitioning problems, when used together with the special treatment of the CSG with only two nodes as mentioned previously.

The proposed algorithm has the following features:

1. Weighted graph partition algorithm.
2. Effective due to the greedy strategy employed.
3. No seed effect in most cases.
4. Interactive procedure easily implementable.
5. It yields an optimal solution in most cases (only a few additional torns in near optimal cases).

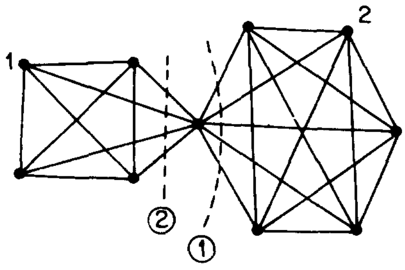


Fig. 7. An articulation point of two cliques to illustrate the seed effect.

Nevertheless, it was found that the proposed algorithm encounters some difficulty for graphs of uniform structure such as honeycomb-like one, which necessiates further investigation. It was also found that the seed effect still exists for an articulation point of two cliques such as Fig. 7, since, given N=2 and $w_{max}$=6, we end up with the partition as indicated by the dotted line ①(②) if node 1 ② is selected as the seed. Applications of the algorithm to VLSI layout design and diakoptic analysis have been suggested and will be published elsewhere in full context.

## Appendix

### *Program Partition*

1. Read input data and construct the associated graph (it becomes the initial C-graph).
2. Set up the array D(k) of the weighted degree for each (super) node of the C-graph. k=1~NN; NN = the number of (super)nodes.
3. Initialize the node status NSTS (k).

$$\text{NSTS}(k) = \begin{cases} \text{0, initially} \\ \text{a positive integer, which indicates the number of the CSG to which node k belongs.} \end{cases}$$

```
4. For k=1 to NN do
     if NSTS(k) = 0 then call CSGFND(k)
   end
5. Make a C-graph and update NN and D(k).
```

```
6. If NN $\leqslant$ N (the given number of partitions)
   then goto 7 else goto 2.
7. Print out the results.
8. Call exist
9. end partition
```

Procedure CSGFND (SEED)

```
1. Initialize and set up the table of adjacent
   nodes, ANS.
2. Find a node from ANS which gives the
   largest connection index.
3. If criterion (12) is satisfied then update
   ANS, C, and DELTA: goto 2
4. Check the number of nodes in the current
   CSG.
   If it is two and there remain other nodes to
   be explored, then delete the current CSG;
   goto 2
5. Save the current CSG and update the status
   of its nodes
   end CSGFND
```

**References**

[1] A. S. Vincentelli, L. K. Chen, and L. O. Chua, "An efficient heuristic cluster algorithm for tearing large scale networks," *IEEE Trans. Circuits and Systems,* vol. CAS-24, pp. 709-717, Dec., 1977.

[2] L. O. Chua and L. K. Chen, "Diakoptics and generalized hybrid analysis, *IEEE Trans. Circuits and Systems,* vol. CAS-23, pp. 694-705, Dec., 1976.

[3] F. Luccio and M. Sami, "On the decomposition of networks in minimally interconnected subnetworks,"*IEEE Trans. Circuit Theory,* vol. CT-16, pp. 184-188, May, 1969.

[4] B. W. Kernighan and S. Lin, "An efficient heuristic procedure for partitioning graphs," *Bell System Tech. J.,* vol. 49, no. 2, pp. 291-307, Feb., 1970.

[5] A. E. Engle and D. A. Mlynski, "Cliques and partition of graphs," *IEEE Proc.* ISCAS/78, pp. 81-82, 1978.

[6] W. Tang, A. N. Michel, and H. W. Hale, "On structure and stability of interconnected dynamical systems," *IEEE Trans. Circuits and Systems,* vol. CAS-27, no. 5, pp. 391-404, May, 1980.

[7] P. Rosenstiehl, J. R. Fiksel, and A. Holliger, "Network of finite automata capable of solving graph problem."

[8] R. P. Tewarson, *Sparse Matrices,* Academic Press, 1973.

[9] G. Guardabassi and A. S. Vincentelli, "A two levels algorithm for tearing," *IEEE Trans. Circuits and Systems,* vol. CAS-23, pp. 783-791, Dec. 1976.

[10] G. S. Dantzig and P. Wolfe, "The decomposition algorithm for linear programming," *Econometrica,* vol. 29, no. 4, pp. 141-150, Jan., 1970.

[11] N. B. Rabbat and H. Y. Hsieh, "A latent micromodular approach to large scale sparse networks," *IEEE Trans. Circuits and Systems,* vol. CAS-23, no. 12, pp. 745-751, Dec., 1976.

[12] E.C. Ogbuobiri, W. F. Tinney, and J. W. Walker, "Sparsity directed decomposition for gaussian elimination on matrices," *IEEE Trans. Power, Appr. Syst.,* vol. PAS-89, pp. 141-150, Jan., 1970.

[13] I. N. Hajj, "Sparesity consideration in network solution by tearing," *IEEE Trans. Circuits and Systems,* vol. CAS-27, no. 5, pp. 357-366, May, 1980.

[14] F. F. Wu, "Solution of large scale networks by tearing," *IEEE Trans. Circuits and Systems,* vol. CAS-23, no. 12, pp. 705-713, Dec., 1976.

[15] C. W. Ho, A. E. Ruehli, and P. E.Brennan, "Modified nodal approach to network analysis," *IEEE Trans. Circuits and Systems,* vol. CAS-22, no. 6, pp. 504-509, Jun., 1975.

[16] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, "The design and analysis of computer algorithms," Reading, Mass.: Addison-Wesley, 1974.

[17] R. M. Karp, "Reducibility among combinatorial problems, in complexity of computer computation," pp. 85-703, New York: Plenum Press, 1972.

[18] G. Persky, D. N. Deutch, and D. G. Schweikert, "LTX a minicomputer based system for automated LSI layout," *J. of Minicomputer Based System,* vol. 1, no. 3, pp. 217-255, May, 1977.

[19] D. Ferrari, "Improving locality by critical working sets," *Comm. of ACM*, vol. 17, no. 11, pp. 614-620, Nov., 1974.
[20] J. C. Gower, "Comparison of some methods of cluster analysis," *Biometrices*, vol. 54, pp. 623-637, Dec., 1967.
[21] B. W. Kernighan, "Some graph partitioning problems related to program segmentation," Ph.D. These, Princeton Univ., pp. 74-726, Jan. 1969.
[22] L. R. Ford and D. R. Fulkerson, "Flows in Networks," Princeton, N.J.: Princeton Univ. Press, 1962.
[23] S. Lin, "Heuristic programing as an aid to network design," *Biometrics*, vol. 54, pp. 623-637, Dec., 1967.
[24] G. Kron, "A set of principles to interconnect the solutions of physical systems, *J. of Appl. Physics*, vol. 24, no. 8, pp. 965-980, Aug., 1959.
[25] P. M. Lin, "A survey of application of symbolic network functions," *IEEE Trans. Circuit Theory*, pp. 723-737, Nov., 1973.