Indirective Random Optimal Component Placement

by

Yeun Tsum Wong, Pecht, Palmer

Indirective Random Optimal Component Placement [1]

Mechanical Engineering Department

College Park, University of Maryland

Yeun Tsun Wong

Dr. Michael Pecht

Dr. Milton Palmer

Abstract

Presented is a new efficient method for the placement of components on a printed wiring board (PWB). In order to obtain shortest total wire length, all nets are considered as randomly connected minimum Steiner trees. A random optimal net process theory is used to eliminate redundant comparison and an indirective optimal technique is used to avoid calculating the exact length of minimum Steiner trees in the iterative process.

Introduction

The increased demand for high density printed wiring boards (PWB)s has necessitated the utilization of advanced computer-aided design (CAD) techniques. This is particularly evident in the design area of component placement and routing where the design time without CAD would be on the order of years; it is presently on the order of days. Furthermore, the introduction of CAD and the subsequent "tuning" of PWB placement and routing algorithms and heuristics has improved the manufacturability, reliability, testability and maintainability of PWBs.

---

[1] The research was partially supported by *the Westinghouse Defense and Electronics Systems Center at Baltimore, MD* and *the Systems Research Center of University of Maryland, College Park, MD* under the NSF grant No. CDR8500108.

The PWB design process consists of three fundamental steps: initial assignment, placement and routing. Initial assignment is predominately a data management problem consisting of entering and utilizing data such as board and component dimensions, nets, and the components and pins in every net.

Placement consists of determining the optimal locations of components on the board with respect to a measure typically defined with respect to the interconnections. In this discussion, we choose to minimize the total weighted wire length of the PWB components, where all the components are assumed to be same size and rectangular. To simplify the process, all pins of a component are assumed to be located at the geometric center of the component. This paper will focus on the placement problem.

Routing is the final design task. It is the selection of the actual paths of the nets interconnecting the components. Routing is a costly process, which typically requires excessive computer time and numerous storage calls. The routing cost is primarily dependent on PWB complexity and component placement. An optimal placement is therefore critical to successful cost-effective routing.

In this paper, a new efficient placement method based on an iterative improvement method will be discussed. Historically, graph partitioning has been employed in iterative improvement methods[1 - 8]. The graph partitioning problem for a circuit can be expressed as: given n elements connected by nets that connect two or more elements, partition the elements into two subsets of size n/2 such that the sum of the weights on the nets connecting the two subsets is minimum. A net is said to be cut if it connects elements in one subset to elements in the other. The weighted function of a partition is the number of nets that are cut. It is always the local measure of a sub-process. So the final result is an union of the local optimal results of a series of sub-processes. Kernighan-Lin's algorithm can be employed to escape from local minima so as to obtain a minimum cut in the iterative process.

In graph partitioning, the effect of the element sizes is not considered. In VLSI, this may not be important because the number of pins per element is small for most circuits. In contrast, the number of pins per component in a PWB is much higher (often more than 64 pins on DIP components). It is thus necessary to consider the size effect in the placement process. If all the nets are processed as minimum Steiner trees in the iterative processing, the size effect can be considered and the process will be consistent with the routing processes[13,14,15] which is similar to the Lee's algorithm[10,12,13]. Unfortunately, the execution time which is required to build Steiner trees for every comparison cannot be tolerated[13]. This in turn limits the capability to incorporate other constraints, such as thermal and mechanical, within the placement process.

In order to obtain the shortest total wire length and avoid building random minimum Steiner trees in the placement processing, it is shown that indirective random optimal nets can be constructed to replace the random minimum Steiner trees. Components being processed may be partitioned only once and divided into two subsets dependent on the weighted function, the total wire length. Using this method, an optimal result can be obtained. In addition, because it is not necessary to collect edges in a net into clusters as it is done in the "min-cut" method, wire congestion can be decreased.

Indirective Random Optimal Net Theory

In an indirective random optimal net process, a "nominal" structure of a Steiner tree must be established. We call this representative structure a nominal net or a nominal tree of its equivalent Steiner tree. This structure is designed to be easily built or re-built and its main data is easily computed. To determine structure types which can function as a nominal net, the following theory is established.

A net can be expressed by a variety of trees. The length of a minimum Steiner tree is a minimum for all trees[9]. We defined a graph G which consists of n minimum Steiner trees as

$$G = \bigcup_{i=1}^{n} N_i ,$$

where the i th minimum Steiner tree N_i is given by

$$N_i = \bigcup_{j=1}^{m} C_j , \quad i = 1, 2, \ldots\ldots, n$$

and where C_j is j th component in the i th tree. We further define the nominal graph of G as NG and the nominal tree of N_i as NN_i, such that

$$NG = \bigcup_{i=1}^{n} NN_i$$

where NN_i include all components in N_i.

If the length of a graph (or sub-graph) x is denoted as L_x, then

$$L_G = \sum_{i=1}^{n} L_{N_i}$$

and

$$L_{NG} = \sum_{i=1}^{n} L_{NN_i} .$$

If minimum nominal nets are their equivalent minimum Steiner trees,

$$L_{N_i} \leq L_{NN_i} \quad \text{for } i = 1, 2, \ldots\ldots, n$$

because a minimum Steiner trees is always an optimal structure with shortest wire length.

Therefore

$$L_G \leq L_{NG} .$$

When L_{NG} is convergent to a minimum,

$$L_G^{\min} \leq L_{NG}^{\min}$$

where $L_G^{\min}$ and $L_{NG}^{\min}$ represent the minimum length of L_G and L_{NG} respectively.

Theory: The Indirective Convergence Theory

If nominal trees NN_i, $i = 1, 2, \ldots\ldots, n$, are equivalent to minimum Steiner trees N_i, $i = 1, 2, \ldots\ldots, n$, and if L_{NG} is a minimum, then L_G is a minimum.

According to the indirective convergence theory, any net which connects all components in a minimum Steiner tree will satisfy this theory if its shortest net length is the length of the minimum Steiner tree. Nominal trees may be constructed in many ways. If NN_i is considered as a closure of N_i, in order to make N_i smallest, we can adjust NN_i rather than N_i.

The Nominal Graph

Based on the indirective convergence theory, the following nominal structure (Fig. 1) will satisfy the engineering requirements in a placement process:

1. Components in the same row are connected one by one from the first terminal component to the last terminal component. If components are rectangular, the shorter edge will be parallel to the row direction because the connection is row oriented.

2. Each row will be connected to its next row with the shortest Manhattan distance.

3. The terminal of a net is a floating node along a given terminal boundary and it is connected with any nearest component. According to the structure of nominal nets, the length of a nominal net with n rows occupied by a net is

$$L_n = \sum_{i=1}^{n} (\text{the coordinate of last terminal component} - \text{the coordinate of first terminal component}) + \sum_{i=1}^{n-1} \text{row distance} + \text{the distance between terminal boundary and its nearest component.}$$

For a nominal net with 20 components and 6 rows, it takes an average of 8 additions and 1 multiplication to compute the nominal net length. Using a Steiner tree, it takes more then 38 additions and more then $\sum_{n=1}^{19} n\,(20-n) = 1330$ comparison to compute the length of the spanning tree which is related to this Steiner tree[13]. Thus, the computing time on a spanning tree related to the Steiner tree with 20 components is almost 1330 times that for a

nominal tree. This does not include the computing time spent on modifying the spanning tree into the Steiner tree.

The Computation of the Random Optimal Net

To determine the optimal location of components on a PWB, a measure for the processing must be defined and the computational rule for the measure must be derived. This is discussed in the following.

Let $G(x, y)$ be a set of n nodes connected with m nets and

$$G = \bigcup_{i=1}^{n} \{ \bigcup_{j=1}^{m} N_j \}.$$

Before discussing the computation of the gain obtained by exchanging two nodes, the following definitions are established.

Definition: Net-node Relation

If $e \in N_j$, then N_j is e relative. If $e \subseteq N_j$, N_j is not e relative.

Definition: Weight of a Net

The weight of a net is the total length of the net. The weight of the jth net is denoted as WN^j. If there is a node c, such that $c \in G$ and N^j is c relative, then the weight of the jth net is denoted as WN_c^j, and as $WN_{\neq c}^j$ if the net is not c relative. The weight of net i is L_{NN_i} here.

Definition: Weight of a Node

The weight of a node is the sum of weights of all nets passing the node. The weight of node c is denoted as W_c. The weight of node c is denoted as $W_{c,d}^k$ if k nets are c and d relative, and denoted as $W_{c,\neq d}^k$ if k nets are c relative, but not d relative. It is noted that $W_{c,d}^k = W_{d,c}^k$.

Definition: Gain of Exchanging Nodes a and b

The gain g_{ab} of exchanging node a and node b is the difference of weights prior to exchanging and after exchanging. It is noted that $g_{ab} = -g_{ba}$.

The Computation of the Gain in the Random Optimal Net Placement

We now let p and q be the number of the activated pins in component a and component b respectively. If

$$G_a = \bigcap_{j=1}^{p} N_j^a \subset G ,$$

$$G_b = \bigcap_{j=1}^{q} N_j^b \subset G ,$$

then

$a \in G_a$ for $(n-p)$ nets are not a relative,

$b \in G_b$ for $(n-q)$ nets are not b relative.

Next, suppose p' nets are both a and b relative. It is noted that

$$p' \leq p , \text{ and } p' \leq q .$$

The weight of node a and node b can be expressed as

$$\begin{aligned} W_a &= \sum_{k=1}^{p} WN_a^k \\ &= \sum_{u=1}^{p'} WN_{a,b}^u + \sum_{v=1}^{p-p'} WN_{a,\neq b}^v \\ &= W_{a,b}^{p'} + W_{a,\neq b}^{p-p'} , \end{aligned}$$

and

$$\begin{aligned} W_b &= \sum_{k=1}^{q} WN_b^k \\ &= \sum_{u=1}^{p'} WN_{b,a}^u + \sum_{v=1}^{q-p'} WN_{b,\neq a}^v \\ &= W_{b,a}^{p'} + W_{b,\neq a}^{q-p'} . \end{aligned}$$

Denoting the position of node n as $S(x_n, y_n)$ in space $S(x, y)$, and swapping a and b, $a \rightarrow b^*$, $b \rightarrow a^*$, gives

$$S(x_{a^*}, y_{a^*}) = S(x_b, y_b),$$

$$S(x_{b^*}, y_{b^*}) = S(x_a, y_a),$$

$$W_{a^*} = W^{p'}_{a^*,b^*} + W^{p-p'}_{a^*,\neq b^*},$$

$$W_{b^*} = W^{p'}_{b^*,a^*} + W^{q-p'}_{b^*,\neq a^*},$$

$$g_{ab} = (W_a + W_b) - (W_{a^*} + W_{b^*}).$$

The random connection of a Steiner tree is only relative to the positions of nodes in the tree, rather than to their identifiers. Exchanging positions of two components inside a net will not change the weight of the net. Furthermore, since

$$a, b \subset N^1_{a,b} \cap N^2_{a,b} \cap \ldots\ldots \cap N^{p'}_{a,b},$$

and

$$W^{p'}_{a,b} = W^{p'}_{b,a} = W^{p'}_{a^*,b^*} = W^{p'}_{b^*,a^*}.$$

then

$$g_{ab} = (W^{p-p'}_{a,\neq b} - W^{p-p'}_{a^*,\neq b^*}) + (W^{q-p'}_{b,\neq a} - W^{q-p'}_{b^*,\neq a^*}),$$

This is the computational formula of the gain in random optimal net placement. From this computational formula, we can determine the computing time of weights in one pass

$$T_{net} = 2\,[(p-p') + (q-p')].$$

From T_{net}, we note that

1. If $p = q$ and $p' \to p$, then $T_{net} \to 0$. In this case, most nets are relative to both of two exchanging nodes.

2. The greater p' is, the less the T_{net} is for a given p and q.

3. If conclusion 1 is extended to any pair of components, every net will connect most components in the board. Therefore, for a given number of activated pins, the longer the nets are, the less the process time. For example, assume there are n components in a board and n activated pins in every component. If every net goes through each component, any

exchange between the components can not improve the interconnections between the components. So the process time should be zero in this special case. Furthermore, since the length of the nets is also a measure of the complexity of a PWB, the more complex the PWB is, the faster the process speed.

The Placement Algorithms

In the iterative improvement placement method, components must be partitioned into subsets. Components may be partitioned into m subsets such that $n \leq m \leq 2$ for a PWB with n components. Next, all subsets are compared each other and exchanges which obtain maximum gain are performed. If the maximum gain is zero, the minimum total length of nets has been obtained and the processing is terminated. For example, assume the components are partitioned into three sets named A, B and C and suppose A and B, B and C, and C and A are compared and the maximum gains are g_{ab} , g_{bc} , g_{ca} respectively. If $g_{ab} > g_{bc} > g_{ca}$ and $g_{ab} > 0$, then the exchanges which lead to g_{ab} are performed in A and B. If $g_{ab} = 0$, then the processing is terminated. A global placement (n subsets) and a two-subset placement are described in the following.

1. The Global Placement Algorithm

Suppose there are n components in a PWB, and each component is denoted as a or b. The algorithm of the global placement is

```
Repeat
    For a=1 to n do
    For b=1 to n do
        If a ≠ b then
            begin
                Compute p , q , p' ;
```

Compute

$g_{ab} = (W^{p-p'}_{a,\neq b} - W^{p-p'}_{a*,\neq b*}) + (W^{q-p'}_{b,\neq a} - W^{q-p'}_{b*,\neq a*})$;

Store the maximum $G^{\max}_{a,b}$;

end;

If $g^{\max}_{ab} \neq 0$ then

swap a and b;

Until $g^{\max}_{ab} = 0$.

2. The Two-subset Placement Algorithm

Utilizing the indirective random optimal net theory and avoiding local minima[1] in the processing, the two-subset placement algorithm is

Partition a PWB into two subsets A and B of size n/2;

Repeat

Remove mask on all $a \in A$ and $b \in B$;

For i=1 to n/2 do

If both a and b are not masked then

begin

Compute W_a and W_b;

Find a_i and b_i that maximizes

$g^{i}_{ab} = (W^{p-p'}_{a,\neq b} - W^{p-p'}_{a*,\neq b*}) + (W^{q-p'}_{b,\neq a} - W^{q-p'}_{b*,\neq a*})$;

Mask a_i and b_i;

end;

Find k that maximizes

$$g_{ab}^{\max} = \sum_{i=1}^{k} G_{ab}^{i};$$

If $g_{ab}^{\max} > 0$ then

exchange $a_1, \ldots, a_k$ with $b_1, \ldots, b_k$

Until $g_{ab}^{\max} = 0$.

The comparison time spent for the do loop is

$$T_c = C_m^2 [(n/2-1) + (n/2-2) + \ldots + 2 + 1] = n(m-1)(n-m)/(2m)$$

$(2 \leq m \leq n/2)$;

$$T_c = [(n-1) + (n-2) + \cdots + 2 + 1] = n(n-1) \quad (m = n).$$

The comparison time is minimum when $m = 2$ or $m = n/2$, or maximum when $m = n$.

The data processing system of the two-subset placement

The data processing system (Fig. 2) is composed of three systems or 10 sub-systems. The data management system consists of a net-component-location data base, net-row-column relation record, component-net relation record and location-component-weight relation record. There is also a knowledge system, and a placement controller system.

In the procedure for the indirective random optimal placement, the data structures are expressed as

1. Net-row-column relation record (nominal net record):

row_array=array[column-index] of column-coordinate;

net_array=array[net-identifier, row-index] of row_array;

In the placement process, column coordinates of nodes in a net are stored in row_array. The first dimension in net_array gives the net-identifier and second dimension gives the row index. Using nt as a pointer pointing to net_array, in Fig. 4a, data in net_array will be

nt[3,1].col[0]=1, nt[3,1].col[1]=1, nt[3,1].col[2]=3;

nt[3,2].col[0]=4; nt[3,2].col[1]=1, nt[3,2].col[2]=5;

nt[3,2].col[0]=5, nt[3,3].col[1]=3.

If a node location on (1,1) moves to (4,2), then data in the record will be

nt[3,1].col[0]=1, nt[3,1].col[2]=3;

nt[3,2].col[0]=4, nt[3,2].col[1]=1, nt[3,2].col[2]=2, nt[3,2].col[3]=5;

nt[3,3].col[0]=5, nt[3,3].col[1]=3.

Note that row coordinate is stored in col[0]. The nominal net length may be computed as following

$$L_{N_3} = \sum_{n=1}^{3} (\text{nt[3,n].col[last]} - \text{nt[3,n].col[1]})$$

$$+ \text{ratio} \times \left(\sum_{n=1}^{2} (\text{nt[3,n+1].col[0]} - \text{nt[3,n].col[0]}) \right.$$

$$+ \text{nt[3,1].col[0]},$$

where "ratio" is the ratio of the length and the width of the grid occupied by the component and "last" is the number of the last component in a row.

2. Component-net record:

Chip_net_relation=array[component-identifier, index] of net-identifier.

In the array, the first dimension gives component identifier and the second gives the item index. For example, if cnr is a pointer pointing to chip_net relation in the procedure and net 1, 2, 4 and 7 pass component 10, thus the data in the array will be

cnr[10,1]=1, cnr[10,2]=2, cnr[10,3]=4, cnr[10,4]=7.

By the use of the array, the number of nets passing two comparing components can be detected.

3. Location-component-weight relation record:

chip_record=record

identifier;

weight;

restricted-status;

masked-status;

end;

chip_array=array[x-coordinate..y-coordinate] of chip_record.

This record is used in the iterative placement process. The component identifiers are stored in identifier items and the weight of components in weight items. Restricted-status and masked-status are used to recognize the status of every component in the placement process. For example, if some components are restricted in some regions, restricted for those components is true. In a pass of the processing, masked is true when disregarding those exchanged components in the further processing. Two dimensions in chip_array express the space of the PWB.

Knowledge which used in the process includes:

1. The nominal net building knowledge is provided to build or re-build a nominal net.
2. The net weight computing knowledge and the component weight computing knowledge are provided to compute the weight of a component so as to make components comparable.
3. The maximum gain knowledge is provided to determine which exchange may obtain maximum gain in a pass.

Suppose there are 12 components named 1 to 12, 4 nets in a PWB and the ratio of length and width of the components is 1.5. Nets in the PWB are expressed in the form {component name; ..... }. thus, for this example,

Net 1 = { 1; 2; 6; 11; 12; 4 };

Net 2 = { 1; 2; 7; 11; 9 };

Net 3 = { 5; 6; 10; 11; 8 };

Net 4 = { 3; 7; 8; 10; 5 }.

Fig. 4 is the process of the two-subset placement system given above. In this figure, (1) is the initial assignment, (2) though (5) are processing cycles and (6) is an optimal result. The experimental process shows that most of the exchanges between components occur in cycles 1 and 2.

Conclusion

The method presented in this paper is being implemented as part of an experimental program of PWB design automation. It is used to process same size components. The basic research approach involves using the indirective total wire length as the optimal weighted function in the iterative process. Because the partitioning is only done once in the two-subset placement and the redundant comparisons are eliminated, the process is efficient and the result is optimal. The method can also be extended to handle different size components. The complete implementation of the described method is still under development.

Reference

[1]. B.W. Kernighan and S. Lin, "An efficient heuristic procedure for partitioning graphs," Bell Syst. Tech. J.,vol.49, No. 2, pp. 291-308, 1970.

[2]. D.C. Schmidt and L.E. Druffel, "An iterative algorithm for placement and assignment of integrated circuits," in Proc. 12th Design Automation Workshop, 1975.

[3]. D.G. Schweikert, "A 2-dimensional placement algorithm for the layout of electrical circuits," in Proc. 14th Design Automation Workshop, 1976.

[4]. L.I. Corrigan, "A placement capability based on partitioning,"in Proc. 16th Design Automation Workshop, 1979.

[5]. B.W. Kernighan and S. Lin, "Method of minimizing the interconnection cost of linked objects," U.S. patent 3617714, Nov. 2, 1971.

[6]. U. Lanther, "A min-cut placement algorithm for general cell assemblies based on graph partitioning," in Proc. 16th Design Automation Workshop, June 1979.

[7]. C.M. Fiduccia and R.M. Matteyses, "A linear-time heuristic for improving network partitions," in Proc. 19th Design Automation Workshop, 1982.

[8]. A.E. Dunlop and B.W. Kernighan, "A procedure for placement of standard-cell VLSI circuit," IEEE trans. Computer-Aided design, vol. CAD-4, No. 1, Jan. 1985.

[9]. E.N. Gilbert and H.O. Pollak, "Steiner minimum tree," SIAM J. Math., vol. 16, 1968.

[10]. M.A. Breuer, "Design automation of digital systems," Prentice-Hall, INC. 1972.

[11]. M. Hanan, "On Steiner's problem with rectilinear distance," SIAM J. Appl Math., vol. 14, No. 2, March, 1966.

[12]. C.Y. Lee, "An algorithm for path connections and its applications," IRE Trans. on Electronic Computers, vol. EC-10, No. 3, 1961.

[13]. Y.T. Wong, "Random Optimal Net Oriented PWB Design Automation," M.S. dissertation, Mechanical Engineering Department, University of Maryland, 1986.

[14]. D. Hightower, "A solution to line-routing problems on the continuous plane," in Proc. 6th Design Automation Workshop, June, 1969.

[15]. D. Hightower, "The interconnection problem: a tutorial," Computer, April, 1974.

Figures

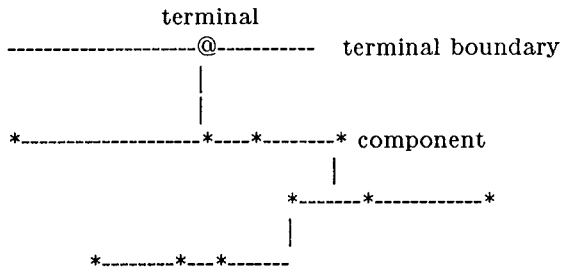


Fig. 1 Nominal tree

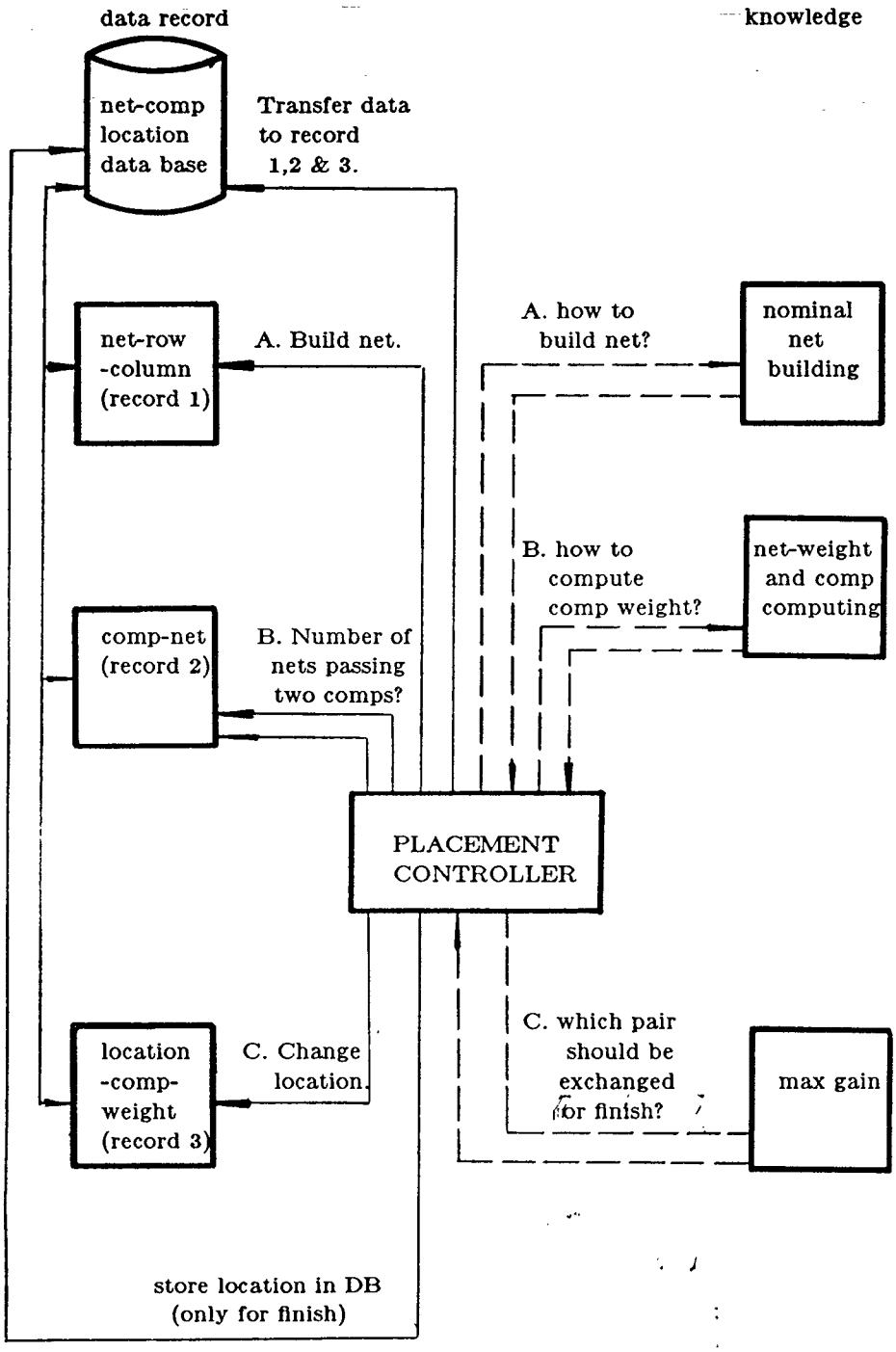
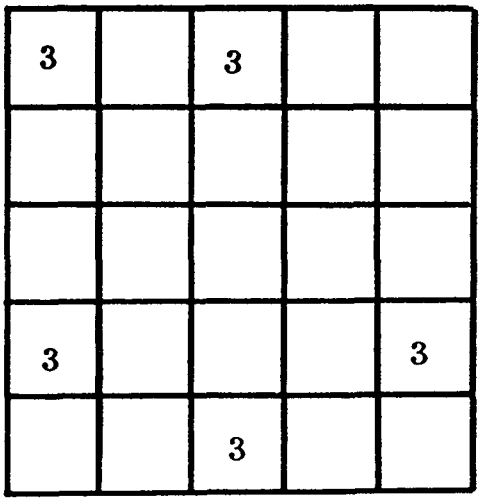


Fig. 2 Placement data processing system



net index: 3

Fig. 3 Pins of a net in a PWB

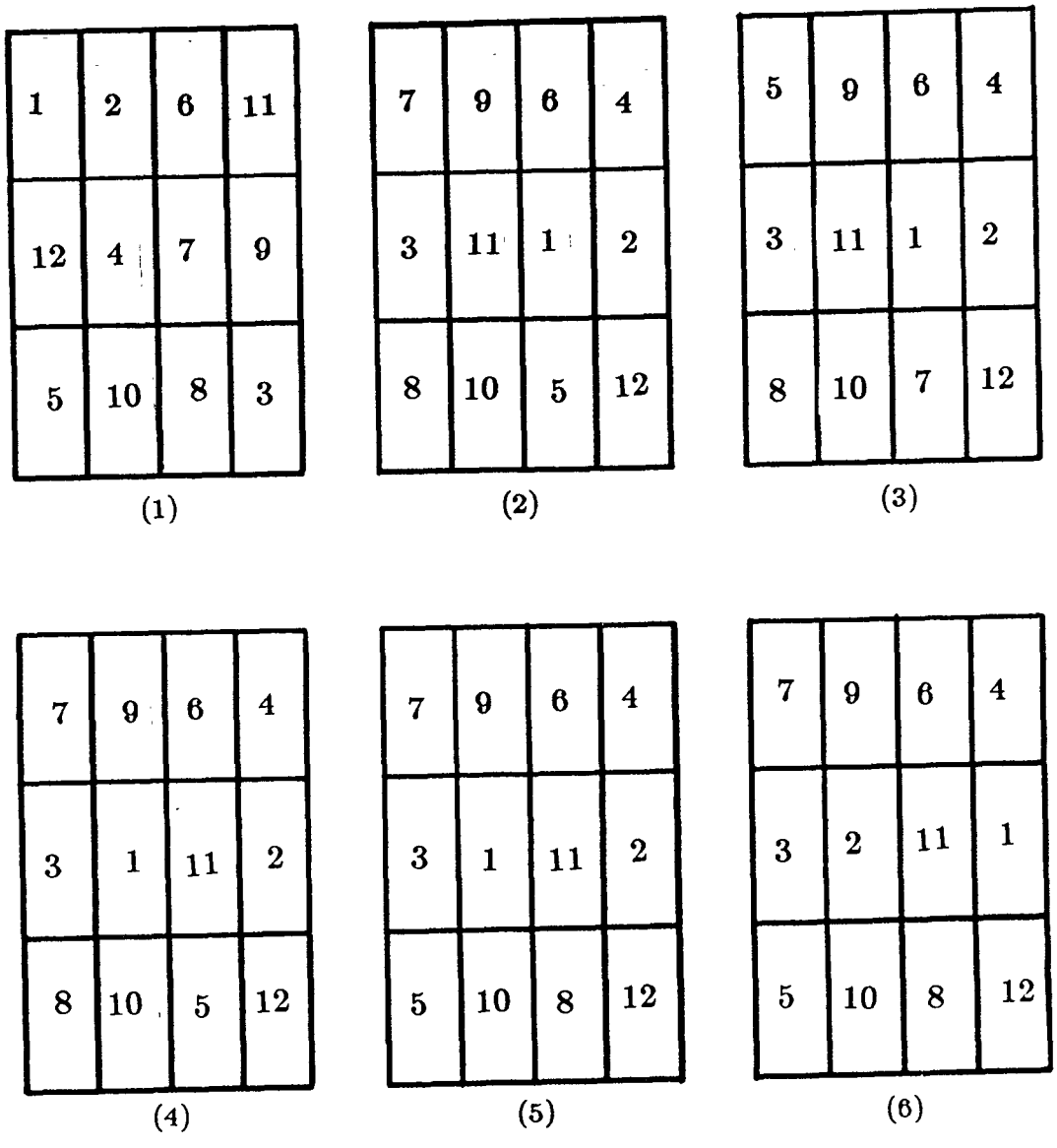


Fig. 4 An example of the two-subset placement