ZOR – Methods and Models of Operations Research (1991) 35: 61–84

Optimal Control of Plotting and Drilling Machines: A Case Study

By M. Grötschel, M. Jünger and G. Reinelt[1]

1 Introduction

Printed circuit board production is one of the basic technologies of electronics industry. A number of interesting and difficult mathematical problems arise in the design phase (e. g., placement of components, wire routing, contact minimization) and in the manufacturing process of printed circuit boards. We report here about two of the latter problems: plotting masks and drilling holes.

This paper is not meant to survey all mathematical techniques that are available and to show what can be achieved by a thorough and careful analysis of a practical problem and by developing sophisticated and special purpose algorithms. We simply want to show what we could achieve in an industrial setup with a prespecified tight time frame for the code development and with quite low limits on the running times for the algorithms.

The results were remarkable in the sense that relatively straightforward heuristics were able to improve the solutions used in practice by a substantial margin. In particular, the positioning head moves of the plotting and drilling machines could be reduced by about 10 % to almost 90 %. This led to reductions of the total production time by 5 % to almost 35 % and thus to a considerable increase of the production capacity of the machines on the average.

In Section 2 of this paper we outline the project and give a protocol of the cooperation with industry. Section 3 describes the plotting problem, its mathematical modelling and the algorithms we developed for its solution. The drilling problem is discussed in Section 4. A brief discussion of the timing function used in our models can be found in Section 5. Sections 6 and 7 contain the computational results for the plotting and drilling problems, respectively. Pictures of some plotting and drilling problems, our solutions and the original solutions are shown in the Appendix.

[1] M. Grötschel, M. Jünger and G. Reinelt, Institut für Mathematik, Universität Augsburg, Universitätsstraße 8, D-8900 Augsburg

0340-9422/91/1/61-84 $2.50 © 1991 Physica-Verlag, Heidelberg

2 The Project

The Siemens Werk für Systeme, Augsburg (Siemens WS), manufactures, among other electronic products, all main frame computers of the Siemens computer family. A major and quite sophisticated branch of this plant is the printed circuit board production. During one of the more or less formal contacts between Siemens managers and university officials (the distance between the Siemens WS and the University of Augsburg is only a few hundred meters) one of us pointed out various possibilities of productivity improvements by using Operations Research models and optimization techniques. After a series of meetings at the end of 1987 and in the beginning of 1988 it was suggested to try to optimize two types of NC-machines that – at times of high production volume – turned out to be bottlenecks in the production process: plotters for printed circuit board masks and drilling machines. Agreements were reached in July 1988 for a joint project.

Siemens was going to provide four instances of drilling problems and five instances of plotting problems that the Siemens engineers considered typical for the range of problem instances that occur in this particular production process. The team of the Chair of Applied Mathematics II of the University of Augsburg had to model the problems mathematically (in close cooperation with Siemens to make sure that the models were considered realistic) and to develop and code algorithms for the heuristic solution of the problems.

Knowing the problem sizes that arise in this area it was clear beforehand that the use of exact optimization algorithms was out of question. Moreover, a very tight bound on the running times of the algorithms was specified beforehand. *For any given instance, a solution has to be produced within 5 minutes CPU* time on *a 3 MIPS workstation.* Although we violently questioned that requirement our Siemens partners pointed out that this is a realistic constraint due to organizational rules in the present production environment. The final arrangement was to concentrate on very fast heuristics. We were asked, though, to code also more sophisticated and more time consuming methods to check whether further substantial improvements could be achieved by spending a lot more computer time.

Measuring success is not as easy in this case as it might appear. We first proposed to design lower bound heuristics so that we could come up with instance dependent (and hopefully good) performance guarantees for our approximative solutions. Since high quality lower bounding methods are quite time consuming (we saw no way – given the instance sizes – to achieve reasonable results within the specified tight time bounds) this option was ruled out. The agreement was to compare the solutions produced by the Siemens algorithms with our solutions using our mathematical model and then to compare the solutions in "real life" by running both alternatives on the real machines and measure the differences of the respective production times.

Having fixed the rules of the game by the end of July 1988 we made a contract for the development of experimental software as described above for the drilling and plotting problem to be delivered and tested on a total of nine typical Siemens instances by the end of November 1988. Siemens was going to compare our solutions of these problems with its own ones on its NC-machines in December 1988. All deadlines were met and we report here on the results of this case study.

3 The Plotting Problem

Complex printed circuit boards are usually produced by a photochemical process. For each layer of the board, the pattern of wires and contacts is produced by a sequence consisting of covering the board with light sensitive material, exposing this material to light, etching, cleaning etc. The process is similar to the usual production of photographs. The structures that later on should appear on the board have been "drawn" on a mask (a negative) that is between the board and the light source so that certain parts of the board are not exposed to light. These unexposed areas will finally form the conductors, pads, and contacts of the layer. The question we address here is the generation of the masks.

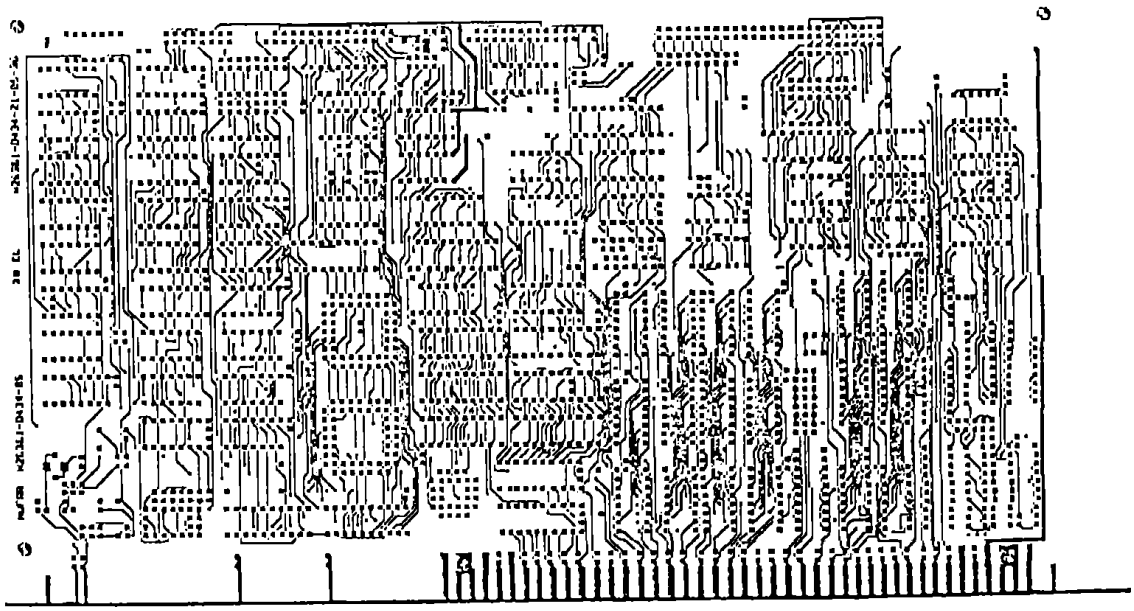

Fig. 1. A Layer of a printed circuit board

The masks are made of glass and the patterns on the glass are generated optically using either ultraviolet light or laser beams. In our case, a photo plotter is used for the mask production.

Figure 1 shows an example of one layer of a printed circuit board. It is one of our test cases.

The photo plotter (as basically all other plotters) works as follows. It has two modes, a "drawing mode" with which lines are plotted and a "flashing mode" to plot points. As one can see from Figure 1, points may be of various sizes and shapes and lines of different width. So, before plotting, an aperture has to be chosen that produces the required shape or width.

Points are plotted by moving the light source to certain coordinates on the board, choosing the aperture, and flashing the light. Lines are plotted by moving the head to one end of the line, choosing the aperture, opening the shutter, moving along the line with the open shutter and closing the shutter at the end of the (not necessarily straight) line.

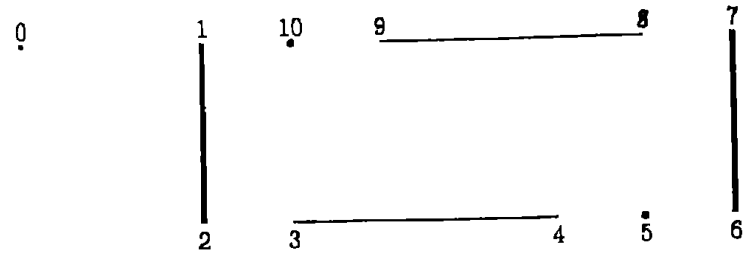

Fig. 2. First example problem

This is merely a description of the principle of the plotting process. Plotting machines vary concerning their mechanical realization. In some cases, only the head of the light source is moved, in some cases only the compound table, and in others the head moves in one direction and the table in the other. For our purposes, the "real mechanics" is irrelevant. We only need to know the time it takes to move from one point to another. For ease of exposition we will assume from now on that the head moves.

There is, given a pattern, nothing to be done about the time needed for drawing and flashing. This process requires a certain fixed time depending on the plotter characteristics. What can be optimized is the time needed for positioning head moves, i. e., moves of the head without drawing. (What we call "positioning head moves" is often called "wasted head moves", a slightly misleading term because such "wasted" moves are actually necessary!)

We will now describe the mathematical modelling of the plotting process in some detail since a number of publications in this area provide models that are either vague or imprecise or simply wrong. Even if one could solve these models to optimality, the solutions would just be heuristic (and not necessarily optimal) solutions of the real problem.

Depending on technological side constraints there are several options to model this problem mathematically. We discuss two examples that came up in our application to illustrate this.

Consider the line and point plotting problem shown in Figure 2. It seems obvious how to proceed. We begin at the starting position 0, move to point 1, choose the "drawing mode", and switch the aperture to "drawing thick lines", while moving, draw the line from 1 to 2, move to point 3 and change the aperture to "thin drawing", draw the line from 3 to 4, move to point 5 and change to the "point flashing mode" etc. The trouble here is that during every positioning move the aperture has to be changed. The head is (in many cases) a mechanically delicate device that – after a certain number of aperture changes – has to be readjusted (respectively substituted). This is a costly procedure. Therefore it may be wise to proceed as follows. One first chooses an aperture, plots everything that can be plotted with this aperture, changes the aperture etc. This approach decomposes the problem into various plotting subproblems and adds a new problem, namely, how to choose an optimal sequence of apertures. For Figure 2 an optimum solution in this case would be as follows. We first choose the thick drawing aperture, go from 0 to 1, draw the line from 1 to 2, move to point 6, draw the line from 6 to 7, change to the thin drawing aperture and move to 8, draw the line from 8 to 9, move to 3, draw the line from 3 to 4, change to the flashing mode and move to 5, flash, move to 10, flash, and return to 0.

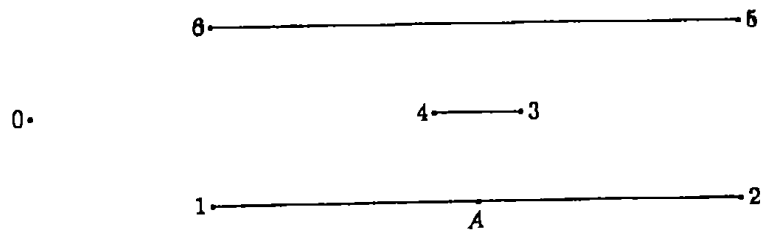

Fig. 3. Second example problem

Clearly, the second choice produces longer (and sometimes substantially longer) positioning head moves and – provided that aperture changes do not require a lot more time than the moves – longer machine running times.

In practice one has three choices. One either ignores aperture changes and uses the first option, one decides to decompose the plotting problem into subproblems (one for each mode and each aperture), or one mixes the two by penalizing those positioning moves where also aperture changes occur. Anyway, the person responsible for mask plotting has to decide – based on his knowledge of the technical characteristics of the machine – whether or not aperture changes

are considered crucial operations that have to be kept at minimum and which of the three options should be used.

There is another option that may produce shorter positioning head moves: preemptions. Consider Figure 3, where all three lines are drawn with the same aperture. An optimum solution here is to move from 0 to 1, draw the line from 1 to 2, move to 3, draw the line from 3 to 4, move to 5, draw the line from 5 to 6 and return to 0. We have made so far the assumption, without explicitly stating it, that whenever the plotter has started drawing a line it continues drawing until the other endpoint of the line is reached. If we allow interruptions (technically often called preemptions) we could do better. For instance, in Figure 3 we could draw the line from 1 to 2 only until point A is reached then we move to point 3, draw the line from 3 to 4, return to point A and continue drawing from A to 2 etc. This choice would produce shorter positioning head moves than the "optimum solution" sketched above. So the question is, whether preemptions should be allowed or not. There is no general answer to this. In each individual case one has to make a decision based on the particular conditions.

In our case the decision was to execute as few aperture changes as possible and not to allow preemptions. Therefore the following combinatorial optimization problems arose.

3.1 Point Flashing Subproblem

Given an aperture, select all points that are flashed with this aperture and determine a shortest hamiltonian path through these points.

(We will discuss later how to determine distances between points and whether the starting and terminal point of the hamiltonian path have to be predetermined.) □

3.2 Line Drawing Subproblem

Given an aperture, select all lines that have to be plotted with this aperture and determine a sequence of these lines such that the total distance travelled by positioning moves is as short as possible. □

3.3 Aperture Sequencing Subproblem

Determine a sequence of apertures and, for each aperture, a starting point and a terminal point for the point flashing or line drawing process such that the total time for positioning moves (with and without aperture changes is a short as possible. □

The aperture sequencing problem is by far the most complicated and we do not have any idea how to solve it. Fortunately, in our practical problems, it is

almost of no importance. In all our examples the maximum number of different apertures is nine and whether or not a good sequence of apertures is chosen or not is (practically) almost irrelevant. We thus treat it in a "very heuristic" manner as follows.

Wie first noticed that changing an aperture for a photo plotter is a very fast procedure. So aperture changing has little effect on the running time of the machine and we can simply concentrate on the time needed for head moves.

To heuristically solve the aperture sequencing problem we follow a nearest neighbor strategy. We begin at the starting position of the plotter head and search for the nearest point (endpoint of a line or flash point) and decide to move to that point changing to the aperture necessary to plot that line or flash that point. Then we solve the line drawing subproblem or point flashing subproblem for that aperture where the first point is the point chosen above. The solution will produce a final point p so that no further lines or points have to be plotted with the present aperture. Then we determine the closest point to p that has to be plotted with a different aperture, move to that point and change the aperture along the way. This procedure is continued until all lines and points are plotted.

Having made this decision we can now present a precise mathematical model of the point flashing problem (3.1) for a fixed aperture. We consider the points to be flashed with the given aperture as the nodes of a complete undirected graph $K_n = (V, E)$ for which we know how long it takes to move from one node i to another node j. This time will be called the "length" c_{ij} of the edge ij between nodes i and j. By our nearest neighbor algorithm for the heuristic solution of the aperture sequencing problem we have determined one node in K_n, say v.

The point flashing problem is now nothing but finding a shortest hamiltonian path through K_n that starts in v and that may have any other node of K_n as the other endpoint of the hamiltonian path. This problem is known to be $\mathfrak{NP}$-hard.

By introducing a further node $n + 1$ and edges from $n + 1$ to all nodes of K_n having length 0, with the exception of the edge from $n + 1$ to v that gets length $c_{v,\,n+1} = -M$ (M a large positive number) one can transform this hamiltonian path problem into a symmetric travelling salesman problem and solve it with exact or heuristic TSP algorithms.

The mathematical model of the line drawing problem (for a fixed aperture) is a little more complicated. We have treated it in the following way.

We consider the endpoints of all the lines to be plotted with the given aperture as the nodes of a complete undirected graph $K_n = (V, E)$. Each line has two endpoints, but there are cases where one point is the endpoint of two or more lines. Whether or not such cases come up depends on the designer of a printed circuit board. In our case such situations occured. Thus the number n of nodes of the complete graph K_n is at most twice the number of lines to be drawn. We view the complete graph K_n as the graph of possible positioning head moves and thus the "length" c_{ij} of moving from i to j is the time it takes to move from point i to

point j without plotting. (Note that moves without plotting may be much faster than moves in the drawing mode.) Now we extend K_n to a graph $\hat{K}_n = (V, \hat{E})$ by adding certain parallel edges, namely, for every line (with endpoints i and j, say) to be plotted we add a further edge from i to j. Let us denote the set of these edges by E', so $\hat{E}$ is the disjoint union of E and E'. As the "length" of an edge $ij \in E'$ we can either choose the time it takes to plot the edge from i to j or we can set it to 0. (It will be clear soon that this does not matter.) We set $c_{ij} = 0$ for all $ij \in E'$.

The nearest neighbour heuristic for the aperture sequencing problem determines a starting node, say v, for our "plotting trip". Using the terminology introduced above, the line drawing problem can now be stated in the following way. Determine a walk, (i. e., an alternating sequence $v_0, e_1, v_1, e_2, \ldots, v_{k-1}, e_k, v_k$ of nodes $v_0, v_1, \ldots, v_k$ and edges $e_1, e_2, \ldots, e_k$ such that, for $i = 1, \ldots, k$, the endnodes of e_i are the nodes v_{i-1}, v_i), that starts at node $v_0 = v$, contains all edges $e \in E'$ and has minimum cost $c_{e_1} + \ldots + c_{e_k}$.

This combinatorial optimization problem is a version of the (well known) **rural postman problem** and is $\mathfrak{NP}$-hard in general. There are some special cases, though, that are solvable in polynomial time. For instance, if the graph $G' = (V, E')$ induced by the edge set E' is connected then the rural postman can be transformed into a Chinese postman problem which has been shown in [2] to be solvable in polynomial time using Edmonds' blossom shrinking algorithm of [1] for the perfect matching problem. The transformation works as follows.

Let $G^* = (V^*, E^*)$ be the graph consisting of all the endnodes of the lines to be drawn (i. e., this is the node set V) and an additional node $n + 1$. E^* consists of all edges in E' and the edge linking $n + 1$ to the starting node v. Let V_0 be the set of nodes of odd degree in G^*. For any pair $i, j \in V_0 \setminus \{n+1\}$, let $\bar{c}_{ij}$ be the running time of a positioning head move from i to j. Set $\bar{c}_{n+1,i} := 0$ for all $i \in V_0 \setminus \{n+1\}$. We now determine a minimum weight perfect matching M of the complete graph on the node set V_0 where the edge costs are the $\bar{c}_e$ just defined. Since $n + 1$ has odd degree in G^* there is an edge in M containing $n + 1$, say the neighbour of $n + 1$ in M is node w. By adding the edge set $M \setminus \{n + 1, w\}$ to G' (parallel edges may occur) we obtain a graph that contains an Eulerian trail from v to w. This Eulerian trail is the optimum solution of our line plotting problem (with fixed starting node v).

In geographical map drawing applications, where, for instance, streets, cable networks, or pipeline systems have to be plotted a slightly different version of this situation often occurs. The graph to be plotted is connected but the drawing pen is in some initial position and has to select a starting point and a terminal point (from which it returns to the initial position) for its plotting trip. A naive solution of this problem would be to try all possible pairs of nodes as starting resp. terminal points of the tour and solve the corresponding Chinese postman problem as outlined above. For a graph with n nodes this gives rise to $\binom{n}{2}$

Chinese postman problems. A better method is to transform this problem into a b-cover problem (solvable in polynomial time with any b-matching algorithm, see [7] for a survey of these) on the graph G^* consisting of the given graph to be plotted and an additional node i (representing the initial position) that is linked to all other nodes by an edge and where $b_v = 0$ for each even node v, $b_v = 1$ for each odd node v and $b_i = 2$ for the initial position. ,

Many geographical plotting problems consist of graphs with very few components, and in such cases a very good heuristic (but slightly incorrect) model is to connect the components cheaply and solve the resulting Chinese postman problem. This approach leads in general to good approximate solutions of the rural postman problem, see e. g., [4]. However, in the PCB application discussed here, this approach produces quite an incorrect model since the given graphs are usually very disconnected, i. e., the number of components is in general linear in the number of nodes. Note that connectivity of the graph means that all wires of one layer are on the same electric potential, in other words, they form a single net. Clearly, this does (almost) never happen and, in fact, we do not see how a correct model of the given rural postman problem in the PCB application can be obtained by reducing the rural postman problem to a Chinese postman problem and in case "the graph which has to be traversed is not connected one has to connect its components first in a minimal way before constructing the perfect matching", as claimed in [5]. After all, the line plotting problem is $\mathcal{NP}$-hard.

These remarks finish our description of the mathematical modelling of the line plotting problem. It is more complicated than it appears at first sight since a number of technical side constraints play a role. As we pointed out, it is sometimes quite hard to produce a correct model, and in fact, it may be advisable – for practical purposes – to use a slightly incorrect model (as we do) that nevertheless is guaranteed to provide a feasible solution (which one hopes to be "good").

4 The Drilling Problem

The drilling problem for printed circuit boards is a standard application of the symmetric travelling salesman problem. In our case, a CNC mechanical drilling machine is used that can drill several (identical) panels on one stack and up to 6 stacks of identical panels in parallel.

The practical problem coming up in our application is the following. To connect a conductor on one layer with a conductor on another layer or to position (in a later stage of the PCB production) the pins of IC's, holes have to be drilled through the board. The holes may be of different diameters. To drill two holes of different diameters consecutively, the head of the machine has to move to a tool

box and change the drilling equipment. This is quite time consuming. Thus it is clear at the outset that one has to choose some diameter, drill all holes of the same diameter, change the drill, drill the holes of the next diameter etc.

There is no tool changeover problem here since, in any case, after loading of the boards the machine head is at the initial position (where the tool box is) and after having drilled all holes of one diameter it has to return to the initial position to pick up the new drill. Thus, our drilling problem can be viewed as a sequence of symmetric travelling salesman problems, one for each diameter resp. drill, where the "cities" are the initial position and the set of all holes that can be drilled with one and the same drill. The "distance" between two cities is the time it takes to move the head from one position to the other, see Section 5 for a discussion of the distance function. The aim here again is to minimize the travel time for the head of the machine. The (quite substantial) time needed to drill a hole cannot be influenced at all. This is a fixed production time.

Let us make a side remark. Combinatorial optimization can also contribute in the layout phase for printed circuit boards to reducing the drilling time. In the prevailing approach of printed circuit board design, first components are placed, then a so-called transient routing of the conductors is determined, and then the conductors are assigned to different layers of the board such that different nets do not cross. There are fast methods available to assign conductors to layers such that the number of contacts (= holes to be drilled) is minimum, see [3]. Thus a careful layer assignment may help to speed up production in the drilling process.

Coming back to the drilling problem, the question is which heuristic to use to obtain good solutions within the given time frame.

5 The Length Function

For the modelling of the plotting as well as the drilling problem we used edge lengths c_{ij} that we assumed to correctly reflect the time a light head or drilling head needs to move from point i to point j. In theory the c_{ij} should be easy to determine, in practice they are not.

There are two motors that drive the head, one in x-direction the other in y-direction. These motors (in general) do not have the same speed. E. g., in our drilling application the movement in one direction is 10 % slower than in the other. (Movements in one direction are performed by moving the head, and in the other direction by moving the table on which the stack of PCBs resides.) Given two nodes i and j that correspond to points (x_1, y_1) and (x_2, y_2) in the plane the x-motor moves the head from x_1 to x_2 while the y-motor moves the table from y_1 to y_2 (or conversely). The move that takes longest determines the running time of the head move. The time needed to move in x- or y-direction is hard to determine. Starting at x_1, say, the motor accelerates until maximum speed is reached then

runs at maximum speed for a certain distance and slows down to stop at x_2. Of course, if points are close the motor may not even have reached full speed before the slow down phase begins. From the technical data of a machine one can in fact compute c_{ij} exactly. However, the times obtained this way do not have too much to do with reality. The Siemens engineers have run a couple of tests to compare the actual running times with the computed ones. They differ considerably.

Since no accurate and realistic timing function could be found this way we agreed to use a modified maximum norm of the type

$$c_{ij} = \max\{\alpha|x_1 - x_2|, |y_1 - y_2|\}$$

where α accounts for the difference in speed between the two directions. So our timing function has basically been reduced to a distance function that is a modified maximum norm, i. e., the times reported in our computational results for positioning head moves will be nothing but total distances computed in the way described above. This is certainly not ideal, but seemed to be the only reasonable compromise under these circumstances.

6 Results for the Plotting Problem

Siemens provided us with five plotting problems for printed circuit board masks. Table 1 shows the characteristics of these problems which are denoted by unil, ..., uni5 in the sequel. We also list the total number of aperture changes and the total length of positioning moves for the industry solution. All lengths are recorded in machine units according to our distance function (i. e. maximum distance).

We should mention here that the industry solutions had been already obtained by applying heuristics. These heuristics were based on a sorting of flashing points and starting points of lines with respect to their x- and y-coordinates.

As we have pointed out in Section 3, we decompose each plotting problem into a sequence of point flashing and line drawing subproblems.

A point flashing subproblem is essentially a shortest hamiltonian path problem with fixed starting point and arbitrary end point. By adding one node and suitable edge weights we obtain an equivalent symmetric travelling salesman problem as pointed out before.

We demonstrated in Section 4 that the line drawing subproblem can be transformed to a Chinese postman problem if the graph to be plotted is connected. However, in all five testcases of this study, the graph was highly disconnected; in fact, almost all points were of degree one. So we had to discard an initial attempt

to add "virtual edges", i. e., additional positioning moves and then solving a perfect matching problem.

Table 1.

	uni1	uni2	uni3	uni4	uni5
Number of drawn lines	6139	869	1360	49	38621
Length of lines	19123502	2102549	25552950	4761800	124351961
Number of flashes	2157	2496	1477	2478	1060
Number of apertures	7	9	5	5	5
Number of aperture changes	33	261	5	5	5
Length of positioning moves	41285752	26445205	77629210	38382300	296730563

Instead, we used a simple transformation that allows us to apply modified heuristics for the travelling salesman problem to determine good solutions for the line drawing subproblems. This goes as follows.

Suppose we have to plot a collection of m lines with some specified aperture. Each line has two endpoints. We denote the endpoints of line i by i and $m + i$, and we ignore that some of these endpoints are identical in reality. Let node 1 correspond to the starting point selected by the aperture sequencing heuristic. We represent the line drawing problem on the weighted complete graph $K_{2m} = (V, E)$ on $2m$ nodes (corresponding to the endpoints of the lines). The edge weights c_{ij}, for every pair ij of nodes, are the time needed to move the head from i to j. We can assume that $c_{i,\,m+i} = 0$.We now look for a hamiltonian path in this graph starting at node 1, terminating at an arbitrary node and containing all the edges $\{i, m + i\}$ and is as short as possible.

Since we have a complete graph, it is easy to find reasonable hamiltonian paths satisfying this additional constraint. E. g., a simple modification of the standard nearest neighbour heuristic finds a feasible solution. We start at node 1 and go to node $m + 1$. From here we proceed to the nearest neighbour of $m + 1$ (excluding node 1). If i is this neigbour we immediately add also the edge from i to $m + i$ if $i < m$, resp. from i to $i - m$ if $i > m$. Then we look for the nearest

neighbour of this new endnode and so on until all nodes are in the path. We can now apply standard TSP improvement techniques where we only have to guarantee that no special edge $\{i, m + i\}$ is eliminated by local changes of the path.

So in our approach, all subproblems arising from a plotting problem can be treated as symmetric travelling salesman problems with possibly additional constraints.

Many heuristic algorithms are known for the TSP (see e. g., [6]), but due to the sizes of the problem instances and to the severe CPU time restrictions most of them could not be applied directly. To meet the time limits, our road of attack was to reduce the problem sizes considerably and to apply elaborate heuristics to the reduced problems.

There are lots of ways of problem size reductions. After some experiments, we finally settled on a geometric approach and used the geometric structure of the points on the PCB to generate smaller problems that "faithfully" represent the original ones. Of course, the 2-dimensional geometry of the points does not exactly reflect their "machine distances" because of the complicated machine timing function. However, it is reasonable to assume that there is a strong relation between the maximum (or even Euclidean) distance in the plane and the positioning time of the machine.

It is impossible to describe here the various strategies we tried, their effects on the running time and on the solution quality in full detail. This can be found in [8]. We will just give a quick outline of the approach and of the composite heuristic that finally came into use.

To do our computations fast we used a bucketing procedure to reduce the number of points without loosing too much information about their geometry. To this end, the enclosing rectangle for all points is recursively subdivided into four equally sized parts by a horizontal and a vertical line until each arising rectangle is small enough and contains no more than a specified number of points. We then represent each rectangle by the center of gravity of the points contained in it. Then a tour through these representative points is computed. Since we have considerably fewer nodes in this representing system we can apply more elaborate heuristics here to produce a high quality tour of the reduced problem. This tour indicates how a good tour through the original points should "globally" look like. Finally we insert the original points into this tour by looking for the best insertion point in their neighborhood and remove the representative points. This heuristic meets the practical requirements. It is fast (with precisely estimable running time) and can be tuned using several parameters (rectangle sizes, heuristics to be applied on the representative nodes, etc.) depending on the actually available CPU time.

Table 2 displays the results obtained by this heuristic and the CPU time used on a SUN 3/60 workstation.

Table 2.

	uni1	uni2	uni3	uni4	uni5
CPU time (min:sec)	4:33	2:36	1:37	3:47	1:19
Number of aperture changes	7	9	5	5	5
Length of positioning moves	17731273	16345805	26870050	32935300	49737209
Improvement w. r. t. industry solution in %	57.05	38.19	65.39	14.19	83.24

Of course, this table only reflects the improvement in the length of the positioning moves measured in the maximum distance. Our new solutions were then compared with the industry solutions in real runs of the photo plotter. The decrease in the overall production time was 15.77 %, 33.33 %, 23.68 %, 7.98 %, and 8.78 % for the respective problems.

Still there is a considerable decrease in production time. The small decrease for problem uni5 of 8.78 % compared to the enormous decrease in the length of the positioning moves is explained as follows. This problem has a large number of long lines to be drawn. So the time to plot these lines exceeds the positioning time by far.

If more CPU time is available one should attempt to find better TSP solutions. For practical purposes the following is a reasonable approach. Compute a suitable set of "candidate" edges which is likely to contain good tours or at least many edges that are contained in good tours. Then modify known heuristics in such a way that priority is given to these edges, i. e., tour construction methods try to construct tours contained in this set and local improvement heuristics try to insert candidate edges into the current tour. It is clear that depending on the candidate set this approach can speed up computation enormously. In [8] it is reported how reasonable candidate sets can be computed very efficiently using methods from computational geometry and that very good solutions can be obtained based on these sets.

In a second experiment we ran much more sophisticated and more CPU time consuming algorithms to get an impression of the improvement potential still left in the solutions of Table 2. The corresponding results are displayed in Table 3.

Table 3.

	uni1	uni2	uni3	uni4	uni5
CPU time (min:sec)	128:01	15:57	16:29	13:30	57:47
Number of aperture changes	7	9	5	5	5
Length of positioning moves	15611483	15337249	23499950	30871300	41270713
Improvement w. r. t. to industry in %	62.19	42.00	69.73	19.57	86.09
Improvement w. r. t. Table 2 in %	11.96	6.17	12.54	6.27	17.02

These solutions were not tested in the production environment because the production engineers were very much satisfied with the solutions computed with our fast heuristics.

7 Results for the Drilling Problem

Table 4 shows the characteristics of the four drilling problems that were available to us. The problems will be referred to by da1, da2, da3, and da4 in the sequel. The total sum of the lengths of the positioning moves is again given in machine units as computed by our distance function. As in the case of the plotting problem the Siemens engineers used a heuristic based on sorting the holes with respect to their x- and y-coordinates to find reasonable drilling squences for the holes.

Table 4.

	da1	da2	da3	da4
Number of holes	2457	423	2203	2104
Number of drills	7	7	6	10
Total length of moves	3518728	1049956	1958161	4347092

For the purpose of this project we first applied fast travelling salesman heuristics as described in Section 6 on the single drilling problems. Table 5 displays the result of these heuristics and the running times on the SUN 3/60 workstation.

Table 5.

	da1	da2	da3	du4
CPU time (min:sec)	1:58	0:05	1:43	1:43
Length of positioning moves	1695042	984636	1642027	1928371
Improvement w. r. t. to industry in %	51.83	6.22	16.14	55.14

Using the solutions of Table 5 the total production time on the drilling machine including the time to drill the holes was measured by Siemens engineers. For technical reasons this was only possible for the problems da1 and da4. It turned out that the solutions of our fast heuristics resulted in an improvement of the total machine time of 5.98 %, respectively 10.06 %. Note that the improvement in overall machine time heavily depends on the number of holes to be drilled since the drilling time is considerable.

As mentioned in the outset we tried to convince the engineers to spend more computing time to find good solutions. So far we did not succeed. Just to indicate further potential we ran the more elaborate TSP heuristics on the problems. Table 6 indicates further improvement possibilities.

Table 6.

	da1	da2	da3	da4
CPU time (min:sec)	96:03	6:27	169:15	70:03
Length of positioning moves	1517748	896669	1430556	1809556
Improvement w. r. t. to industry in %	56.87	14.60	26.94	58.38
Improvement w. r. t. to table 5 in %	10.46	8.93	12.88	6.16

8 Final Remarks

The results obtained in the case study presented above were considered quite remarkable by the engineers responsible for the NC-machines. The fact that nothing but reorganization of the plotting and drilling sequence without technical changes and improvements could increase the capacity of the machines considerably had not been expected. Thus the plan arose to transform our case study into production. The decision was made to turn our prototype algorithms into real production codes to be used in everyday production. The realization has taken place in 1990.

Acknowledgement: We would like to thank the printed circuit manufacturing department of Siemens Augsburg for the pleasant and very constructive cooperation. We have all learned a lot from each other.

References

[1] Edmonds J (1965)Maximum matching and a polyhedron with 0,1-vertices. Journal of Research of the .National Bureau of Standards B 69: 125–130

[2] Edmonds J, Johnson E L (1973) Matching, Euler tours and the Chinese postman. Mathematical Programming 5: 88–124

[3] Grötschel M, Jünger M, Reinelt G (1989) Via minimization with pin preassignments and layer preferal: A computational study. Zeitschrift für Angewandte Mathematik und Mechanik 69: 393–399

[4] Asano T, Edahiro M, Imai H, Iri M (1985) Practical use of bucketing techniques in Computational Geometry. In: G T Toussaint (ed.) Computational Geometry. Elsevier Science Publishers BV North-Holland, Amsterdam: 153–195

[5] Korte B (1989) Applications of Combinatorial Optimization. In: M Iri, K Tanabe (eds.) Mathematical Programming: Recent Developments and Applications. Kluwer Academic Publishers, Dordrecht: 203–225.

[6] Lawler E L, Lenstra, J K, Rinnooy Kan A H G, Shmoys D (1985) The Traveling Salesman Problem. Wiley, Chichester

[7] Nemhauser, G L, Wolsey L A (1988) Integer and Combinatorial Optimization.Wiley, New York

[8] Reinelt G (1989) Fast heuristics for large geometric travelling salesman problems, Report No. 185, Schwerpunktprogramm der Deutschen Forschungsgemeinschaft. Universität Augsburg, Augsburg

A Appendix

We illustrate the improvements in the length of positioning moves in this appendix. We have selected two plotting problems and two drilling problems. The following pages display in that sequence

- The masks corresponding to problems unil and uni2 (Figures A1 and A2)
- Industry solution and improved solution (fast heuristics) for unil (Figures A3 and A4). (Only positioning moves are shown.)
- Industry solution and improved solution (fast heuristics) for uni2 (Figures A5 and A6). (Only positioning moves are shown.)

The next pages show similar pictures for two drilling problems. The corresponding boards to be drilled contained a few (5–8) holes relatively far away from the "real" holes. These were used by the engineers for testing purposes. To achieve reasonable figure sizes these holes and positioning moves to them were cut off.

- Holes to be drilled in problems da1 and da4 (Figures A7 and A8).
- Industry solution and improved solution (fast heuristics) for da1 (Figures A9 and A10)
- Industry solution and improved solution (fast heuristics) for da4 (Figures A11 and A12).

Received 10.5.90

uni1

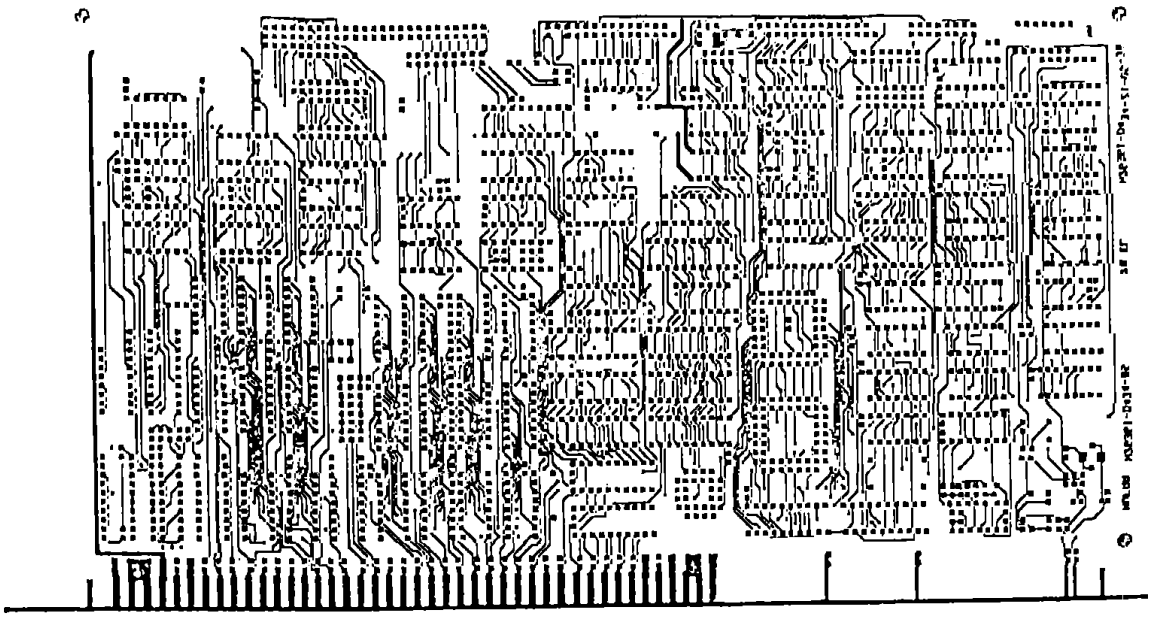

Fig. A1.

uni2

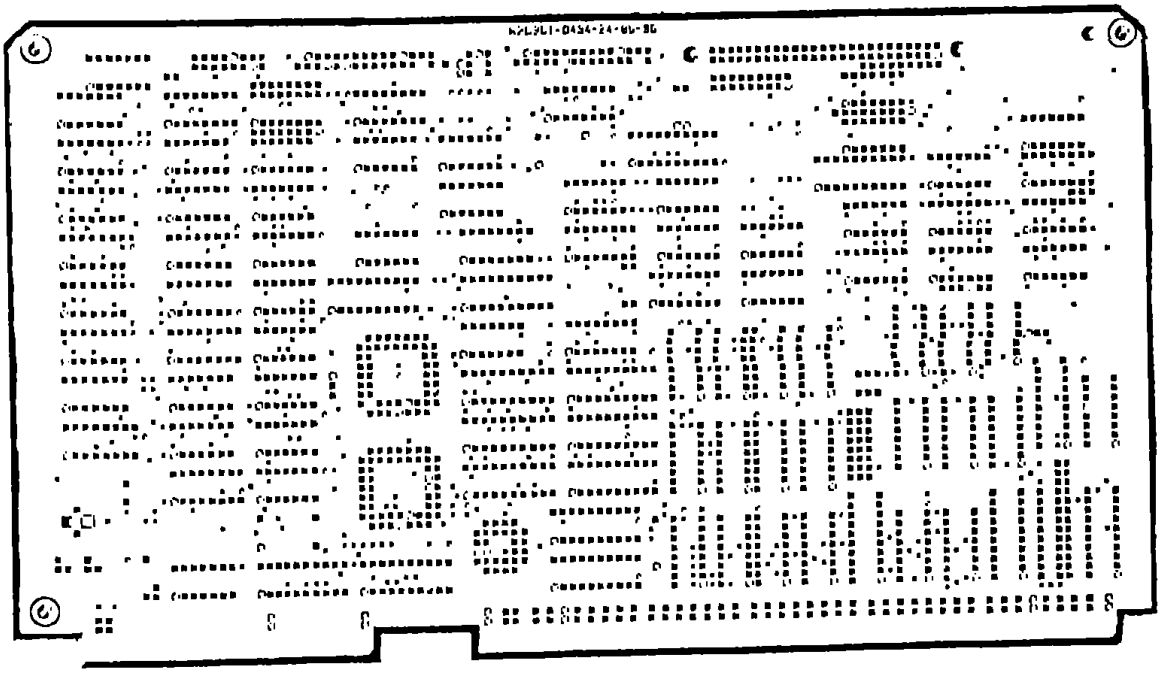

Fig. A2.

uni1

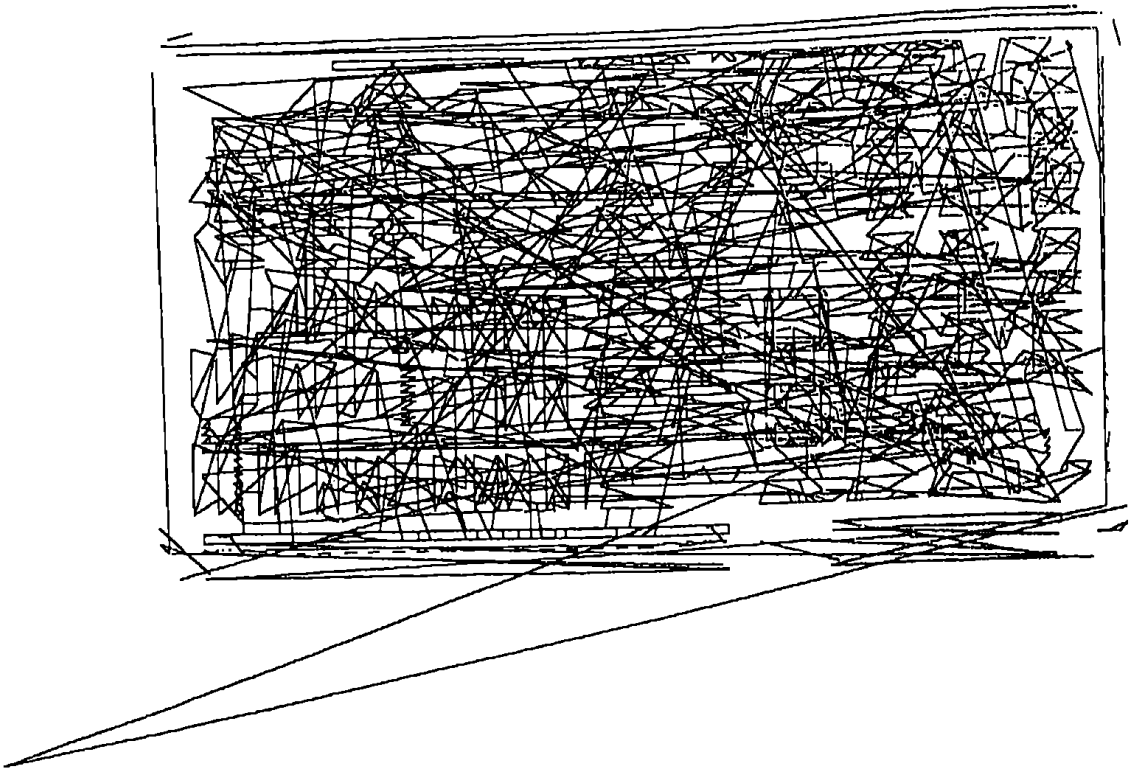

Fig. A3.

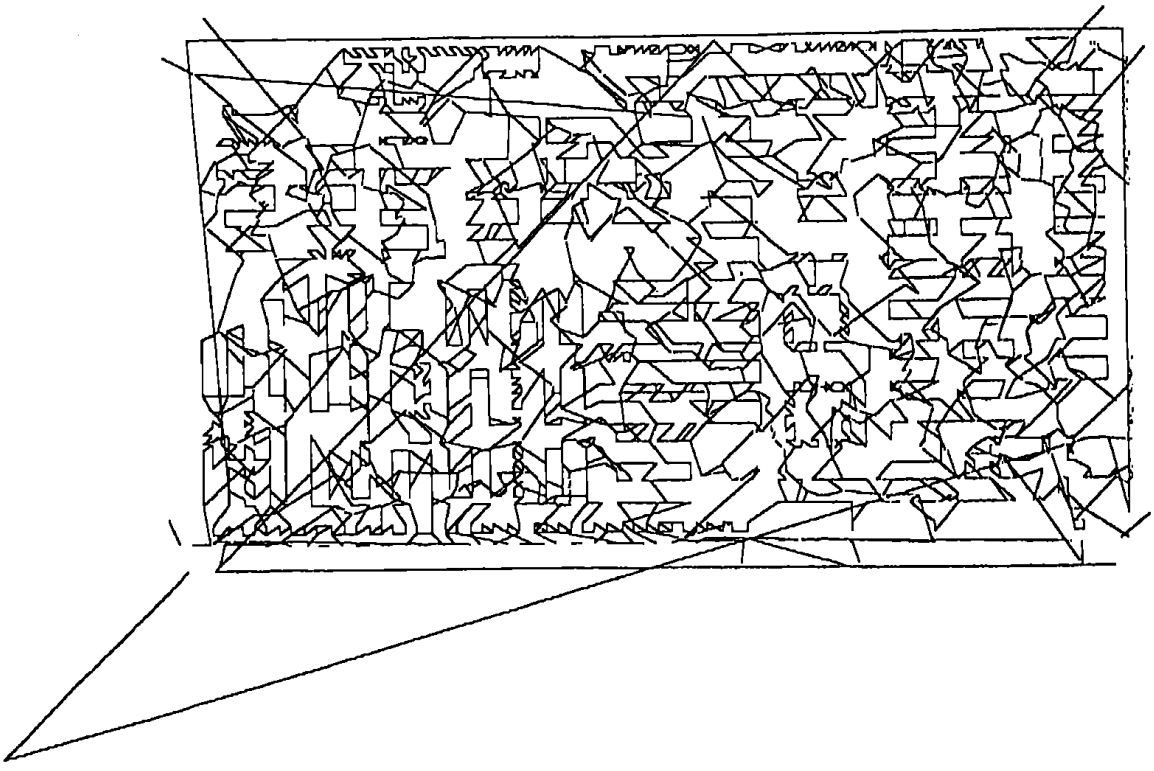

Fig. A4.

uni2

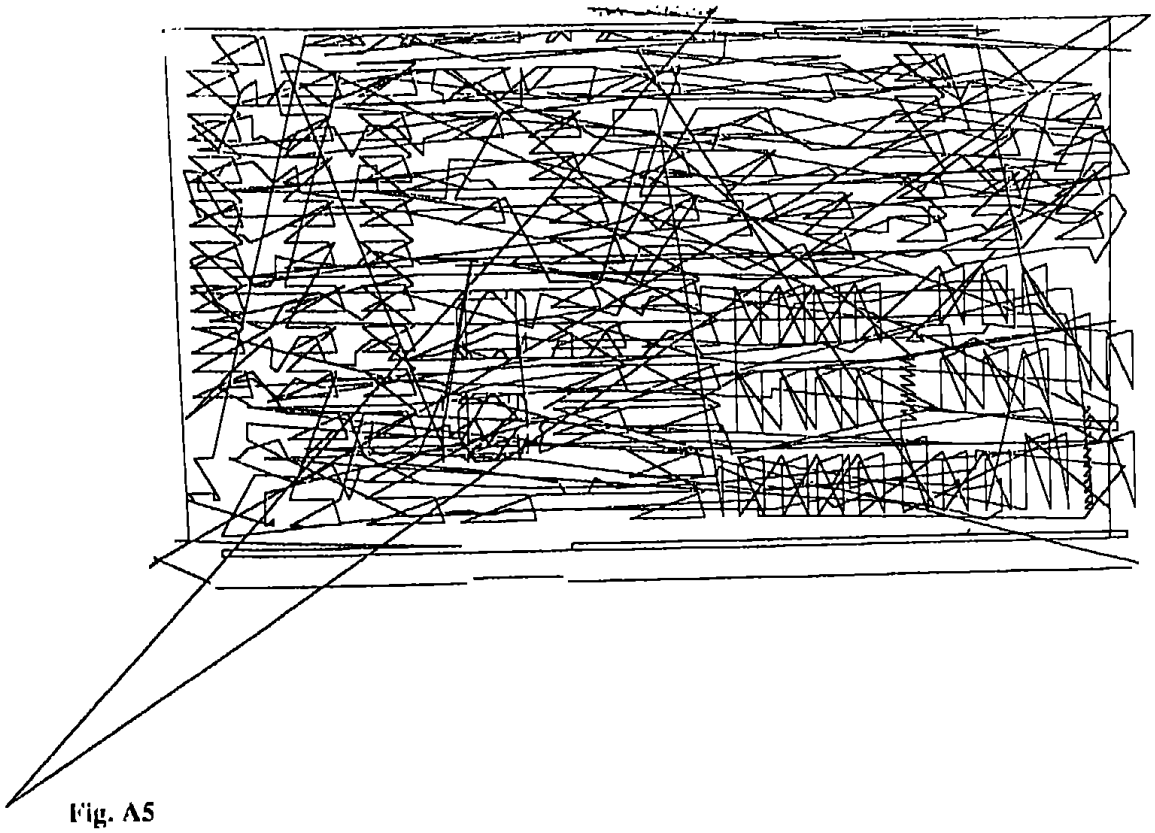

Fig. A5

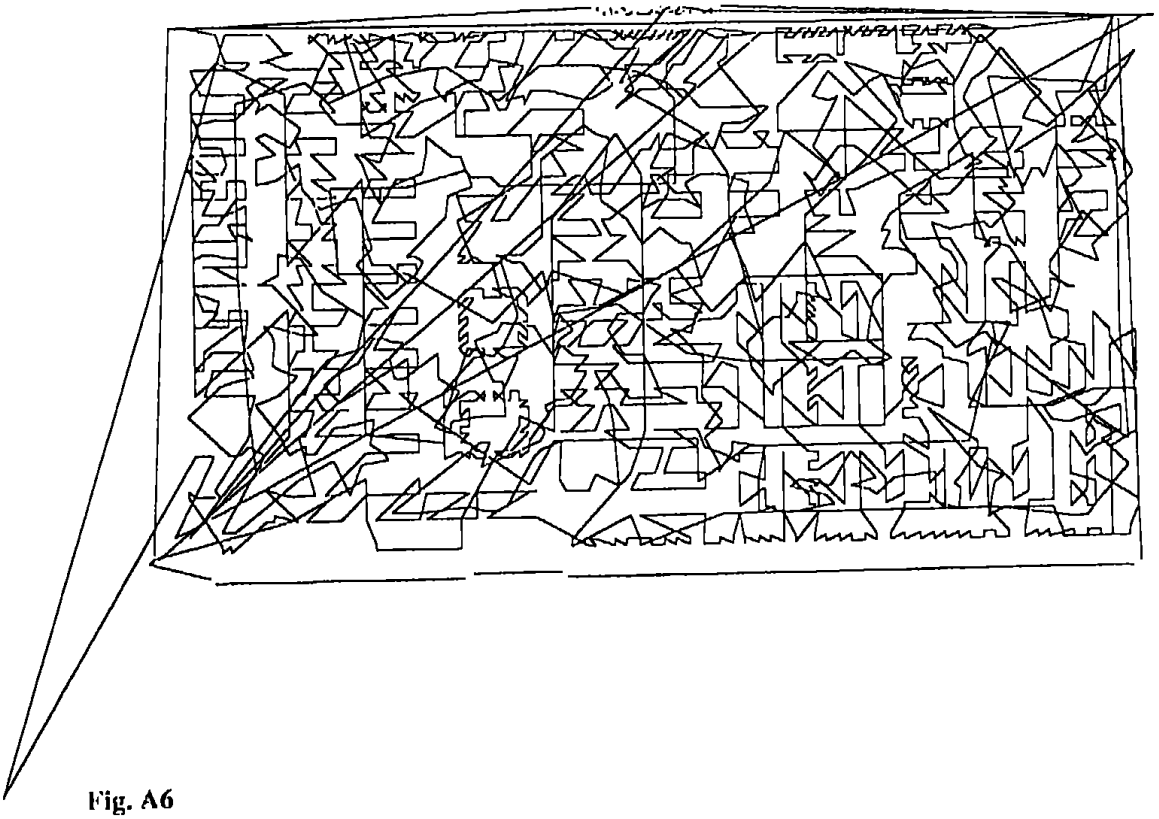

Fig. A6

da1

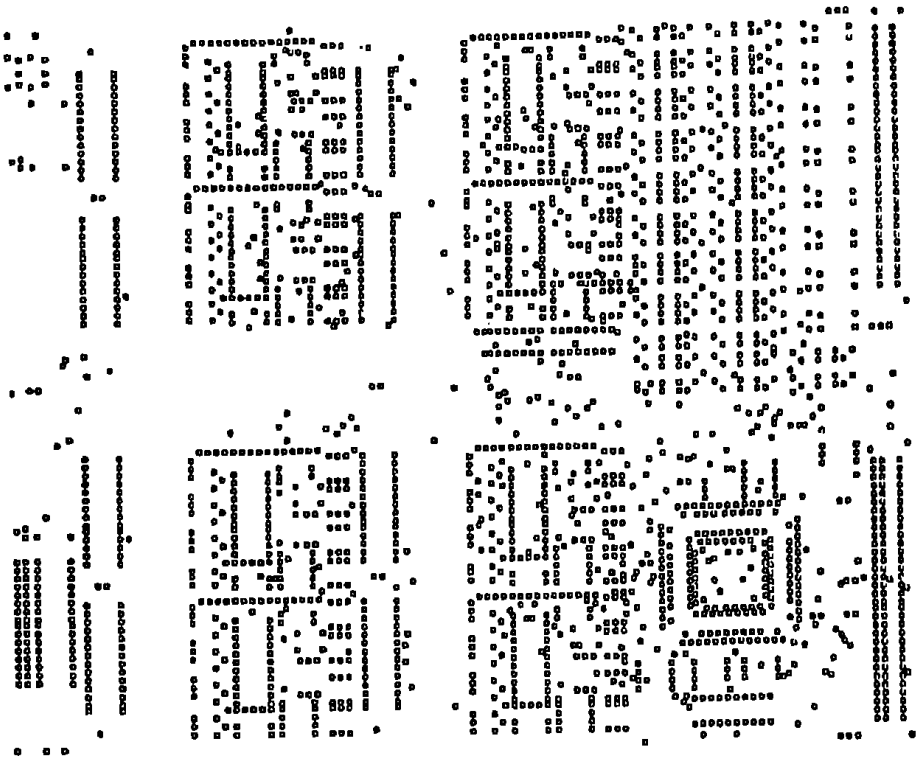

Fig. A7

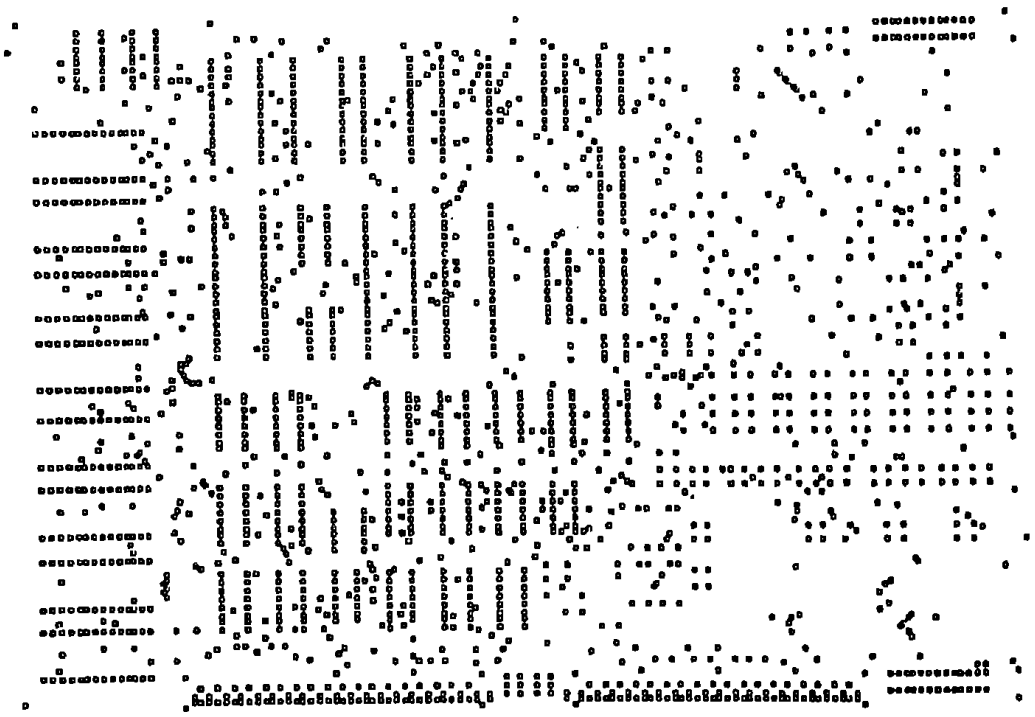

Fig. A8

dal

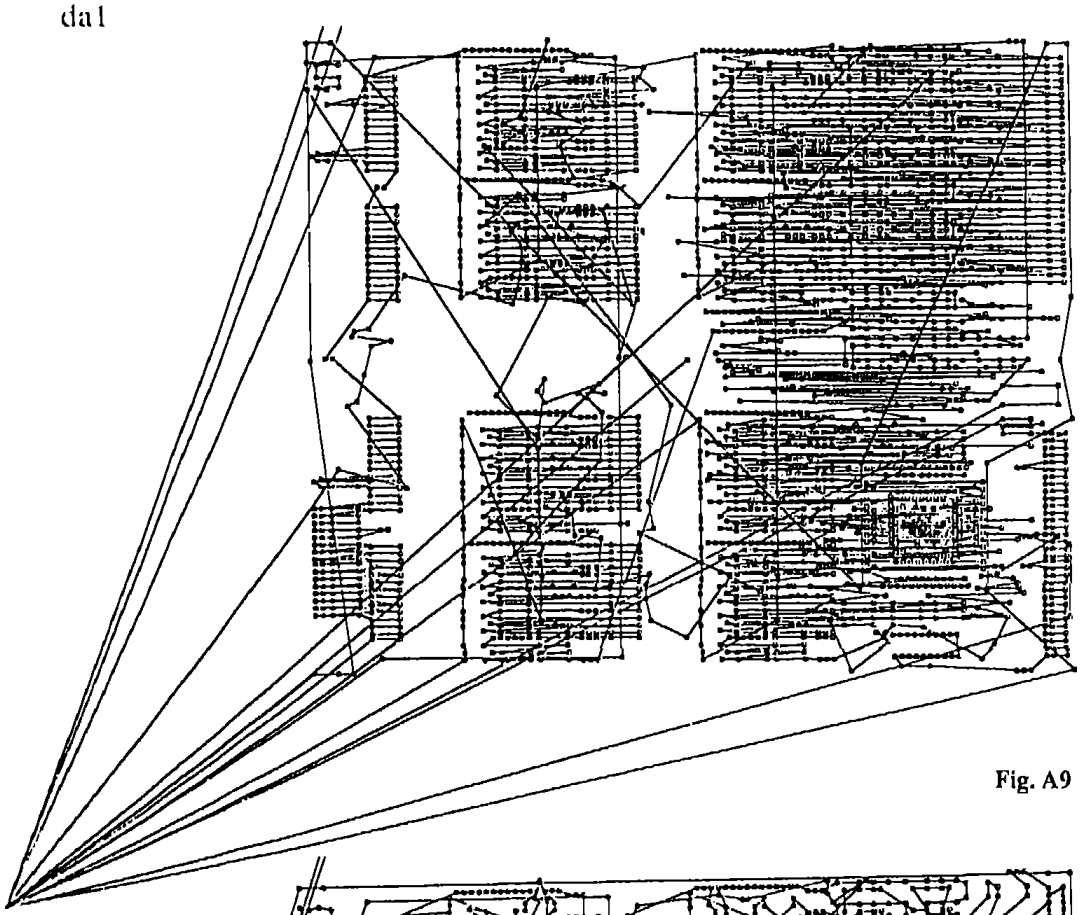

Fig. A9

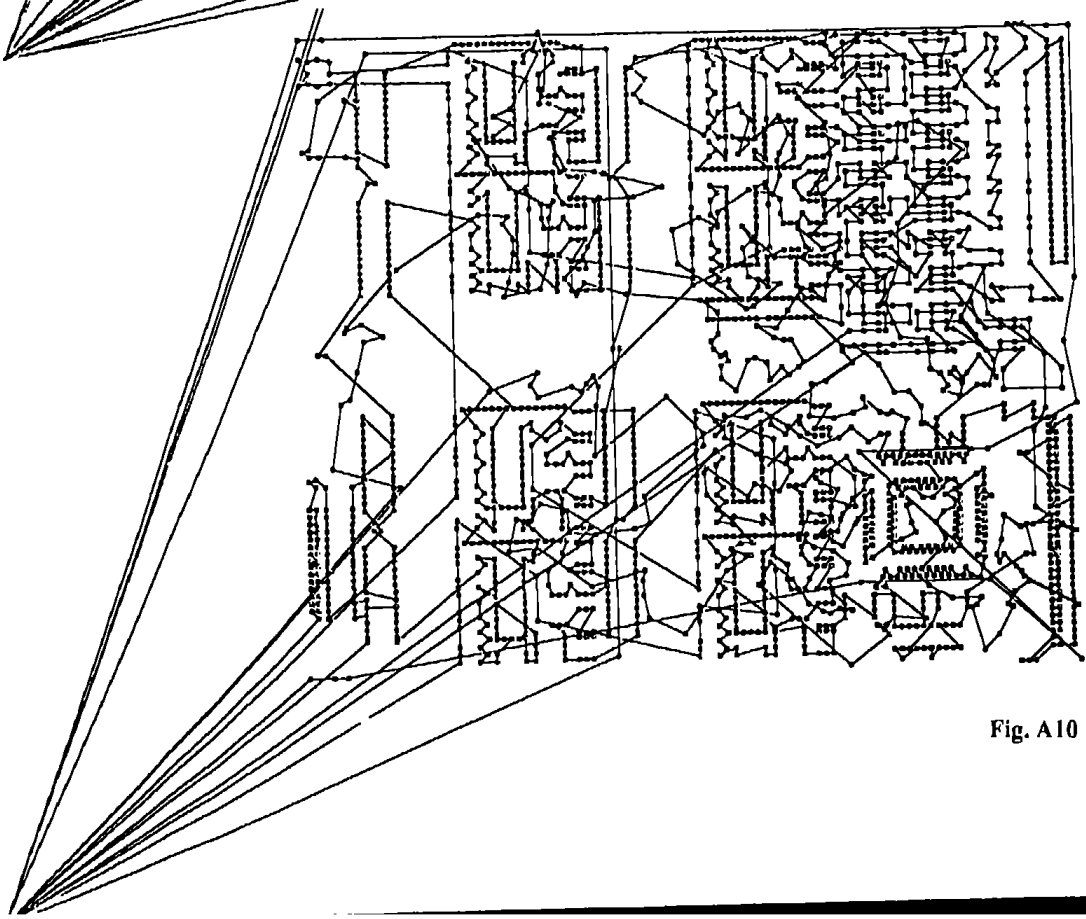

Fig. A10

da4

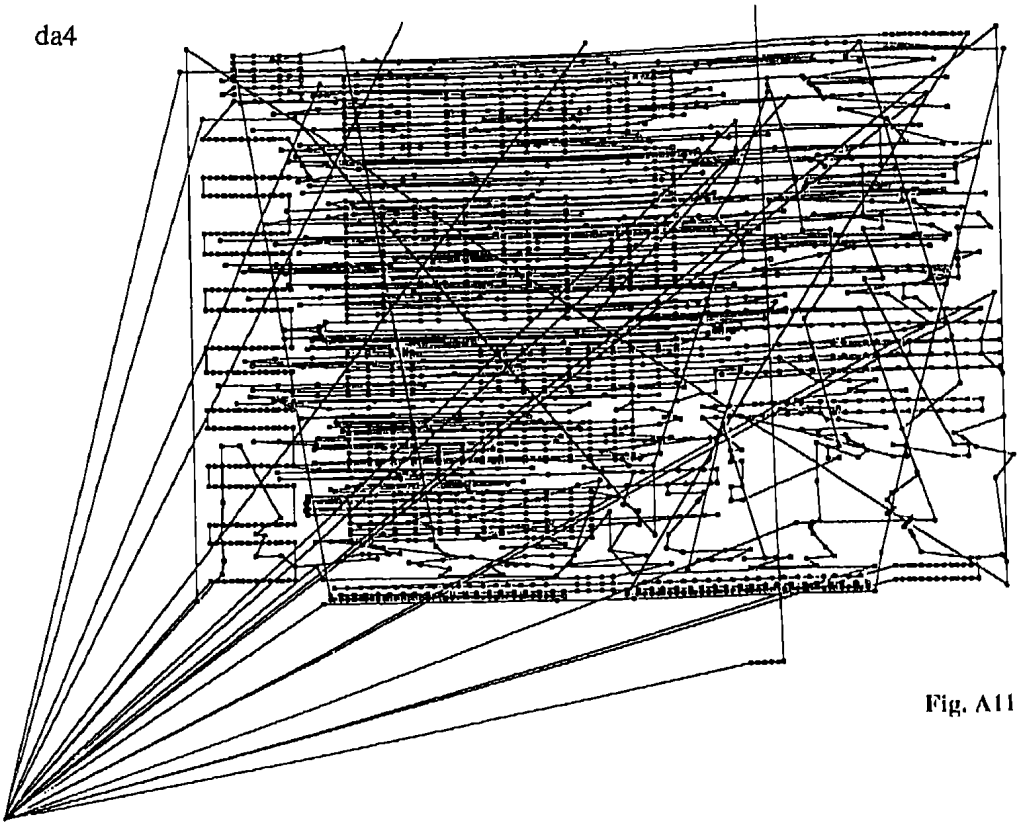

Fig. A11

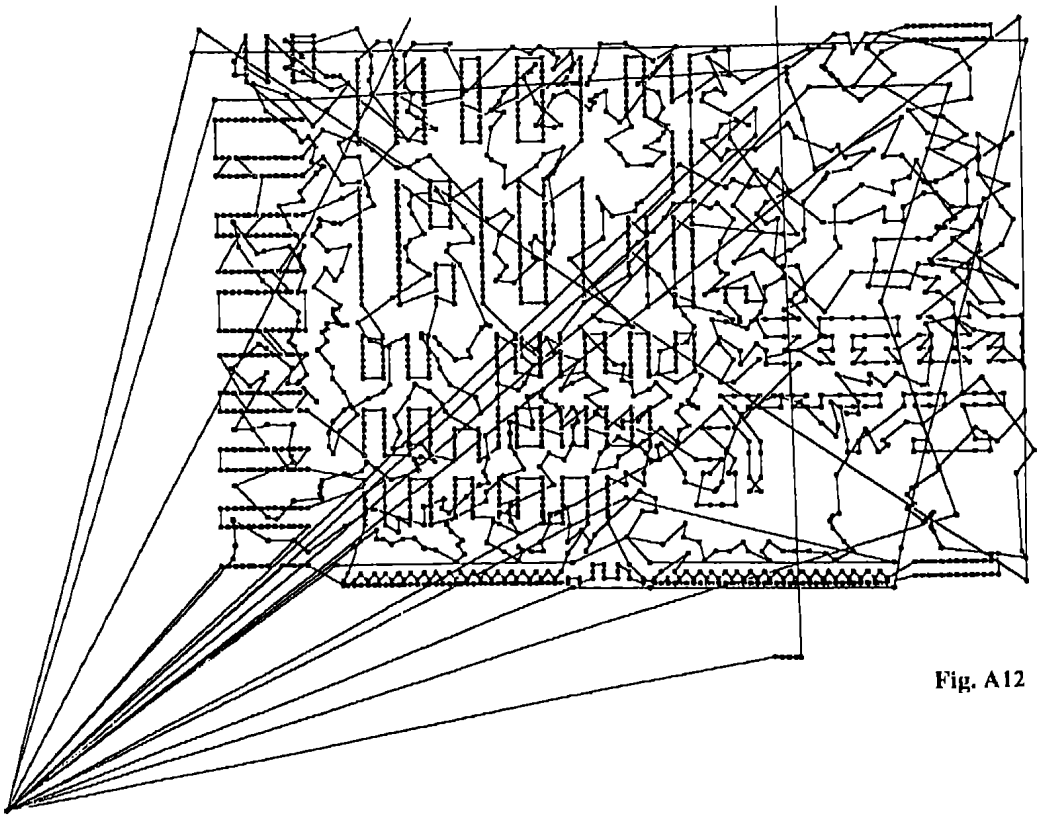

Fig. A12