A plotter sequencing system

T. Leipälä and O. Nevalainen

Department of Computer Science, University of Turku, 20500 Turku 50, Finland

Plotting time depends on the sequence of plotter operations. The system described in this paper selects a sequence in which the movements made while the pen is up are reduced. The separate lines forming a picture are temporarily stored in an internal memory area, from which the line nearest the present location of the pen is plotted next and replaced by a new one. The performance of the sequencer was experimentally tested.

(Received March 1978; revised October 1978)

1. Introduction

Digital incremental drum plotters are widely used graphic output devices. Because of the unrestricted length of the paper on the spool, they make possible the continuous processing of successive plotting tasks, without manual intervention by the operator. This is a desirable feature for a computer installation which performs daily a large number of scientific application programs of varying size and at unplanned arrival times, e.g. university computer centres with multiprocessing. When the workload of the plotter increases, it becomes essential to study the efficiency of plotting operations. If the operation of an output device like the plotter can be improved, a better throughput can be obtained and it becomes possible to manage with a slower and cheaper device.

A plotter sequencing system is described in the present paper. The system works in a multiprocessing environment and has been developed according to preliminary plans drawn up by Nevalainen and Vesterinen (1978) for a single-user operating system.

We consider the case where the picture comprises a set of separate *lines*, which are described, on the level of a general purpose programming language (in the following, FORTRAN), by *straight line segments*, represented by their base points (**Fig. 1**). These are specified by the user. The straight lines between the *base points* can be plotted by standard plotter routines in which the algorithms of Bresenham (1965) and Freeman (1969) are used to control the movements of the pen. In this paper, however, we do not need to know how the straight lines are actually plotted.

The time used for the plotting operation consists of two components, namely the time used for plotting the actual lines and the total time used for transfers from a finished line to a new one. The pen and the drum are moved similarly in both cases, but in the first case the pen is down and in the second case it is up. Movements of the second kind are called wasted movements, and their total length can be shortened by selecting a favourable sequence of lines. On the other hand, the time used for the actual plotting of the lines is independent of the drawing sequence.

If the pen must finally be returned to its initial point, the problem of minimising the wasted movement can be formulated as the so-called *tube passing* problem discussed by Liesegang (1976). In this problem we have a number of separate 'tubes' (lines in our case) and we must choose a route that passes through each tube in such a way that the total length of the route is minimal. Liesegang gives a branch and bound algorithm for the exact solution of the problem. The execution time, however, depends exponentially on the number of tubes. Thus the exact algorithm cannot be used in plotter sequencing, where the number of lines can be very large and the sequencing time is limited.

Closely related to the plotter sequencing problem is the *stacker-crane* problem of Frederickson *et al* (1978), where the movements of the crane, corresponding to the movements of the pen, must be performed in a predetermined direction. In this problem we have a crane which carries goods between fixed points. After unloading, the following loading point is selected in such a way that the total route taken by the crane is minimal. Thus the plotter sequencing problem could be called an undirected stacker-crane problem. A number of variations of the basic problem have been analysed by Frederickson *et al* (closed tour, fixed initial point, etc.). All these problems are shown to be 'NP-hard', i.e. no algorithm working in polynomial time is known for them and it is most unlikely that one exists (see Reingold, *et al*, 1977). Analogously, the plotter sequencing problem can be shown to be NP-hard and we face a difficulty that is common in systems programming: to find the exact optimum we need more resources than when plotting without any sequencing. We therefore restrict ourselves to the use of an approximate algorithm that is fast enough, simple to implement and by means of which we can expect to obtain a reduced plotting time.

An additional difficulty arises from the fact that no restrictions are assumed for the number of lines in the pictures to be plotted. This causes problems in the management of internal storage (at least in computer systems with no virtual memory facilities). We require that the sequencer should operate in random access storage of limited size. This leads to a solution in which only a part of the whole picture is stored at any given moment and the sequencing is performed for a dynamically varying subproblem.

Two sequencing rules, the so-called *k-change* rule in which the drawing sequence is locally perturbed in order to find a sub-optimal solution and the so-called *nearest neighbour* rule which simply selects the nearest continuation point to the pen, were tested in the paper by Nevalainen and Vesterinen (1978) mentioned above. The whole picture was processed at the same time and a simple random model of the picture was used. The results when the latter rule was applied (in the context of scheduling problems also called the *next best* rule) were so much better that it was selected as the sequencing rule also for the generalised case, where only a part of the picture is considered at each moment.

2. The structure of the sequencer

In a FORTRAN environment, for example, an applications programmer can use a standard plotter subroutine, with which it is possible to raise or lower the pen and/or move it to a new location (*cf.* DEC-10 routine PLOT). These operations are then performed exactly in the same order as the subroutine is called. (Consecutive subroutine calls are hence used either to draw a line (pen down) or to move the pen to the beginning of a new line (pen up).)

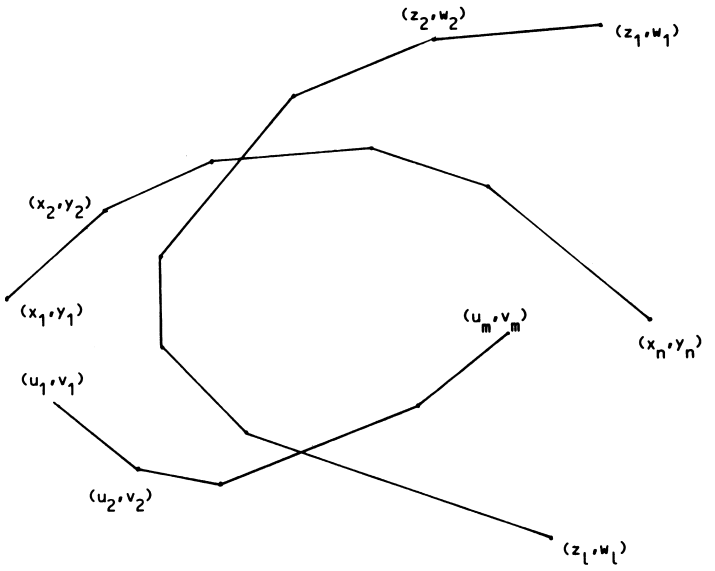

Fig. 1 A picture consisting of three lines:
line 1: $(x_1, y_1)\ (x_2, y_2) \ldots (x_n, y_n)$
line 2: $(u_1, v_1)\ (u_2, v_2) \ldots (u_m, v_m)$
line 3: $(z_1, w_1)\ (z_2, w_2) \ldots (z_l, w_l)$.

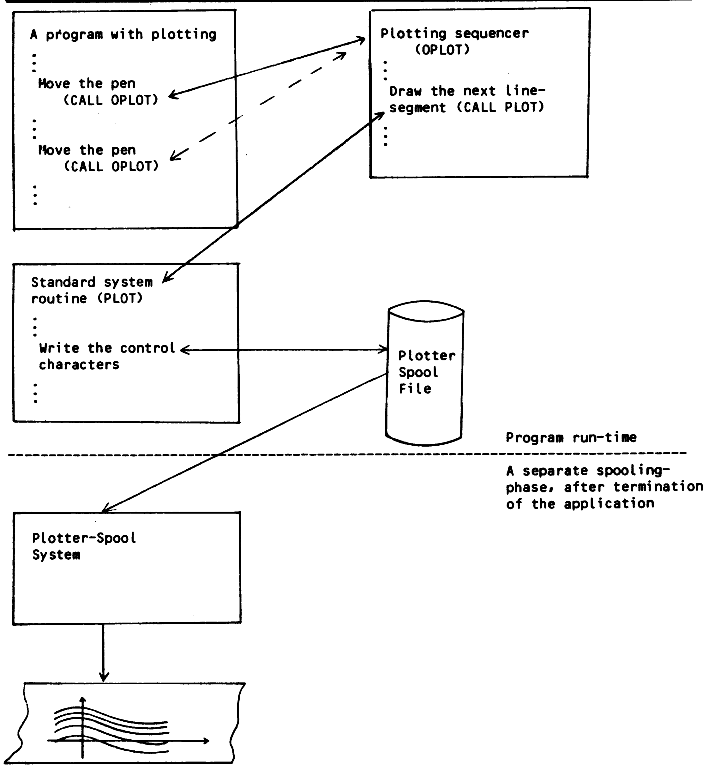
A program with plotting
Move the pen (CALL OPLOT)
Move the pen (CALL OPLOT)
Plotting sequencer (OPLOT)
Draw the next line-segment (CALL PLOT)
Standard system routine (PLOT)
Write the control characters
Plotter Spool File
Program run-time
A separate spooling-phase, after termination of the application
Plotter-Spool System

Fig. 2 The principle of sequenced plotting

The sequencing is accomplished by making a revised version, called OPLOT, of the above subroutine. OPLOT optimises the plotting sequence locally and its use is 'invisible' to the programmer, who can simply replace the old subroutine calls (PLOT) by equivalent sequencer calls (OPLOT). The task of the sequencer is to act as a buffer between the application and the plotter-spool system (see **Fig. 2**).

The locally best line that is plotted next is chosen by the *nearest neighbour rule, looking ahead N lines*:

Let S be a set consisting of N lines. Select from among the 2N end-points of the lines the one nearest to the present location of the pen. Draw the line, update the location of the pen and delete the line from S. Bring a new line into S and repeat the procedure until the picture is complete.

The parameter N is called the *degree* of the heuristic.

In the case of a drum plotter like CALCOMP 565 there are two independent power drives, one for the pen and the other for the drum. The drives can cause simultaneous movements in both x- and y-directions. Thus we measure the distances between points (x_1, y_1) and (x_2, y_2) by the L_∞ metric

$$d\{(x_1, y_1), (x_2, y_2)\} = \max\{\,|x_1 - x_2|, |y_1 - y_2|\,\} .$$

A simplified flowchart for the sequencer is shown in **Fig. 3.** Two different stages can be distinguished in it: (*a*) the construction and management of lines and (*b*) the selection of the next line and its 'drawing', by calling the standard non-sequenced plotter routine (PLOT). The operation of the sequencer is connected with the control of the upper/lower level of the pen: raising the pen terminates a line and lowering the pen triggers the selection of the nearest line and its drawing. A number of special cases, like the handling of lines with a large number of base points and the termination of the drawing task, are not shown in Fig. 3. (The lines have a variable number of base points. To simplify the management of the working storage, a fixed-length data segment is reserved for each line. If, however, the data segment is too short for storing the base points of a particular line, the line is divided into two or more parts.)

3. Effect of the sequencer

The performance of the sequencer can be evaluated in the following ways.

Mathematical modelling

A mathematical model is built up to describe the drawing tasks. This model in its turn may be *statistical* (e.g. the distribution of the end points of the lines in the plane are given) or *deterministic* (e.g. the locations of the end points are given). The effect of the sequencer is then tested by this model. Two factors must be analysed: the *total length of the wasted movements* as a function of the parameters of the sequencer and the *processing time of the sequencing* algorithm. The first gives the gain in drawing time and the second gives the increase in processing costs.

The advantage of mathematical modelling is that general results can be achieved, in which the effect of certain parameters (e.g. the degree of the heuristic) can sometimes be stated clearly. However, this approach involves considerable difficulties, especially in the construction of a realistic model and in the estimation of the wasted movements. In addition, when searching for the optimum processing we must take into account the properties of the particular operating system and the workload of the computer. This difficulty is present also in experimental testing.

In the case where the end-points of the lines are independently and uniformly distributed over the whole drawing area, we can achieve analytical results (see Kuokkanen, Leipälä and Nevalainen, 1977).

Experiments

Real or simulated drawing tasks are used and the CPU time and peripheral device times are measured. The experiments are easy to perform but general conclusions cannot easily be drawn.

3.1 *Experiments with random pictures*

A set of experiments was performed with artificial pictures. Each picture consisted of random line segments. More exactly, the end-points of the line segments were uniformly and independently distributed in a square of 5 × 5 inches in area. The aim was to determine the effect of the degree of the heuristic (N) on the plotting and CPU time for a DEC-10 with a CALCOMP 565 plotter.

Fig. 4 shows the total plotting time (T_p) as a function of N. Five different sets of 100 random lines were generated. The points in Fig. 4 are the averages of the observed plotting times. The same sets of lines were used for each given N-value. It will be observed that for $N \leqslant 15$ an increase of N causes considerable reduction of the plotting time. The effect of sequencing is negligible for larger N-values.

For a larger number of lines or a larger square, Fig. 4 can be used by multiplying the T_p-axis by a scaling factor. As can be seen from Fig. 2, the CPU time is needed in the processing of three different algorithms: the program that generates the line segments, the plotting sequencer (OPLOT) and the standard plotter routine (PLOT) that writes the control characters for the plotter. In **Fig. 5** the components of the CPU time are shown as a function of N for 100 and 200 line segments. If we subtract from the total CPU time the time used for PLOT-calls we get a linear dependence on N. At first sight the effect of the sequencer on the processing time is surprising; there is a decrease in the time observed for small values of N. This is because the sequencer reduces the total number of control characters formed by PLOT.

The above results are very similar to those obtained by the mathematical model.

3.2 *Experiments with non-random pictures*

A set of experiments was performed by using the sequencer with the hidden line plotting systems of Watkins (1974) and Williamson (1972). With the algorithm of Watkins we used three different test surfaces. Although the successive contours were plotted in reverse order without our system, we could still reduce the total plotting time by eliminating the movements of the pen along the contours of the invisible parts of the pictures. The saving in the plotting time varied between 11% and 30%. The effect of the sequencing was observed also with a plotter spool file of reduced size. The reduction varied between 10% and 39%. Still better results were achieved when using the algorithm of Williamson. For three test surfaces the maximal saving was 44% in plotting time and 45% in file size.

The plotting time observed includes some idle periods in the plotting. This is due to interruptions in the output process caused by the operating system. The workload of the system is thought to be about the same in the different tests, all pictures being plotted successively. We therefore suppose that the times are comparable. (The results of Section 3.1 were run during a low workload period of the computer so that the operation of the plotter was uninterrupted.) The reduction of the plotter spool file is a more accurate measure and the same trend can be seen in time and space savings.

The effect of the degree of the heuristic was also tested with the algorithm of Watkins, see (Kuokkanen, *et al.*, 1977a). As in the case of random pictures, the selection $N \sim 10$ seemed to give good results. The fastest plotting was observed when the lines of the picture used 1-2 data segments each. If the length of the data segments is increased at the expense of their number, the resulting low degree of the heuristic causes poor results. It is not advantageous to select a data segment such that many of the lines must be split into two parts, because the two parts are most probably plotted successively anyway. In the extreme case each data segment consists of only two points and as much as $2n$-2 points of the data area are needed to store a line of n points.

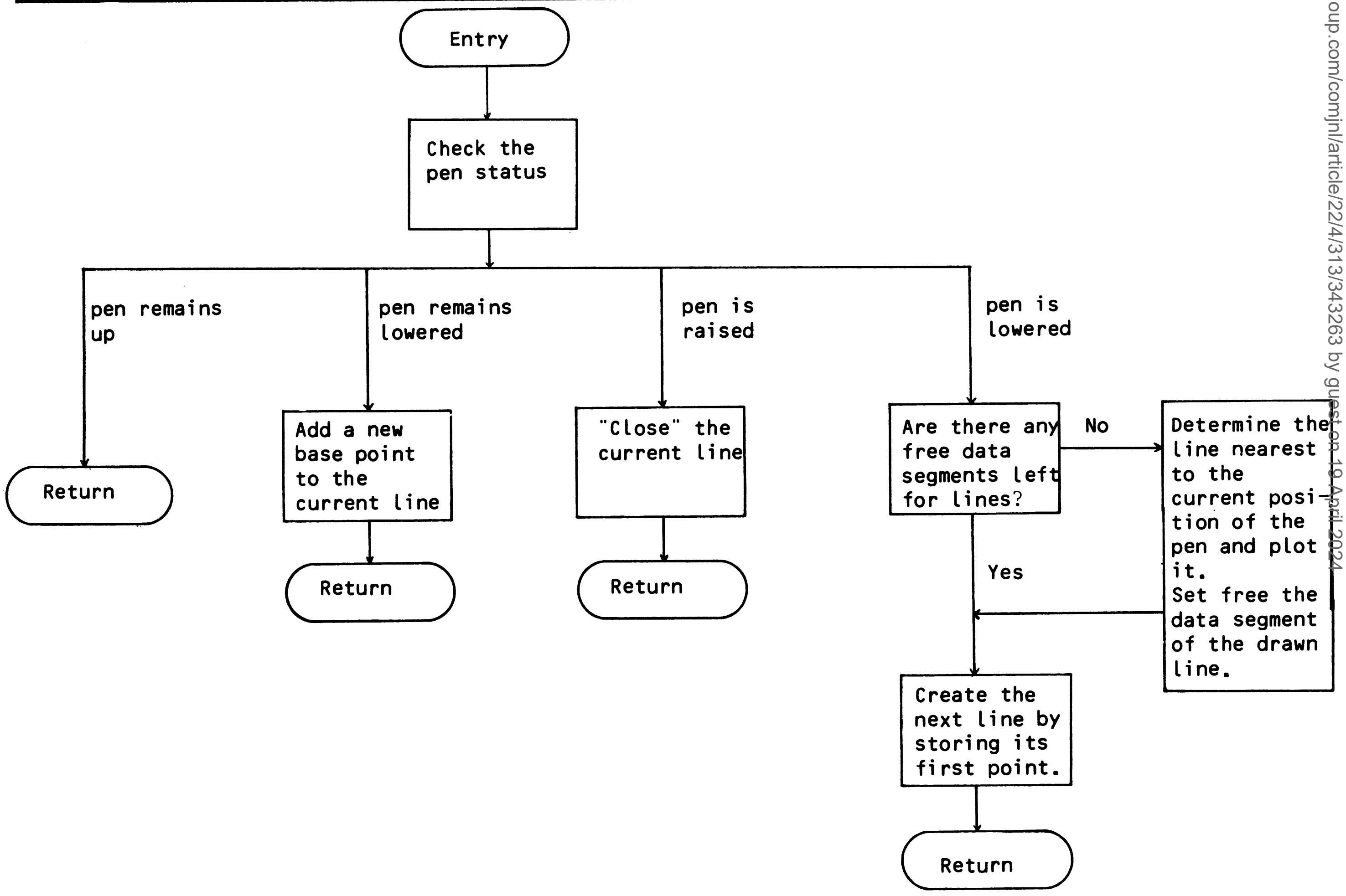
Entry
Check the pen status
pen remains up
pen remains lowered
pen is raised
pen is lowered
Return
Add a new base point to the current line
Return
"Close" the current line
Return
Are there any free data segments left for lines?
No
Determine the line nearest to the current position of the pen and plot it. Set free the data segment of the drawn line.
Yes
Create the next line by storing its first point.
Return

Fig. 3 **Flowchart for the plotting sequencer. CALL OPLOT (IC, X, Y), where (X, Y) is the new point and IC gives the control of the pen: IC = 0/1/2/−1 means 'state unchanged'/'raise'/'lower'/'last sequencer call'**

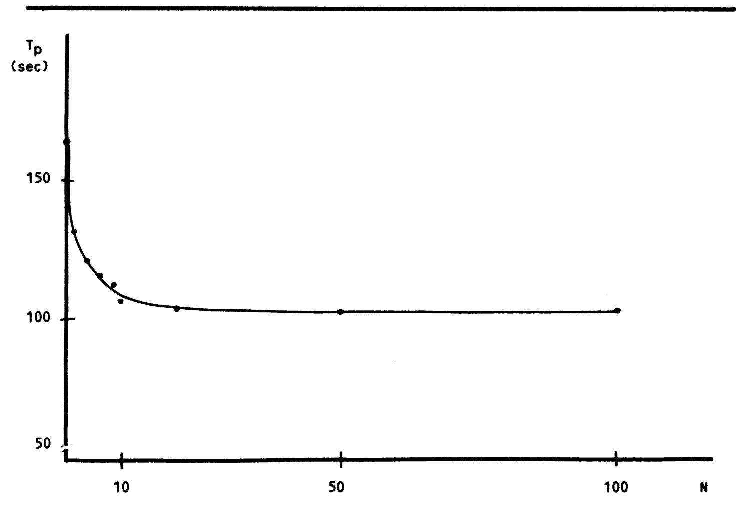

Fig. 4. **The total plotting time (T_p) as a function of the degree of the heuristic (N), when 100 random lines are drawn in square 5 × 5 inches in area**

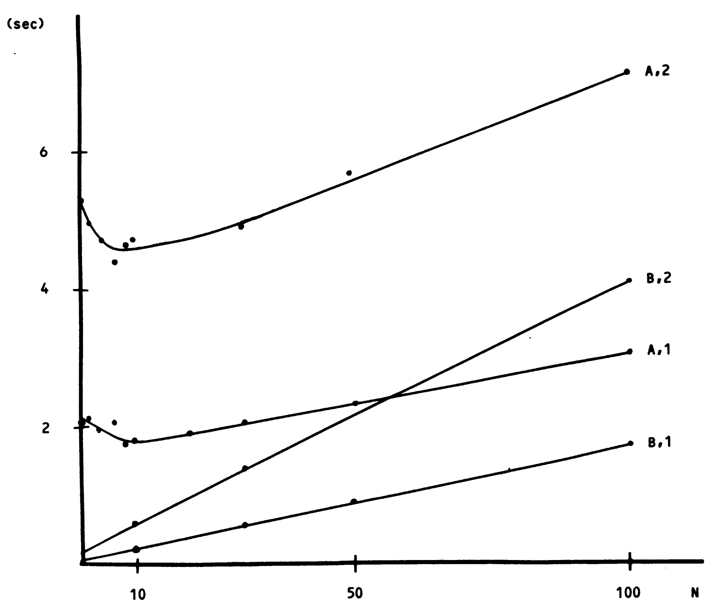

Fig. 5 **CPU time as a function of the degree of the heuristic. Legend:** ***A:*** **total observed time,** ***B:*** **time without PLOT-calls, 1: 100 random lines, 2: 200 random lines**

4. On the performance of the nearest neighbour rule

Let us suppose that the end-points of the K line segments are uniformly distributed in a picture of area R. Leipälä (1978) has shown that for the nearest neighbour rule the expected length of the wasted movement is

$$\bar{s}(N, K) \sim \sqrt{R/2}(K/\sqrt{N} + \sqrt{N})/2 \qquad (1)$$

when the degree of the heuristic is N.

As an average lower bound of the wasted movements we can use $s_1(K) = K\bar{d}_{\min}(K)$, where $\bar{d}_{\min}(K)$ is the expected minimum distance between the end-point of one line and all end-points of the other K-1 lines. Then we can obtain that $\bar{s}_1(K) \sim \bar{s}(K, K)/2$. Thus, in the best case, where $N = K$, the average wasted movement is about twice the average lower bound. On the other hand, the ratio of $s(N, K)$ to the optimal solution is not bounded. It has been shown by Rosenkrantz, *et al.* (1977) that for K-city travelling salesman problems the nearest neighbour rule guarantees a solution that is at most $0{\cdot}5 \cdot \log_2 K$ times the optimum, if the distances satisfy the triangle inequality. In the case of the plotter sequencing problem the triangle inequality is not satisfied between the lines.

An important observation can be made if $\bar{s}(N, K)$ is compared with the non-sequenced case. Then the total length of the wasted movement is on the average $\bar{s}_r(K) = \bar{b}K$, where $\bar{b}$ is the mean distance between the end-points of the lines. The length of the wasted movement thus increases linearly with K. Formula (1) shows that also $\bar{s}(N, K)$ increases linearly with K but at a much slower rate. In the best case where $N = K$, we obtain a square root increase rate $\bar{s}(K, K) = (R/2)^{0\cdot 5}K^{0\cdot 5}$. The execution time is then $O(K^2)$, whereas for a fixed N we have an 0(1) algorithm.

One way of improving the sequencing might be to modify the 'large-arc' algorithm of Frederickson *et al.* (1978) so that it works also for undirected arcs. This algorithm guarantees a solution that is at most three times the optimum, but its processing time is of the order $O(K^3)$ and it cannot be adapted to dynamic situations. In static situations, where we can store the whole picture simultaneously, it could also be possible to modify the $O(K^2)$ travelling salesman algorithms of Rosenkrantz, *et al.* (1977) to produce better plotting sequences.

5. Summary

The paper described a stepwise plotter sequencer for reducing the time needed for the wasted movements made by the pen. Only a part of the picture was considered at each moment. This made possible the processing of pictures of unlimited complexity. The effect of the sequencer was experimentally tested in the cases of both random and non-random pictures. The sequencer was capable of reducing both the actual plotting time (Fig. 4) and the CPU time (Fig. 5), if the degree of the heuristic, i.e. the number of lines to be considered at a time, is about 10-15.

The sequencer is available as a FORTRAN subroutine in the DECUS-Library of programs (Kuokkanen and Nevalainen, 1977b).

References

Bresenham, J. E. (1965). Algorithm for computer control of a digital plotter, *IBM Syst. J.*, Vol. 4, pp. 25-30.

Frederickson, G. N., Hecht, M. S. and Kim, C. E. (1978). Approximation Algorithms for Some Routing Problems, *SIAM J. Comput.*, Vol. 7, pp. 178-193.

Freeman, H. (1969). A Review of Relevant Problems in the Processing of Line-Drawing Data, in *Automatic Interpretation and Classification of Images*, ed. by A. Grasselli, Academic Press, New York, pp. 155-174.

Kuokkanen, L., Leipälä, T. and Nevalainen, O. (1977a). Implementation and analysis of a plotter sequencing system, Report B9, Department of Computer Science, University of Turku, Turku, Finland.

Kuokkanen, L. and Nevalainen, O. (1977b). Revised Plotter Subroutines for DEC-10, DECUS Program Library No. 10-292.

Leipälä, T. (1978). The probabilistic analysis of the nearest neighbour algorithm in plotter sequencing and stacker-crane problems, Report B16, Department of Computer Science, University of Turku, Turku, Finland.

Liesegang, D. G. (1976). The Tube-Passing Problem and the Travelling Salesman Problem, to be published by North-Holland in *Proceedings of the IX International Symposium on Mathematical Programming.*

Nevalainen, O and Vesterinen, M. (1978). Solving the Order of Line Drawing, *Angew. Inf.*, 1/78, pp. 15-19.

Reingold, E. M., Nievergelt, J. and Deo, N. (1977). *Combinatorial Algorithms: Theory and Practice*, Prentice-Hall, Englewood Cliffs, N.J.

Rosenkrantz, D. J., Stearns, R. E. and Lewis II, P. M. (1977). Approximate Algorithms for the Traveling Salesman Problem, *SIAM J. Comput.*, Vol. 6, pp. 563-581.

Watkins, S. L. (1974). Masked Three-Dimensional Plot Program with Rotations, *CACM*, Vol. 17, pp. 520-523.

Williamson, H. (1972). Hidden-Line Plotting Program, *CACM*, Vol. 15, pp. 100-103.