NOTICE
This report was prepared as an account of work sponsored by the United States Government. Neither the United States nor the United States Energy Research and Development Administration, nor any of their employees, nor any of their contractors, subcontractors, or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness or usefulness of any information, apparatus, product or process disclosed, or represents that its use would not infringe privately owned rights.

UILLU-ENG 75 5502
COO-1195-344
J. Tortora
May 29, 1975

Technical Report

MATCH: Improving the efficiency of the decision lattice scanner

A serious problem has been found with the decision lattice scanner (sub-routine PSCAN) used by MATCH. Before presenting the problems and their solutions we will begin with a section on the method used by PSCAN to find solutions.^{1/}

The paper is therefore divided into three parts:

- A) How PSCAN works
- B) The problems
- C) The solution.

A) Backtracking

PSCAN scans a given two dimensional decision lattice for solutions of length L.

Each two dimensional lattice corresponds to a pair of views. Each point within each lattice corresponds to a pair of tracks, one from each view. A point is said to be "on" if that pair is a possible match for this event. A solution of length L is a set of L points all of which are on and no two of which are in the same row or column. The problem of finding a solution of length L is thus isomorphic to the problem of putting L rooks on a chessboard with some squares cut out, such that no rook can attack any other.

The technique for finding solutions is called backtracking and the method is flow charted below.

MASTER

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency Thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

Set all points untried.
Set all points not part of trial solution.
Set all rows not skipped.
ISOL=0. ROW=0.
LL =0

Parameters for running loop,
MROWS, MCOLS, L.
Retruns ISOL= number of solutions.

As an example, suppose we wish to find all the solutions of length 3 in the following 3 by 3 lattice. For convenience let us define an available point as one which is on and which is not in any row or column which is already a part of a trial solution, and which has not been tried since the last back-up past this row. A point which is on will be indicated by a heavy dot. Parts of a solution will be circled. Points tried will have a slash through them.

We take the first available point in row 1.

We take the next available point in row 2.

There are no available points in row 3 so we back up one and take the next available point.

Then we take the next available point in row 3. One solution has now been found. It is the pattern represented at the left.

To find the next solution we see that there are no more points available in row 3 so we back up to row 2.

In row 2 there are no points available so we back up to row 1.

In row 1 we take the other available point.

Then we take the next available on in row 2.

There are no available points in row 3 so we back-up to row 2 and try the other available point.

Then we take the only available one in row 3. We have now found a second solution.

There are no more in row 3 so we back-up to row 2.

There are no more in row 2 so we back up to 1.

There are no more in 1 so we are done, there are no more solutions.

B) The Problems

The backtracking method outlined in Section A will always work and will always find all the solutions. The problem is the amount of time required to find the solutions.

Consider the lattice shown in figure 1a. This lattice is one which MATCH actually encountered on a 14 prong event. The lattice caused PSCAN to absorb more than an hour of CPU time without finding even one of the 3.2×10^6 ($9 \times 10!$) solutions.

Let us follow the backtrack logic and see what happens. We will move along picking the first available point in each row until we get to row 11. (See figure 1b.) At this point there are no available points in row 11 so the algorithm requires that we back-up to row 10 and try again. (figure 1c) We still cannot find a point in row 11 so we back-up to row 10 again and take the last remaining point. (figure 1d) We still have not found a solution and there are no points left in row 10 so we back-up to row 9 and take the next available point.

This will continue until we have reached row 1 and try a different point there. The algorithm will then march forward to row 11 and be forced to backtrack all the way to row 2. Then forward to row 11 and back to row 3, etc. The total number of tries necessary to find the first solution is $1 + 1/2 \sum_{m=4}^{11} m!$ or 2.2×10^7 . If we figure roughly one millisecond per iteration it will take about 6 hours to find the first solution.

There is another related problem which is that it may also take a large amount of time to discover that a given lattice has no solution. Imagine for example that in the lattice of figure 1a all of the points in row 1 are off except for column 1. Then it will take $11!/2$ iterations or 5-1/2 hours to discover that there is no solution.

fig. 1a

fig 1b

fig 1c

fig 1d

fig 1e

fig 1f
SINGLE ORDERING

C) The Solution

The real problem with the present algorithm is that it cannot efficiently handle rows with small numbers of points which are deep into the lattice. One solution is to arrange the rows in increasing numbers of points before trying to find the solutions. Let us call this row ordering.

With this method the first solution would be found on the very first try in the example just given (See fig. 1f). The saving in time is a factor of 22,000,000 to 1. This method should always be better since it maximizes the probability that the right guesses are made at the beginning. This observation has previously been made by Golomb et al.^{1/}

An even better solution is to order all the rows as above and then to order all the columns. Figure 2a shows a row ordered lattice in which it would take 6! tries to find the first solution. With both row and column ordering (fig. 2b) it would take only one try.

There are still lattices which are both row and column ordered in which the solution is not found immediately. For the lattice shown in fig. 2c it would take 48 tries to find the first solution.

When searching for just the first solution a further saving in time can be achieved by modifying the algorithm to recognize identical rows. Obviously in searching for the first solution it does no good to backtrack to an identical row. Once a solution has been found, however, additional solutions may be generated from identical rows.

Since in MATCH rows are stored as words and columns as bits we have only implemented row ordering. At present we also do not check for identity of rows. Before implementing this change we used a cut off of 100,000 iterations of the loop in PSCAN. About half of the 14 prongs hit this limit. On a sample of data in which 15% of the events were between 10 and 14 prongs, the average time per event spent

in PSCAN was 4.2 seconds. After the above modification this time was reduced to .5 seconds per event. If it is assumed that all of this saving comes from the 10, 12, and 14 prong events, then the modified algorithm runs about 55 times faster on these events.

It should be noted for posterity that there is an algorithm due to A. Snyder in which rows or columns are interchanged until there are L points on the diagonal. The method has not been looked into extensively but it appears to be no better than the present one.

Many thanks are due to R. Barth, R. T. Cutler, R. Plumer, and A. Snyder for many helpful comments.

DOUBLE ORDERING

¹For an explanation of the theory of decision lattices and backtracking see the following two papers.

Samuel J. Penny and James H. Burkhard, Multidimensional Correlation Lattices As An Aid To Three-Dimensional Pattern Reconstruction, UCRL-16478, Proceedings, Spring Joint Computer Conference, Boston, 1966, pp. 449-455.

Solomon W. Golomb and Leonard D. Baumert, Backtrack Programming, Journal of the Association for Computing Machinery, Vol. 12, No. 4 (October, 1965), pp. 516-524.