INFORMATION TECHNOLOGY

41 Automation of the Crozzle
   GH HARRIS, JJH FORSTER

49 Formal Computing Education in Australia: Do We Practise What We Teach?
   S ELLIOT

61 A Randomised Schema Mutator for Evolutionary Database Optimisation
   P van BOMMEL

70 Realisable Memory Management Algorithms
   MR HANNAFORD, DWE BLATT

SPECIAL FEATURES

82 Books Received

82 Reviews Published Electronically

83 Book Reviews

Published for Australian Computer Society Incorporated
Registered by Australia Post, Publication No. NBG 1124
Automation of the Crozzle

GH Harris and JJH Forster
Division of CAD,
Griffith University,
Nathan, Queensland 4111, Australia

The Crozzle, a complex crossword variant, is described. The Crozzle consists of entering words from a given lexicon into an empty grid such that a total score based upon, (i) the individual scores for the letters at which words intersect and (ii) the total number of words entered, is maximised. This process also determines the number and location of black squares in the grid. An algorithm, using depth first recursion and a 2-level look-ahead, to generate solutions to the unconstrained crossword puzzle problem is modified and applied to the Crozzle. The solution scores generated are compared to the score of the solutions produced by humans. Three data sets are presented as benchmarks for future research efforts.

Keywords: Crozzle, tree traversal, lookahead algorithms and pruning search spaces.
Short Title: Crozzle Automation.

1 INTRODUCTION

The crossword puzzle word game is well known. Some very successful attempts to automate the process of generating the entire solution set to a given puzzle grid and lexicon have been published (Smith and Steen, 1981; Berghel, 1987; Harris, Spring and Forster, 1991). The standard crossword computational problem is, given a puzzle grid of black and white squares, and a lexicon (typically large), generate all the solutions to that puzzle grid that involve words from and only from the supplied lexicon. Harris (Harris, 1990) has identified a problem that is closely related, the generation of unconstrained crossword puzzles: given only a lexicon and the external dimensions of the puzzle grid, generate puzzle solutions. That is, the internal puzzle geometry, defined as the number and location of black and white squares consistent with the external grid dimensions, is also to be determined by the algorithm.

An attempt to automate another crossword variant, the Crozzle word game is now reported. This game, in essence, involves the production of an unconstrained crossword puzzle from a supplied lexicon and external puzzle grid dimensions but, in the crozzle, each solution is then “scored” according to a set of rules. The highest scoring entry submitted is then judged to be the winning solution. The computational problem is then no longer one of finding a solution but traversing the search space so as to find the highest scoring solution. There is no element of chance in the competition, which is offered monthly in each issue of the Australian Women’s Weekly.

For this particular puzzle the grid dimensions are 10 rows and 15 columns and each month’s lexicon is typically no more than 140 words. This size of puzzle and lexicon produces an enormous solution set over which to search for the maximum score.

2 THE CROZZLE

The published rules of the competition are reproduced in Appendix 1. Each month a different theme, such as names of constellations, Australian explorers and so on, is used in choosing the words in the lexicon. The object of the game is to create a wholly interconnected “standard” crossword puzzle solution, lying entirely within the grid, using the words supplied, with no word being used more than once.

Any legal puzzle solution is then scored according to the number of words fitted into the puzzle, and the numerical value of those letters appearing at the intersection of words. Each word fitted into the final puzzle scores 10 points and, as shown in Table 1, each letter of the alphabet has a fixed numerical value, ranging from 2 to 64, scored if and only if the letter is at a word intersection square. In Appendix 2 are shown three typical Crozzle lexicons, and the corresponding (human-compiled) winning entries. The winning entry is the legal puzzle solution, correctly submitted within 1 calendar month, which scores highest. In the event of a tie, there is count-back involving the number of words of longest length used. Nevertheless it is quite common for two or three individuals to share the prize, having identical scores derived from
identical, or nearly identical, entries. In one case, (May, 1990), human players must have found the puzzle particularly "simple" as there were 39 identical winning entries. The reason for simplicity from the human's point of view appears to be that two intersecting z's appear in the winning solution with a resulting score of 734 points. However, in the time limit of 1 hour available to it (specified later in the paper), the current implementation did not reach the part of the tree which included the two intersecting z's and scored only 560 points, 69.7% of the winning human score.

Table 1. Points scored for intersecting letters.

<table>
<thead>
<tr>
<th>Letter</th>
<th>Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>a,b,c,d,e,f</td>
<td>2</td>
</tr>
<tr>
<td>g,h,i,j,k,l</td>
<td>4</td>
</tr>
<tr>
<td>m,n,o,p,q,r</td>
<td>8</td>
</tr>
<tr>
<td>s,t,u,v,w,x</td>
<td>16</td>
</tr>
<tr>
<td>y</td>
<td>32</td>
</tr>
<tr>
<td>z</td>
<td>64</td>
</tr>
</tbody>
</table>

3 THE CHALLENGE FOR AUTOMATION

At present the challenge for scientists is to beat the highest score of any of the human competitors within the time limit specified by the Crozzle conditions: a calendar month. There are three aspects of the Crozzle that pose problems for automation:

1. the one month time limit,
2. the size of the search space, and,
3. the apparent efficiency of the human "experts" (entrants).

It is problem (2) that poses the greatest challenge. Ultimately the aim is to achieve the global maximum score and, quite obviously, complete enumeration would produce the highest possible score. However, the search space size is too large to allow this approach. For example, just to enumerate all the possible word slot geometries and try to solve their constrained solutions (see Harris (1990) for a discussion of this) would take over $10^{31}$ years on a machine that could generate and produce the solution sets to 1,000,000 geometries per second. Further, there are over $10^{20}$ combinations of 20 or more words chosen from the roughly 100 available (typically, winning solutions contain between 20 and 30 words). Allowing for over 1,000 combinations examined per second, which is more than three times faster than the current implementation, and assuming heuristics reduce the search space by eight orders of magnitude, it would still require over 30 million centuries to traverse the remaining portions of the tree. Each of these would usually yield over 100,000,000 possible solutions to be scored, resulting in an estimated $10^{25}$ solutions to be generated. This is a very large search space to traverse and, consequently, any efficient generator of solutions must eliminate not only sub-trees that contain no valid solutions, but also all those sub-trees that can be guaranteed not to contain the global maximum score.

It is the lack of fixed word slot locations, unlike the standard, i.e., constrained, crossword puzzle where the position of word slots is determined by the position of the black squares, that makes it impossible to consider using the constrained puzzle solution generators that have been described in the literature to date (see, e.g., Berghel, 1987; Harris, Spring and Forster, 1991). To illustrate, consider the constrained puzzle grid consisting of the positions of the black and white squares for the human compiled solution to the first lexicon in Appendix 2: an efficient solution generator, using the supplied lexicon of 100 words, generated and printed all possible solutions (4) in just 0.66 seconds. However, if the position and number of word slots is unspecified, even using a lexicon consisting of only the 23 words from the winning solution shown in Appendix 1 it took the current crozzle algorithm implementation over 20 hours to arrive at the winning solution. (After the 20 hours, the algorithm had generated over 30 million non-trivial solutions using an average of 19 words from the 23 supplied.)

4 THE ALGORITHM

The crozzle producing algorithm employed for the results in this paper is a modification of the unconstrained algorithm reported in Harris (Harris, 1990). The principles and implementation of this algorithm are briefly described.

The algorithm utilises a letter slot table (as opposed to the word slot table used in the constrained problem algorithms). Essentially the letter slot table contains the order in which each letter is inserted into the puzzle grid. However there are major differences in this slot table to that of the constrained problem. This slot table:

(i) contains the co-ordinates of each letter in the grid, and a horizontal/vertical flag describing the orientation of any words that may intersect with that letter,
(ii) is determined during run-time and hence is dynamic, as opposed to the static slot tables typically employed in the constrained problem (see, e.g., Smith and Steen, 1981; Berghel, 1987; and Harris et al, 1992), and,
(iii) does not have the condition that every slot be filled (i.e., that all letters intersect).

4.1 An Illustrative Example

Before concluding the description of the general algorithm, a cut-down Crozzle example is used to illustrate some of the principles of the construction and use of the above slot table:

AC AMC BL BMW BSA FIAT GM JAG MG RR VW

This theme lexicon (famous marques of cars and motorbikes) illustrates several points. First, what is considered a word is arbitrary so that for our purposes the initials GM (General Motors) are as much a word as FIAT. Second, consider words such as RR (Rolls Royce): no other word contains an R, which means that RR could only appear in a solution on its own as it would fail to link with any other word. However this would break the Crozzle interconnectedness rule number 4 (Appendix 1). Consequently RR can be eliminated from consideration, although RR could appear on its own as a valid solution. Third, illustrating some of the problems for computation, the RR "on its own" solution can appear...
in 22 different geometries in a 3 x 5 grid, each involving RR in a different position, although always with 13 black squares and only scoring 10 points (1 word used, no letter intersections). Fourth, the word FIAT can appear in a valid solution but only in a horizontal position. This is because of the vertical dimensions of the puzzle grid being less than four, the number of letters in FIAT.

Initially, we have no words in the grid and thus no entries in the (letter) slot table. Hence, a word is required to initialise the slot table, say the third word from the dictionary:

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>B</td>
<td>M</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The numbers around the word BMW represent rows and columns of the “work-grid” within which the puzzle grid will eventually be located. It is the words and their positions in a solution which determine the puzzle grid location.

As the word BMW is inserted, the slot table is initialized as shown in the table below:

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>4</td>
<td>Vertical</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>Vertical</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>Vertical</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The numbers above refer to locations within the work-grid and the flags (Vertical) specify that any intersections at those locations must come from vertically inserted words. Also included is an asterisk (*) pointing to the slot table entry to be filled next.

Now, using the first entry from this slot table, the word BSA intersects vertically with the letter B at (5,4):

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>B</td>
<td>M</td>
<td>W</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

and the slot table will now be:

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>4</td>
<td>Vertical</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>Vertical</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>Vertical</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>Horizontal</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>4</td>
<td>Horizontal</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note that the letter B at (5,4) maintains its original flag (Vertical) as the letters S (6,4) and A (7,4) are the only insertions from the word BSA. Also note that prior to using BSA the algorithm will have tried and found all those solutions possible using BL to intersect, if it were working sequentially through the lexicon.

The algorithm then progresses to the second entry, (5,5, Vertical), in the slot table. Although there are 2 remaining words available from the dictionary containing the letter M, neither will insert into the grid without causing an entry to the slot table that would be impossible to fill when it was reached. For example, if the algorithm was to insert the word MG as below:

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>B</td>
<td>M</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>S</td>
<td>G</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

the slot table would then be

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>4</td>
<td>Vertical</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>Vertical</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>Vertical</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>Horizontal</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>4</td>
<td>Horizontal</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>Horizontal</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

This would mean the entries (6,4) and (6,5) in the slot table will result in solution failure. Thus, the current algorithm performs a “look-ahead” in the slot table so as to ensure that such potential solution failures are not encountered. For an example of the use of “look-ahead” in automated crossword solving see Berghel and Yi (1989, pp278-279). That is, the above slot table would not occur as the algorithm would have determined the conflict and hence MG will not be inserted into the grid.

Similarly, an attempt could be made to insert GM at (4,5) as shown below.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>G</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>B</td>
<td>M</td>
<td>W</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>S</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

and the slot table would then become
But clearly this will also result in failure as the algorithm will check that the dimensions of 3 x 5 external grid would be exceeded. Clearly no allowable intersection at (5,5) would be possible.

The algorithm would then progress to the third entry in the slot table, (5,6,Vertical). Again no successful intersection is possible as VW would fail the same constraint. In due course, utilising the same procedures as above, as well as backtracking, the algorithm will arrive at the following structure:

```
1 2 3 4 5 6 7 8
1
2
3
4
5 B M W
6 S
7 A C
8
9
```

and the slot table will be

```
1 2 3 4 5 6 7 8
1
2
3
4
5   B   M   W
6   S
7   A   C
8
9
```

At this juncture no more words can be inserted without transgressing the rules of either crosswords in general or the Crozzle in particular so that this will be a solution. It is important to note

(a) this is not necessarily the optimum scoring solution but it is a valid solution because no more words can be inserted, and,
(b) none of the intermediate stages involving either BMW alone or BMW plus BSA are complete solutions because further words (namely AC, FIAT and JAG) could be inserted. Of course, intermediate stages even if considered as solutions must always score lower than terminal solutions.

Indeed, it is straightforward though messy to verify (although it only took the current implementation 0.06 seconds to determine) that the highest possible, and thus winning, score is 78 points for the following grid:

```
1 2 3 4 5 6 7 8
1
2
3
4 B S A
5 M M G
6 V W C
7
8
9
```

This solution is scored as follows: (5 words x 10 points per word) + (W=16) + (A=2) + (B=2) + (M=8) = 78 points. This is as opposed to the (3 words x 10 points per word) + (A=2) + (B=2) = 34 points for the solution used in the preceding example.

To recap, the example has demonstrated not only most of the principles on which the algorithm operates, but also that identical solutions in the same solution set may have different slot tables. Having used an example to explain and illustrate the use of the letter slot table, the principle features of the general algorithm are now discussed.

4.2 Description of the Algorithm

The present algorithm uses recursion with backtracking to traverse the search space. In the constrained problem backtracking occurs upon failure to fill a word slot. However, for this problem, backtracking occurs upon failure to intersect a word from the lexicon with the current letter slot if the current letter slot forms part of an incomplete word.

Recursion is employed to move through the letter slot table, initialised with a word chosen, either at random or user defined, from the lexicon. At each entry in the slot table in turn, the algorithm attempts to add another word to the work-grid, intersecting with the letter already in the work-grid that is represented by that entry in the slot table. This intersection may already have been achieved as a result of considering an entry higher up in the slot table, in which case the algorithm progresses to the next position in the slot table. Otherwise, if a word is successfully inserted into the grid, the slot table is adjusted accordingly and the algorithm progresses to the next entry in the slot table. However, if no allowable word can be inserted into the grid and there are no letters on either side of the current letter being considered, then the algorithm will again progress on to the next entry in the slot table. For all other cases, a failure has occurred: that is, a sequence of letters has been encountered which is neither in the lexicon, nor contained within a word in the lexicon. The algorithm then backtracks up one position in the slot table. If a word has been intersected with this position, it is removed and a new word inserted, otherwise the algorithm continues the backtracking.

The essential features of this part of the algorithm are more readily seen and understood with the aid of the pseudo-code of the implementation:
The algorithm continues until normal termination, which occurs when the algorithm tries to backtrack above the first entry in the slot table.

5 IMPLEMENTATION AND HEURISTICS

The algorithm was encoded in Turbo Pascal on a 25 MHz IBM 386 Clone. As it is not anticipated that the current algorithm will be used on large lexicons, the data structure limitation of 64 Kbytes inherent in the compiler is not of consequence. The program requires an ASCII text file containing the (assumed small: less than 256 words) lexicon. The program also requires some runtime control parameters: timeouts, a desired size of the puzzle grid, as well as a minimum score required of a solution before printing out commences.

After reading in the lexicon, an index was set up describing the ith occurrence of the jth letter of the alphabet; for example, the second occurrence of the letter p is at the third letter of the tenth word in the supplied lexicon. A work-grid, as illustrated in the example above, was established consisting of a character matrix of dimensions at least twice the size of the maximum puzzle grid size desired. Arbitrarily, a work grid of 64 by 64 was chosen. The (dynamic) letter slot table was implemented as a data structure consisting of the array indices describing the position of letters within the work-grid, as well as a flag indicating whether intersections at any one letter are to be horizontal (1) or vertical (2). The final major data structure used was a list showing the "block" entries of the letters of a word as they were inserted into the work-grid, and hence, the slot table. Consider the example given earlier: notice how the slot table grows in "lumps" or blocks. These blocks need to be identified to enable the removal of words from the work-grid, but without destroying the puzzle solution structure built up to that point.

The implementation of the program's logical controls required to perform basic data management functions on the entries in the work grid is not a trivial task. The insertion of a new word may make it impossible to satisfy the intersection criteria of a slot higher up in the letter slot table. This is readily seen in the example with the first use of the word MG. Although the recursion and backtracking is bounded, and hence will detect and correct this problem, a "forward-check" was also added to the program in order to eliminate any unnecessary processing time.

Finally, as the search space is so vast, timeouts are required. Two timeouts were introduced into the program to enable a wider sampling of the search space and, hence, the solution set. Both timeouts are user supplied. One sets the time spent looking for solutions before the index is randomly sifted. Obviously, this sifting moves the search to a different position in the solution set. The other is a "global" setting on the overall length of time the program has to run. Even relatively small dictionaries can lead to excessively long run times to completely traverse the search space.

A final, though very important, heuristic was imple-
mented: rather than only adding letters to the slot table as they are inserted (as shown in the above example), the letters from any one word are added to the grid, and hence the slot table, according to their point score. For example, if the word “puzzle” were interlocking with the existing word “zoo”, then the letters would be added to the grid (and slot table) in the order of z,u,p,l,e. This simple heuristic resulted in an increase in scores by 10% per unit time. In fact, it was the addition of this heuristic, coupled with eliminating words over 7 characters in length, that produced the near 80% attainment of the human achieved scores, column 4 of Table 2.

5 PROPOSED CROZZLE BENCHMARK

As a guide to the effectiveness and efficiency of the implementation of future research into the Crozzle word game, the following benchmark tests are proposed. It is proposed that implementations be tested against the 3 lexicons supplied in Appendix 1. These were chosen from the many available for different reasons. The first one has a human-produced solution which is rather sparse (there are no large interlocking blocks in the solution). The second has two significant and interlocking blocks of letters. The third, “between” the other two, has one small interlocking block (2 by 2), but is otherwise sparse.

The benchmark tests are to run an implementation for less than 1 hour on each of the supplied lexicons retaining the highest score produced for each. This is to be compared with both the human-produced score and the score produced by the current implementation. To enable a valid comparison, the computer hardware used should be a 25 MHz 80386 based IBM/IBM clone machine. Table 2 below summarises the scores obtained by the winning human compiled solutions (column 2) and those obtained by the current algorithm (column 3). Clearly human players, at least when considered as a whole, are currently producing higher scores than the automatic Crozzle solver presented.

To allow a comparison of the best of human efforts with typical automated solutions the three winning human solutions are presented in Appendix 2 as figures A2.1, A2.3 and A2.5, with corresponding automated solutions in figures A2.2, A2.4 and A2.6. For the first pair of A2.1 and A2.2 the most striking feature is the block of the words LYONS, EYRE, MENZIES and JANTZ which appears in the same position and with the same interlocking structure in both solutions. Beyond this, however, there is no similarity apart from a lack of any interlocking blocks in the rest of the two solutions. For the second pair, A2.3 and A2.4, the solutions appear completely different, with the winning human solution having a large 3x4 interlocking block along with a 2x3 interlocking block, while the automated solution has two entirely dissimilar 2x2 interlocking blocks. Figures A2.5 and A2.6 demonstrate no apparent significant similarities, apart from the inclusion of BLITZEN and RAZOR intersecting at the letter Z.

6 CONCLUSION AND FUTURE RESEARCH

This paper has introduced the Crozzle word game and demonstrated some of the difficulties involved in the automated generation of the solution set. Due to the size of the solution sets, the implementation described above is not able, in a reasonable time limit, to be competitive with the solutions generated by human entrants. To generate competitive solutions by way of generating the global maximum is not a trivial problem. A sample of 3 Crozzle lexicons and their respective human generated winning solutions are presented as benchmark tests for any future research performed.

One highly significant aspect of this work that has not yet been explored is the set of methods used by human players. Clearly they do not have the enumerative abilities of the current algorithm but they produce, at least as a group, significantly higher scores than the algorithm. It is also completely unclear as to how often, if ever, the human players achieve the global maximum score. Nor is it known how many human players submit scores, to the judges, let alone the number that try but never submit. A series of fascinating opportunities for comparisons of human and automated players exist as it does for chess.

Quite obviously there is much further work to be achieved on this problem. Although complete enumeration is not a practical method of generating solutions, more work needs to be accomplished on the use of heuristics to trim the solution set, but which do not delete the global maximum solution.

REFERENCES

APPENDIX 1 CROZZLE CONTEST RULES
The rules are the same for each new data set. The following was taken from the February, 1990, issue of The Australian Women’s Weekly (page 244):
1. Use only the words given in our list opposite.
2. Words cannot be used more than once.
3. Don’t run single words together — make sure, as in normal crosswords, that you have at least one black square between words which are not joining or interlocking.
4. All words used must be in one interlinking block (as in the example on the facing page).
5. Letters standing alone have no value”.

APPENDIX 2 SAMPLE CROZZLE DATA SETS AND WINNING SOLUTIONS
The following input data set is for the January 1990 Crozzle competition. The corresponding winning solution is shown in Figure A2.1.

bass giles deakin tasman hargrave
boyd jantz dennis turner mitchell
cook jones dobell wattle paterson
dark kelly eureka chifley gallipoli
eyre lahor fadden dampier hargraves
fisk lewis florey griffin macarthur
gold melba hartog hilkert mackellar
holt nolan heysen menzies macquarie
hume oxley hovell phillip moncrieff
quay smith lawson scullin leichhardt
reid stand hunter blackett strzelecki
rudd burton mawson blaxland sutherland
bligh wills mawson drysdale
bruce barton murray
burke burnet parkes flinders
emden curtin sirius grainger
fleet dawson stuart greenway

Figure A2.1: The human winning solution to the January 1990 Crozzle competition scoring 612 points.

ayah artist singer milkman optician
chef author tailor painter publican
crew banker teller pianist reporter
drug barber typist plumber retailer
herd barman umpire postman salesman
maiden bishop waiter printer sculptor
monk brewer warden saddler surveyor
page broker welder servant unionist
vet butcher worker shearer waitress
actor carter writer soldier announcer
agent cowboy acrobat steward architect
baker critic actress surgeon barrister
boxer dancer artisan teacher carpenter
clerk doctor auditor trainer financier
coach draper barmaid apiarist fisherman
envoy driver builder attorney fruiterer
guard drover butcher botanist inspector
guide editor caterer comedian librarian
miner farmer cleaner dairyman locksmith
navvy fitter courier engineer osteopath
nurse hawker dentist examiner quarryman
pilot jockey equestrian gardener scientist
rabbi matron farrier inventor secretary
tiler porter fireman jeweller solicitor
usher priest furrier magician stationer
valet ranger glazier mechanic accountant
vicar rigger grazier milliner auctioneer
airman sailor manager musician bricklayer

Figure A2.2: The highest scoring solution produced by the structured random search to the January 1990 Crozzle competition scoring 506 points.

The second data set, shown next, was taken from the October 1989 Crozzle competition. The corresponding winning solution is shown in Figure A2.3.
The third, and final, data set, shown below, was taken from the December 1989 Crozzle competition. The corresponding winning solution is shown in Figure A2.5.

The third, and final, data set, shown below, was taken from the December 1989 Crozzle competition. The corresponding winning solution is shown in Figure A2.5.

belt punch turkey slippers
box quilt wallet tricycle
card razor wreath stocking
food santa almonds surprise
ham scarf bicycle umbrella
home socks blitzen champagne
iron table candles chocolate
joy tools cutlery christmas
noel torch festive greetings
nuts video friends jewellery
pets xiven hampers mistletoe
ring watch holiday seasoning
sing amulet lighter streamers
ties bangle parcels surfboard
toys basket pendant celebrations
tree broccoli perfume decorations
vase camera pudding handkerchiefs
wine church pyjamas invitation
yule dancer racquet neighbours
bells dasher raisins traditions

books dinner toaster
carol donner children
cheer family crackers
china guests epiphany
clock lights feasting
drink locket football
fruit ribbon goodwill
games scales nativity
gifts skates nicholas
holly sleigh presents
party sweets reindeer
peace tinsel shopping

**Figure A2.5:** The human winning solution to the December 1989 Crozzle competition scoring 678 points.

**Figure A2.6:** The highest scoring solution produced by the structured random search to the December 1989 Crozzle scoring 504 points.

**BIOGRAPHICAL NOTE**

Geoff Harris received his BSc, MPhil and PhD degrees from Griffith University in 1982, 1986 and 1992 respectively. His research areas include computational physics, algorithms, and the automation of games.

John Forster received his BA, MSc and PhD from the universities of Keele, London and McMaster. His research areas include decision making, strategic management and the automated and human solutions of complex problems.
Formal Computing Education in Australia: Do We Practise What We Teach?

Steve Elliot
Business Information Systems
Open Learning Institute of Hong Kong
700 Nathan Road, Mong Kok, Hong Kong
Internet: SElliot@oliv1.olii.hk

We teach that the discipline of computing promotes efficiency, effectiveness, and can be applied to increase the level of quality of an organisation's products and services. But do we apply these same principles to formal computing education?

The Report of the Discipline Review of Computing Studies and Information Sciences Education, released in 1992, identified a number of principal problem areas. It is the contention of this paper that many of these problem areas would benefit from the application of our discipline.

This paper examines the range of technologies applicable to education; considers the results of a survey into the use of Educational Technology in computing courses in Australia; looks at international experience with Educational technologies in computing courses; and draws conclusions about their applicability to address problems identified with computing education in Australia.

Key words and phrases: Higher Education, Computing, Educational Technology, Policies, Australia, Discipline Review.

1 INTRODUCTION

In our computing courses we teach that the discipline of computing, in whichever form — computer systems engineering, computer science, information technology, information systems or whatever — promotes efficiency, enables organisations to improve their operations thereby becoming more cost effective, and can be applied to increase the level of quality of an organisation's products and services. It's not just the technology, we emphasise to students, it's the opportunity the technology affords to really improve an organisation that makes all the difference in these difficult times between the winners and the losers, the survivors and the statistics.

The benefits accruing from the application of computing technology are not just enterprise based. On a national basis, the effective use of IT is seen as being an essential feature of the "clever country". (DEET, 1992).

Few, if any, computing academics would disagree about the importance of the computing discipline, or its potential to support success at an enterprise or at a national level. But do we apply our discipline to formal computing education? If we don't, would it make any difference if we did? This paper seeks to address the answers. The questions should concern all involved in computing in formal education.

A starting point is to review the state of formal computing education in Australia. The Report of the Discipline Review of Computing Studies and Information Sciences Education, released in 1992, identified the following principal problem areas:

— uniformity between institutional missions
— over fragmented organisational structures
— a lack of inter and intra institutional cooperation, coordination and communication
— inadequate IT course and career counselling of students
— inadequate preparation of students for the workforce
— lowering of quality of courses due to rapid expansion
— acute shortage of qualified and experienced staff
— credit transfer. (DEET, 1992).

More than 12 months after the release of the report it appears to an observer that many of these problems remain. It is the contention of this paper that many of these problem areas would benefit from the application of our discipline. The broad aim of this paper is to identify the potential benefits of application of the discipline of computing to the processes of formal education of computing. In particular, this paper considers the application of Educational Technology.

The structure of this paper is: introduction and definition of terms; examination of the range of technologies applicable to education; the results of a survey into the use of Educational Technology in Australian computing courses; research into the experience with Educational Technologies in USA and UK Higher Education computing courses; and consideration of the applicability of the USA and UK experiences to the Australian situation. Conclusions have been drawn and specific recommendations made.
2 RANGE OF TECHNOLOGIES IN EDUCATION

Computer technology is seen to be potentially applicable to educational institutions in five distinct areas.

— The study and utilisation of computers as an integral part of teaching and research in computer science and electrical engineering.
— Administrative use of computers for information and student record management.
— The use of generic software for text processing, spreadsheet manipulation, and personal data base management for both staff and students.
— The use of computers in a communications network which enables, through the facility of electronic mail, the submission of assignments and communications between staff and students.
— A fifth area, which is probably the most controversial, is the use of the computer as a integral part of the instruction process — Educational Technology.

3 DEFINITION OF TERMS

The term ‘Educational Technology’ embraces three distinct areas: product; process; and a mixture of product and process. (Heinich, 1986).

Technology as a product refers to the software and hardware used in instructional systems. The process is that of systematically designing, implementing and evaluating the total teaching and learning system. (Gray, 1988). Educational Technology is also the mix of product and process.

There is considerable confusion about the ingredients comprising Educational Technology, in this paper the term is deemed to include both the processes of education and also those products which directly relate to Educational Technology, such as:
— interactive video;
— video;
— computer based education (or training);
— computer managed learning;
— optical scanning/scoring;
— Artificial Intelligence (especially expert systems);
— tele-conferencing and satellite delivery; and
— desktop publishing.

However, the use of computers to program assignments and/or manipulate data is not included.

The umbrella term used in this paper for the use of a computer in support of the educational processes, is Computer Based Education (CBE). Computer Based Education has two major components: Computer Aided Learning and Computer Managed Learning.

Computer Aided Learning (CAL) is the portion of Computer Based Education which delivers learning material to the student by the computer. CAL is also commonly referred to as Computer Based Training (CBT).

Computer Managed Learning (CML), is the remaining portion of Computer Based Education that monitors and controls the student’s progress. CML’s testing facilities can be implemented with a range of degrees of complexity from entry level assessment to module and subject competency testing. At a product level, there is a merging of the two CBE technologies. Some CAL products include limited management functions, and some CML systems also present tutorial material.

Student centred learning as a concept represents the antithesis of traditional learning. In a traditional or ‘lock step’ system, students are grouped together in a course which begins and ends on pre-determined dates regardless of the ability of the students; where the rate of instruction is determined by the instructor; and where the rate of learning is often ignored. (Freeman, 1985).

Student centred learning is based on the capability of students to progress through the course at their own pace (ie the course is self-paced). Educational Technology facilitates the capability of the course provider to conduct and manage, and the student to progress in a high quality self-paced course.

In this paper, the terms ‘formal education’ and ‘higher education’ refer to universities, former colleges of advanced education and colleges of Technical and Further Education. Professional computing courses are those offered by higher education aimed at preparing students for a professional career in computing.

4 UTILISATION OF EDUCATIONAL TECHNOLOGY IN COMPUTING COURSES IN AUSTRALIA

A survey was distributed in April-May 1989 to 67 organisations. The objective of this survey was to determine the level at which computing courses were utilising Educational Technology in teaching and subsequently the environment within which it was used. Results of the survey follow. Further details can be obtained by contacting the author.

The range of education providers surveyed included the formal educators: universities, former colleges of advanced education, and colleges of TAFE, as well as ‘in-house’ educators and commercial educators.

‘In-house’ and commercial educators were included to provide a perspective on the use of Educational Technologies by formal education.

While the differing roles of formal and private education are recognised, there are also significant overlapping areas of computing education between the two groups. This area of overlap is becoming more formalised in recent years, with the introduction of ‘sandwich’ type courses. Sandwich courses interleave practical industry experiences with formal education. An innovative Bachelor of Information Technology course offered jointly by several universities also includes semester-long subjects taught by commercial organisations to their ‘student’ employees.

The survey included all computing courses provided by the organisations contacted, and was not limited to specific software or hardware streams.
The survey target group was selected from an educational digest of Universities, CAEs, Institutes and TAFE Colleges offering computing courses (World of Learning, 1988). This list was updated in light of the new Universities incorporated in 1989. Wherever available, the surveys were addressed by name and title to the Professor or Head of Computing/Information Systems.

Additionally, the survey instrument was sent to a comparative sample of commercial and in-house education providers, finance industry organisations, armed services, utilities and computer companies. The industry providers were selected as being known to conduct significant education and training in computer technology for their own employees.

Profile of survey recipients

<table>
<thead>
<tr>
<th>Type</th>
<th>Total</th>
<th>Distributed</th>
<th>Returned</th>
<th>Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Banks</td>
<td>67</td>
<td>5</td>
<td>15</td>
<td>35</td>
</tr>
<tr>
<td>Computer suppliers</td>
<td>23</td>
<td>3</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>Armed Services</td>
<td>14</td>
<td>2</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>Utilities</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>Insurance companies</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>Private computer educators</td>
<td>23</td>
<td>5</td>
<td>3</td>
<td>15</td>
</tr>
<tr>
<td>TAFE</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>Universities (and University College)</td>
<td>23</td>
<td>4</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>CAEs</td>
<td>14</td>
<td>15</td>
<td>1</td>
<td>13</td>
</tr>
<tr>
<td>Institutes of Technology</td>
<td>5</td>
<td>5</td>
<td>2</td>
<td>5</td>
</tr>
</tbody>
</table>

Profile of experience with educational technologies

Q2 In what type of course would the majority of your students enrol?

<table>
<thead>
<tr>
<th>Type</th>
<th>Total</th>
<th>Distributed</th>
<th>Returned</th>
<th>Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a) degree</td>
<td>25</td>
<td>25</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>(b) Diploma, Associate Diploma</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>(c) Nationally accredited Certificate</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>(d) other award</td>
<td>6</td>
<td>6</td>
<td>5</td>
<td>2</td>
</tr>
</tbody>
</table>

Q3 What is the duration of the course taken by most students?

<table>
<thead>
<tr>
<th>Duration</th>
<th>Total</th>
<th>Distributed</th>
<th>Returned</th>
<th>Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a) 3 or more years 'full time'</td>
<td>36*</td>
<td>36*</td>
<td>15</td>
<td>25</td>
</tr>
<tr>
<td>(b) 18 months to 3 years</td>
<td>36*</td>
<td>36*</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>(c) 6 to 18 months</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>(d) 1-6 months</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>(e) 1 week to 1 month</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>(f) less than 1 week</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Q4 Primary objective of this course?

<table>
<thead>
<tr>
<th>Objective</th>
<th>Total</th>
<th>Distributed</th>
<th>Returned</th>
<th>Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a) general preparation for a career in computing</td>
<td>25</td>
<td>25</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>(b) preparation for a specific career, eg programmer</td>
<td>5</td>
<td>5</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>(c) training in a specific computing application or tool</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>(d) other (please specify)</td>
<td>3*</td>
<td>3*</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

4.1 Survey results

Profile of responding organisations.

Q1 Type of Educational Institution?
The organisations were categorised into Universities, Colleges of Advanced Education, Colleges of TAFE, Commercial Educators, In-house Educators, and 'Other'**.

Q5 All respondents could identify one or more educational uses of the specified technologies.
FORMAL COMPUTING EDUCATION IN AUSTRALIA

(a) interactive video 6 6 6
(b) video 26 23 21
(c) computer based training 17 13 17
(e) computer managed learning 10 7 9
(f) optical scanning / scoring 8 8 1
(g) expert systems 12 10 9
(h) desktop publishing 24 23 12
(i) other (please specify) 8* 8* 8*

Not all of the 16 responses had both categories. The responses that had both categories tended to be either uniformly low in both categories, or high in one and low in the other.

As can be seen, the most significant use of Educational Technology is for the delivery of subject content via video. Desktop publishing is the next most used, but with use heavily oriented towards staff, perhaps for the preparation of lecture materials.

Least used are expert systems and interactive video. Published details of the uses of expert systems in Australian tertiary institutions suggest that the expert systems identified in the survey largely relate to the use of the expert systems software, and not in the delivery of course content via the captured experience of an expert. (Kay, 1987).

Interactive Video is subject content oriented. The survey shows that formal education has not embraced interactive video. Only one of the six users of interactive video is a university. The remainder were private providers. The reasons for this may be diverse and particular to each institution. However, when considered collectively, the failure of formal education to utilise this technology when it is used extensively by private educators is seen as significant.

Analysis of the responses to this Question show clearly that private organisations, be they in-house educators or commercial education providers, are significantly higher users of Educational Technology than formal education institutions.

Profile of educational process

Q6 Are there any computing courses in your organisation based on the concept of student self-paced learning?

<table>
<thead>
<tr>
<th>'Yes' Response</th>
<th>No Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>Universities</td>
<td>5 8</td>
</tr>
<tr>
<td>CAE</td>
<td>3 9</td>
</tr>
<tr>
<td>TAFE</td>
<td>1 2</td>
</tr>
<tr>
<td>Commercial/in-house/other</td>
<td>7 0</td>
</tr>
<tr>
<td>TOTAL</td>
<td>16 19</td>
</tr>
</tbody>
</table>

Q7 Could you estimate the percentage of your computing courses that have a student self-paced learning content?

Responses were sought for the categories of computerised and non-computerised self-paced learning.

Q8 Do your computerised self-paced learning systems include?

(a) delivery of theory and content 11
(b) testing 11
(c) computer managed learning 8

There were 12 responses to this question.

Note that Q5 had 10 responses to the use of CML. This is not necessarily inconsistent as staff could use the system independently of students.

Q9 What is the origin of the computerised self-paced learning system(s)?

The responses, included multiple selections.

(a) developed by computer department staff 6
(b) developed by other staff in your organisation 3
(c) obtained from another educational institution 4
(d) purchased from another educational institution —
(e) purchased from a commercial supplier 7
(f) public domain 1
(g) other (please specify) *

Of the 13 responses, three were from universities, two from CAEs, one from TAFE and seven from private providers.

Q10 Do your students prefer courses that include computerised self-paced learning?

There were 23 responses, with one multiple selection.

(a) yes 6
(b) no 3
(c) uncertain 6
(d) not applicable 9
(24)
Of the six 'yes' selections, five were from private providers. Of the three 'no' selections, two were from private providers.

It appears from this response that formal education is not only not interested in self-paced learning, but in the limited cases where it is being used, institutions generally have not determined the students' opinion of its operation.

Q11 Could you indicate the mean age of your students?

There were 35 responses, with one multiple response.

(a) less than 18 years 0
(b) 18 to 21 years 19
(c) 21 to 30 years 15
(d) 31 to 40 years 2
(e) more than 40 years 0

Universities and CAEs reported greater than 60% of students in the 18-21 year old range, with universities at 69%. The TAFE student age groups were evenly distributed between the three groups 18-21, 21-30, and 31-40 years.

Private providers had 71% in the 21-30 years group. This older student age grouping may be significant in providing the maturity to properly manage self-paced learning.

Q12 Do your students mostly attend ‘full time’ or ‘part time’?

<table>
<thead>
<tr>
<th></th>
<th>Uni.</th>
<th>CAE</th>
<th>TAFE</th>
<th>Private</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>FT</td>
<td>12</td>
<td>9.5</td>
<td>1</td>
<td>7</td>
<td>29.5</td>
</tr>
<tr>
<td>PT</td>
<td>1</td>
<td>2.5</td>
<td>2</td>
<td>0</td>
<td>5.5</td>
</tr>
<tr>
<td></td>
<td>13</td>
<td>12</td>
<td>3</td>
<td>7</td>
<td>35.0</td>
</tr>
</tbody>
</table>

These attendance patterns appear representative of formal education, with universities and CAEs primarily involved with full time students. Similar attendance patterns for formal and private educators, in addition to common primary course objectives (as outlined in Question 4), reinforce the relevance of industry’s educational experiences to formal education.

Q13 Numbers of students during the courses.

There was a wide variation in numbers as most formal institution responses included students in introduction to computing courses serviced by the faculty of computing. There were twenty nine responses in part or full to this question. Many responses were highly qualified even though responses were confidential e.g “response misleading-many objectives”; “too many courses to answer”; “course too new (in fourth year)”; “vague estimate”; “too much change to identify”; “too early to know”.

The question construction was appropriate to a formal education environment but was not able to provide the relevant details of private educators such as the total number of students educated annually in courses with a duration less than one year.

Excluding two responses which had beginning student numbers greater than 1,000 the average of the responses indicated a heavy emphasis on beginning students with a rapid drop off in numbers. It was not clear as to the reason(s) for this rapid drop off in numbers. A significant percentage is thought to represent drop-outs from their courses.

Q14 Estimate the cost per student of Educational Technology

Of the 35 surveys used, there were 22 responses to this question. The 12 responses which provided costs can be clustered into three groups:

<table>
<thead>
<tr>
<th></th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>cost per student</td>
<td></td>
</tr>
<tr>
<td>$200-$300</td>
<td>2</td>
</tr>
<tr>
<td>$1000-$7000</td>
<td>9</td>
</tr>
<tr>
<td>$15,000</td>
<td>1</td>
</tr>
</tbody>
</table>

It appeared from the responses that the question was not understood by some respondents.

One CAE reported 1,200 students at the mid-point in its computing courses with an average of $4,000 per student worth of Educational Technology. An investment in Educational Technology of $4.8 million seems exceptionally high! This CAE included the comment, ‘computers used for practical work, not CBT’. 

There were 14 blank responses to the question, and nine responses that the figures were not available or that the head of the computing school or department could not provide an estimate.

Since the survey was confidential, one reasonable conclusion from this inability to estimate or provide a value is that cost effectiveness in course delivery is not seen as an educational or operational performance indicator.

Q15 Estimate the ratio of Educational staff costs to Educational Technology costs.

Interestingly, while only 12 educators could answer with a value for Q14, 15 could provide a ratio of Educational staff: Educational Technology.

However, one response, (1:25,500) was not considered as it did not seem appropriate to the question. Perhaps the ratio should have been 25,500:1, but this is inconsistent with the Educational Technology cost per student of $5,000 from this CAE, which would give staff costs in excess of $127 million.

With this exception, responses were:

<table>
<thead>
<tr>
<th>Ratio of staff to Ed. Tech.</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>less than 2:1</td>
<td>6</td>
</tr>
<tr>
<td>2:1 to 5:1</td>
<td>4</td>
</tr>
<tr>
<td>6:1 to 10:1</td>
<td>1</td>
</tr>
<tr>
<td>11:1 to 20:1</td>
<td>1</td>
</tr>
<tr>
<td>100:1 or greater</td>
<td>2</td>
</tr>
</tbody>
</table>

All of the responses from private educational providers were in the less than 2:1 category.
Q16 Is Educational Technology cost effective over five years?

Nineteen responders reached a conclusion. Fifteen were "Yes" and four were "No". Five were uncertain.

Of the "No" responses:

— one university was unwilling or unable to estimate the cost of the Educational Technology it was using, (mainly CBT), but felt it was not cost effective.

— one CAE had an opinion but no direct experience.

— one CAE had experience with an 'in-house' developed package which had a high cost of $5,000 per student, (with 150 students, so an overall cost of $750,000!), and

— one CAE which declared that it was a heavy user of Educational Technology but while it could not estimate the cost of the technology, thought it was not cost effective.

It should be noted that only one of the four could estimate the cost per student, (the 'in-house' development project which appeared to have gotten out of control), at $5,000 per student!

Five expressed uncertainty as to cost effectiveness. Both major and minor users of Educational Technology were uncertain. There was no common type of technology resulting in uncertainty.

Of the 15 'Yes' responses, four added comments:

— One CAE was using video and expert systems and anticipated using a wider range in the near future.

— Two commercial organisations with very substantial 'in-house' education requirements were fully committed to the use of technology, especially in high volume situations.

Out of the seven technology categories listed on the survey, one company utilised five, the other company utilised all seven.

One of these companies stated that it planned to commit $A6.5 million to Educational Technology and expected a payback within 2 years.

— The fourth was a public organisation with a very large training requirement. Their comment was that their return on investment would be marginal over 5 years, but had a realistic life of much longer, with increasing cost effectiveness.

Of the 'Yes' responses, 13 used video; 9 used CBT; 9 used CML; and 5 used expert systems.

Nine of the fifteen 'Yes' responses could estimate the cost per student.

Applying the 'Yes' responses to the Table from Question 14, the cost per student was $2,000.00

4.2 Conclusions from the survey

1 Survey construction. Question 1 permitted an unexpected response. The category 'other' was intended to permit a wider range of educational institutions than was ultimately distributed. Question 13 was appropriate for formal education but was not able to provide the relevant details of private educators such as the total number of students educated annually in courses with a duration less than one year.

In other respects, the survey construction met the requirements.

2 The survey established as fact that courses in computing subjects offered by formal education in Australia make little use of Educational Technology.

3 The survey provides proof that there is no inherent attraction to the use of Educational Technology in computing departments. This represents an inherent contradiction in that computing departments teach the disciplines and benefits of computing but do not utilise the available range of Educational Technology.

4 This survey places most of the formal education activities in the 'not applicable' or 'early exploratory' stages of the implementation of Educational Technology and establishes that industry and 'in-house' educators are active in the growth, maturity and institutionalised stages. (Frand, 1989).

Given the small industry sample of this survey, this conclusion may not be supportable except in its confirmation of US studies by commercial organisations of the cost effectiveness of Educational Technology. (DEC 1983, Bosco 1988, Galagan 1989). Organisations which are concerned with the importance of cost effective forms of education are more likely to embrace the technology.

5 Since the survey was confidential, one reasonable conclusion from the inability of formal education to estimate or provide a value for the cost of education per student is that cost effectiveness in course delivery is not seen as an educational or operational performance indicator.

6 Similar attendance patterns for formal and private educators (from Question 12), in addition to the common primary course objective (as outlined in Question 4), reinforce the relevance of industry's educational experiences to formal education.

It also emphasises the demand for continuing education within the computing industry which could be increasingly met by formal education.

7 Student drop-out rates from formal education. The survey required student numbers to calculate the cost of Educational Technology for each course. Responses on student numbers at the beginning, during and at the end of a course included many qualifications and explanations. Due to these qualifications it was not possible to confidently draw any conclusions.

Informal discussions regarding student retention rates indicated there was a significant problem. However, there is a lack of factual evidence on drop-out rates in all courses. Formal attempts to obtain further details were unsuccessful, with one exception. One institution provided details of reten-
tion rates in computing courses over time on condition that the
details were not made publicly available.

Course retention rates appear to be a particularly sensitive
issue for academic institutions. Publication of retention rates
in conjunction with validated standards of teaching would
provide a much needed focus on teaching practises.

More research in this field is urgently required.

The establishment of a standard for reporting of retention
rates is of paramount importance. The standard should ena-
ble reporting for each year, by course, for each institution;
show commencement and graduating numbers and cope
with transfers between courses. The standard must also
differentiate between computing students and those students
of other disciplines who are obliged to take a computing
subject.

The standard should be structured to enable comparisons
between courses and institutions to focus academic attention
on the importance of teaching. Perhaps the standard could also
incorporate a change in emphasis from 'student drop-out
rate', (surely the student's fault!), to the more responsible
'course retention rate'!

8 Self paced learning is not widely practised in formal educa-
tion. Research is urgently required to identify and quantify the
potential linkage between self paced learning and improve-
ments in retention rates.

The survey has established that, in fact, formal computing
education in Australia does not effectively utilise educational
technologies. This represents an inherent contradiction: we in
formal education teach the disciplines and benefits of computing
but fail to apply them. As well as presenting this contradic-
tion, formal computing education, in general, is ignoring
significant advantages in both education and the cost effective
delivery of education which can accrue from the use of Educa-
tional Technology. Education providers in private indus-
try have been quick to recognise and to gain these advan-
tages. Perhaps educators are aware of these examples in
private industry, but do not find the arguments for their use
compelling. International use of Educational Technology in
computing courses is now considered.

5 INTERNATIONAL USE OF EDUCATIONAL
TECHNOLOGY IN COMPUTING COURSES

Research indicates that the UK government's Nelson Report
and the subsequent Computing in Teaching Initiative have
been the most active national initiative in Educational Tech-
nology. The most internationally significant implementations
of Educational Technology by individual higher education
institutions have been, or have originated, in the USA.

5.1 The UK experience

In December 1983, the Nelson Committee released a Report
on Computer Facilities for Teaching in Universities.

The purpose of the study was:

"To consider, and to make recommendations to the [Com-
puter Board for Universities and Research Councils] on the
type and level of facilities that should be provided for
teaching and on the services that should be provided by
computing centres to support teaching facilities." (Nelson,
1983).

Interestingly, in what must have been a bold opinion in
1983, the Committee adopted a broad view of the use of
computing:

"We interpret ‘computer facilities for teaching’ as the
properly supported hardware and software tools required
by students at all levels and in all disciplines to facilitate
learning. We stress that this extends far beyond the present
dominant use of teaching students to program and manipu-
late computers." (ibid)

This report established the level of utilisation of comput-
ing:

"...existing computer facilities are inadequate, both in
quality and quantity, for the genuine needs of students on
undergraduate and postgraduate taught courses." (ibid).

The Committee considered it necessary to collect details of
the current use of computers in teaching and to seek from the
universities their views on future developments. Unfortu-
nately, this information was not forthcoming. The Report
notes that "there is a lack of planning for teaching facilities in
a majority of institutions", and that "universities do not appear
to accord a sufficiently high priority to teaching needs in this
area". (ibid).

The Committee identified 14 potential uses of Educa-
tional Technology which could be used by the universities.
These included student oriented aids such as syntax check-
ers, debugging tools and user friendly command systems;
computer managed learning; electronic blackboard; elec-
tronic mail; computer assisted learning systems; drill and
practice systems; simulations; and open access systems,
where the student may work autonomously at home or place
of work.

As a result of the Nelson Report, the Computers in Teach-
ing Initiative (CTI) was established in 1984 to facilitate the
integration of computers into the university curriculum.

The primary aims of the CTI:
— "To encourage the development of computer-mediated
training and learning in UK universities.
— To evaluate the educational potential of information tech-
ology within the context of university teaching in the UK.
— To promote an enhanced awareness of the potential of
information technology among academics and students in
all disciplines." (Gardner, 1987).

To 1987, a total of 138 CTI projects had been initiated in
46 universities in the UK, with financial support of approxi-
mately $A23,500,000.

An analysis of the CTI projects over 19 academic disci-
plines and one general category, shows that the most interest
in the application of Educational Technology to teaching was
in Physical and Biological Sciences (15 projects each), with
Engineering, Mathematics, Medicine, Geography, Business
and Languages also substantially involved.
Without discounting the personal and academic aspects, the Technology that there is no inherent attraction to the use of based dissemination centres were established at 20 universi­

ties. The limited interest illustrated by the modest participa­
tion of computer science academics in 1987 had, by 1992, shown a substantial change.

The least involved disciplines were:
Politics (one project),
Archeology (two projects),
and with three projects each:
Philosophy, Classical Languages, and
Computer Science! (ibid)

This represents an international confirmation of the con­
clusion from the survey of Australian use of Educational Technology that there is no inherent attraction to the use of Educational Technology in computing departments.

Speculation as to the cause for this lack of commitment to the use of the technology in the teaching of the technology identifies a number of factors. These factors could range from personal reasons such as a simple fear of change, through to academic disdain for the application of their area of research, (or worse, application of someone else’s area of research). Without discounting the personal and academic aspects, the CTI paper supports a structural factor,

"... current university rewards systems, with unremitting pressure to engage in primary research, do not favour a wholehearted involvement in projects on computer medi­
at ed instruction, (or, for that matter, any form of teaching activity).” (ibid).

Funding of the initial 138 CTI projects ended in 1988. In 1989, in the second phase of a continuing process, discipline­

based dissemination centres were established at 20 universi­
ties.

In 1991 an independent review of the CTI was conducted. In their conclusions, the reviewers stated they were, "continually impressed with the quality of the service provided from a low funding base.” (Darby, 1992).

The CTI is now being seen as a role model, with similar bodies being established in Singapore, Sweden and the Rep­

ublic of Ireland.

Apart from its general success, the CTI must be judged to be a success from the specific viewpoint of computing education. The limited interest illustrated by the modest participa­tion of computer science academics in 1987 had, by 1992, shown a substantial change.

A CTI associated body, the Information Systems Committee Courseware Development Working Party published a report in July 1992 about the physical and human infrastructure required to profit from learning technology. This study reported the results of a survey that almost 70% of respondents were in favour of using computer based programs developed by a nationally funded consortia. The Not Invented Here syndrome, which had been of concern previously, was not indicated by this result. Notwithstanding on-going concerns about the limited supply of quality products, this represented a significant cultural change by academics. (Laurillard, 1992).

5.2 The USA Experience

Literature reviews have revealed a number of internationally significant examples of the use of Educational Technology in US universities and colleges. Rather than attempt a comprehen­sive survey of all activity in this area, several examples have been selected which are most relevant to computing educators.

Project Athena at MIT is a SUS 120 million exploration of the potential uses of advanced computer technology in the university curriculum. Both DEC and IBM have supported the project with hardware, software, equipment maintenance and technical staff.

Athena does not just represent pure research. The current and practical educational requirements of the institution have not been ignored. Approximately $US 10 million has been raised by MIT for allocation to faculty members to develop software for use in MIT’s curriculum.

The cost of the technology received special consideration in MIT’s planning. Their conclusion was,

“The payoff of computers is improvement in the quality of education, and not in reduced cost.” (Balkovitch, 1985).

Stanford University has a formal program of support for faculty developments in Educational Technology. Between 1984 and 1987 the Faculty Author Development Program sponsored the development of 36 projects.

Of particular interest to computing educators is a Stanford project called Turing’s World. Students were asked to build a Turing machine to compute a certain function or solve a particular problem. Through the use of a mouse and menu, the student could draw a flowchart of the logic. The software converted the flowchart into a Turing machine and then executed the logic. Complex functions could be decomposed into smaller integrated modules, each of which could be subsequently executed.

The developers of Turing’s World cited the near impossibility of teaching computability theory without a means of determining if the logic was correct.

The effectiveness of the system becomes very cost effective when combined with Stanford’s interactive classrooms. There the instructor can also display and compare the approaches of different students to the same problem. (Osgood D, 1987).

Carnegie-Mellon University and IBM have jointly developed the Andrew Project. Andrew is aimed at the development of a campus wide networked workstation environment. To use Andrew’s capabilities in their courses, instructors needed to develop course-ware. The CMU Tutor was developed to meet this need. CMU Tutor is an authoring environment for creating interactive course-ware which includes a high level of graphics. (Morris, 1986).

The Carnegie-Mellon development work was undertaken through collaboration between university staff and seconded IBM employees. This appears to be a satisfactory means of ensuring an appropriate focus for the project. The University’s reward structure was maintained, but a separate (and well funded) project structure established.

Of additional benefit is the flow-on effect which raises the general level of academic interest in, and flags the potential benefits to be accrued from, the development of Educational Technology.
One example of a developed product used in Carnegie-Mellon’s computing courses is the Lisp Tutor. Described as an Intelligent Tutor, the authors have produced, “a viable piece of software that teaches a one-semester course” in which “students scored one letter grade higher on final written tests when they worked with the tutor”. (Anderson and Skwarecki, 1986).

The products resulting from this project provide ample proof of the applicability of Educational Technology to computing education.

Brown University developed BALSA in 1983. BALSA (Brown Algorithm Simulator and Animator), is used to provide dynamic visualisations of programs implementing algorithms and data structures. It has proved to be so successful as an experiment in ‘electronically assisted teaching’ that BALSA has now been integrated into a variety of courses in computer science, maths, neural science and even political science.

Following the implementation of BALSA, Brown University established an Institute for Research on Information and Scholarship (IRIS) which, “not only creates scholar’s workstation software, but also has a branch where social scientists study the need, requirements and impact of this technology on scholarly work” (Van Dam, 1988). Van Dam emphasises that students as well as staff are considered scholars.

Distance education has especially benefited from the application of Educational Technology. The major difficulty in conventional distance education is the lack of interaction between students and teaching staff. Technology has been applied to conferencing; messaging; assignment submission and return; as well as for the delivery of course content in order to address this difficulty.

USA course providers include the Electronic University Network; the American Open University; and Western Behavioral Sciences Institute’s School of Management and Strategic Studies (SMSS).

While their names may not be familiar to Australian readers, the course providers cannot be dismissed as being insignificant institutions whose experiences are not relevant. An examination of leading US business management schools by Harvard Business Review in the 1980’s placed SMSS in the top five. (Meeks, 1987).

The Electronic University Network (EUN) acts for a consortium of 200 colleges and universities. Each college develops its own course for inclusion in the EUN catalogue. If a student takes a course from Harvard, then Harvard grants the credit, not the EUN. The essential idea of the EUN is, “Instead of the student having to revolve around the instructor being in a certain building at a certain time on a certain day, now the entire program including the instructor revolves around the student.” (ibid).

The American Open University, established by the New York Institute of Technology, has some 130 courses. Their experience is that Educational Technology has increased their cost effectiveness — which benefits their students:

“On-line students don’t tax our overhead and don’t use buildings or electricity so we don’t have to charge them as much.” (ibid p184).

California State University (CSU), Chico has experience with technology delivered distance learning which dates back to 1975. In that year CSU commenced operation with a microwave network which now delivers 25 courses each semester to students at 16 learning sites in northern California.

From 1984, CSU has been also offering courses for its master’s degree in computer science by satellite delivery, and claims to be the first institution to offer a satellite-delivered bachelors degree (in computer science).

With more than 700 enrolments each semester in the technology delivered courses and a wealth of experience, CSU’s considers that:

“Using telecommunications to provide easy access to education is efficient and cost effective” (Meuter, 1989).

It cannot be said that these applications are so recent, or the technologies so advanced that Australian institutions have not had the opportunity to exploit them. These examples are well established, tried, tested and proven over time. Many date from the early 1980s. The benefits cited, if not accepted as compelling, are at least of sufficient importance for serious consideration. With few notable exceptions, there is little evidence that this has occurred in the vast majority of Australian institutions in formal computing education.

6 POTENTIAL BENEFITS OF EDUCATIONAL TECHNOLOGY FOR AUSTRALIAN EDUCATORS

The examples of international experiences cited have identified several potential benefits of Educational Technology.

These benefits can be grouped as:
— quality of education
— cost effectiveness in providing quality education
— capability for delivery of a high and consistent standard of education in distributed areas.

These potential benefits are now considered in the Australian context.

1 Quality of education

This benefit relates to both the means of overcoming problems associated with traditional methods of education and the potential to further enhance education through the application of technology.

Problems with traditional methods of education include:
— concern over the quality of education following the 1992 Discipline Review of Computing Education in Australia. This Discipline Review identified a range of principal problems which included:
— inadequate student preparation for the workforce;
— problems arising from rapid expansion in student numbers (such as lowering of quality of teaching; reduction in progression rates; poor accommodation for students;
students with lower levels of tertiary entrance scores being accepted; resource problems due to lack of provision of adequate infrastructure); and
— staffing (acute shortage of suitably qualified and experienced academic staff and support staff).
— concern over the cost of traditional forms of tertiary education — which will rise from $2,500 million in 1987 (Dawkins, 1987) to $4,400 million in 1994. (Baldwin, 1993).

Appropriate applications of Educational Technology could enable flexible, self paced, student centred education which catered for the different learning requirements of students with different levels of ability and experience in computing. This opportunity was clearly stated in the conclusions of a US Government evaluation:
“The greatest promise of technology is that it has the capability to manage and deliver learning geared to the needs of each student”. “Group learning ... is a mediocre system that discourages and underserves both the most able and least able”: conclusions of the Report of the U.S. National Task Force on Educational Technology, 1986.

Student Centred Learning is self paced learning, which, when coupled with Educational Technology, can improve the quality of education by enabling:
— better education for the more or the less able students;
— instantaneous feedback to the students on their progress.
— more flexible use of the computer equipment facilities, the rate of student attrition to be reasonably reduced as the students learning at faster and slower rates than the norm are catered for. Benefits stemming from a lower drop-out rate are:
— enhanced personal esteem for the potential drop-outs;
— value added to the community in the nationally strategic area of computing and information systems.
— reduced risk of under utilisation of expensive computer equipment due to insufficient students, and in consequence,
— a higher utilisation rate of the facilities.

Additionally, a substantial benefit in course quality may be accrued simply due to the examination of teaching processes during the development and implementation of Educational Technology.

2 Cost effectiveness in providing quality education
Private education providers, both ‘in-house’ for their own employees and commercial providers, make widespread use of Educational Technology. Their reasons are two-fold: quality of education, and cost effectiveness.

One Australian organisation stated in the survey above that, based on its experience of a high level of use of Educational Technology, it anticipated recovering a $6.5 million investment in Educational Technology within two years.

The survey of Australian experiences identified that 15 out of 19 organisations considered Educational Technology cost effective over a five year period. Furthermore, the cost effectiveness could be achieved at very low levels of investment. Based on this research, it has been determined that the greater the integration of technology and curriculum, the greater the cost effectiveness. In many cases, especially with dated courses, it is necessary to redesign the curriculum to obtain the maximum benefits.

3 Capability for delivery of a high and consistent standard of education in distributed areas
The nature of Educational Technology is such that where subject content material has been prepared, a means of student assessment developed, and a course management process implemented, there is wide flexibility in the location and the times of delivery of the course.

This capability is particularly important where there is a widespread requirement for education in a discipline such as computing which is of strategic national importance but which also experiences a substantial shortage in the number of educators able to present the discipline at an appropriate level. An acute shortage of staff in Australia was highlighted in the Discipline Review.

These technologies have only the potential rather than a guarantee of performance because, as with all technology, it is the proper analysis, design, integration and implementation of the Educational Technology which will resolve problems — not the Technology itself.

In addition to the benefits identified from international experience, it is important to review the list of principal problem areas noted by the Discipline Review. Three of the eight principal problem areas have been dealt with under the heading of Quality of Education, above. These three are preparation of students for the workforce, the impact of rapid expansion of student numbers and staffing. Two additional problem areas have the potential for improvement through the application of either Educational Technology, (inadequate IT course and career counselling), or telecommunications technologies, (fragmented organisational structures and lack of cooperation, coordination and communication).

7 CONCLUSIONS
A survey presented in this paper established as fact that courses in computing subjects offered by formal education in Australia make little use of Educational Technology.

This survey further established that there is no inherent attraction to the use of Educational Technology in computing departments and that this represents an inherent contradiction. Computing departments teach the disciplines and benefits of technology but, with few notable exceptions, do not utilise the available range of Educational Technology.

Experiences with Educational Technology in the USA and the UK were examined. The UK data also indicated that this inherent contradiction had been previously present in British computing departments.
Research has identified that Australian expenditure on Higher Education for 1994 is planned to be $4,400 million, and that there is considerable encouragement, if not actual pressure, from Federal funding sources for Higher Education to become more cost effective.

Furthermore, it has been identified that current courses delivered by traditional means of education suffer from problems which include low levels of quality, lack of relevance to students and limitations due to an acute shortage of staff.

Based on an examination of the range of technologies applicable to education; a survey into the use of Educational Technology in Australian computing courses; research into the experience with Educational Technologies in USA and UK Higher Education computing courses; and consideration of the applicability of the USA and UK experiences to Australia; a conclusion has been reached that Educational Technology has the potential to significantly contribute to professional computing courses in Australia.

Educational Technology has been shown as able to be successfully applied to provide high quality, flexible, cost effective computing education which can help overcome shortages of experienced staff. In fact the majority of the problem areas identified by the Discipline Review were seen as able to be positively assisted by application of technology in general, and Educational Technology in particular.

Unfortunately, this application appears unlikely to occur. There is little impetus for widespread adoption of these technologies.

Factors preventing this adoption have not been conclusively determined. The UK experience identified a structural factor with its pressure on academics to engage in research as opposed to any teaching activity. While this may strike a chord in the minds of many Australian academics, it cannot be presumed to be the sole, or even the most important factor.

Further research is required to identify and to quantify these factors. In the absence of such research, any consideration of the reasons must be considered speculation. However, it can, at times, be fruitful to speculate.

Speculation identifies a number of factors in addition to structural considerations. These range from personal reasons such as a simple fear of change, through to academic disdain for the application of research. Other reasons could be:

— lack of assistance to develop materials,
— lack of facilities to deliver Educational Technology. (At times it can be sufficient challenge to obtain a working OHP!), and
— lack of incentives for staff to get involved.

Academics in all disciplines face these conditions and challenges. However, it appears that academics in disciplines other than computing manage to overcome them. The 1993 National Teacher Development Grant Scheme Awards recognised the work of nearly 90 academics. At least 50 of these awards were for applications of Educational Technology. Not one of the awards was identifiably in the discipline of computing. (ACRW, 1993).

The first part of any solution is to recognise the inadequacies of the current situation. This paper has identified a great need in our discipline for a vehicle to develop an Educational Technology consciousness. This vehicle could also facilitate sharing of experiences and the maintenance of an up to date register of products relevant to the discipline. Establishment of a discipline centre along the lines of the UK’s Computers in Teaching Initiative could be the vehicle to promote and support the effective and efficient use of Educational Technology.

Long term solutions will almost certainly require structural change to the present system to introduce an appropriate emphasis on and reward for teaching.

But in the meantime, there is much computing academics can do to develop a greater understanding of the potential benefits of Educational Technology. In the absence of institutional or departmental support, this may have to be an individual exercise. The starting point for this exercise may well be the answer to the critical question: how long must our students wait before we can claim to practise what we teach?

BIBLIOGRAPHY


THE AUSTRALIAN COMPUTER JOURNAL, VOL. 25, No. 2, MAY 1993 59


Biographical Note

Steve Elliot is currently senior lecturer and programme leader for Business Information Systems degree courses at the Open Learning Institute of Hong Kong. Since commencing in the computer industry in 1972 he has worked in Australia, Europe, North America and Asia in business, government, education and with the United Nations. He holds degrees in economics and information science, and a technical education qualification, from the University of Sydney and the University of Technology, Sydney. His current research interests are strategic information systems planning, management of IT, co-operative education, quality in computing education and applications of Educational Technology.
A Randomised Schema Mutator for Evolutionary Database Optimisation

Patrick van Bommel
Department of Information Systems, Faculty of Mathematics and Informatics, University of Nijmegen, Toernooiveld 1, 6525 ED Nijmegen, The Netherlands
E-mail: pvb@cs.kun.nl

In this paper we focus on randomised evolutionary optimisation. We introduce a general framework for the optimisation of data models, based on the concept of evolution. This evolution is guided by a randomised schema mutator. Although our approach is expressed in terms of database optimisation, our ideas are applicable to other fields of randomised evolutionary optimisation of computer models, especially when similar (graph structured) models are used.

Keywords and phrases: global optimisation, evolutionary optimisation, adaptive search, randomised algorithms, conceptual data models, transformation of data models, database optimisation.

1 INTRODUCTION

1.1 Intention of the paper
In this paper we describe the underlying algorithm of a Prototype Evolutionary Database Optimiser, under development at the Department of Information Systems, University of Nijmegen, The Netherlands.

The paper has three main contributions. Firstly, we introduce a formal framework for non-deterministic evolutionary (or adaptive) optimisation of graph-structured computer models. The models we use describe database structures.

Secondly, we define a pair of basic mutations (adaptation operators). Each operator transforms a given database model uniquely into a new database model.

Thirdly, we introduce a simple evolution algorithm. This algorithm non-deterministically generates mutations for a given set of database models (the parents), in order to find new (better) models (the offspring). The formal nature of the paper makes implementation, theoretical analysis and future extensions easier.

1.2 Background and overview
The optimisation of databases involves several complex task, such as database structure optimisation (Amikam, 1985; Prabhakaran, 1984), optimisation of access paths or index structures (Anderson and Berra, 1977; Ip, Saxon and Raghavan, 1983) and query optimisation (Ceri and Gottlob, 1985). In this paper we focus on the first aspect. We consider this problem as follows: for a given conceptual data model, find an internal representation (database structure) with optimal storage requirements and response times. For a general introduction into the main concepts we refer to van Griethuysen, (1982).

The conceptual data models we use consist of so-called fact types between object types (Nijssen and Halpin, 1989). We do not consider constraints on fact types or subtyping of object types. For optimisation approaches that incorporate constraint transformation, see Halpin, (1991).

The internal representation of conceptual data models is specified in terms of the conceptual model at hand. We show that our internal representations can be easily interpreted as nested tables. Therefore the default target data model used for implementation will be the nested relational model. However, also other data models can be obtained from the internal representations. For example, a normalised relational model is obtained by flattening the corresponding nested tables. More details are found in van Bommel, (1993) and van Bommel, Kovacs and Micsik, (1992).

The search for an optimal internal representation is established by a randomised mutator. This mutator enables us to navigate through the search space (solution space) of all possible internal representations. Recently such evolutionary optimisation has been applied in a number of fields, for instance network architectures, telecommunication networks, aircraft design and chemistry (Davis, 1991; Goldberg, 1989).
and Rawlins, 1991). These powerful evolutionary optimisation techniques are especially interesting for complex optimisation problems with many local optima. For other global optimisation approaches we refer to Benke and Skinner, (1991) and Torn and Zilinskas, (1989).

The organisation of the paper is as follows. In section 2 we briefly discuss the framework for our approach. Then we define the search space of the database optimisation problem in section 3. Next we consider the evolution process within this search space (section 4). Finally we discuss conclusions and directions for further research in section 5.

2 FRAMEWORK

In this section we introduce a framework for evolutionary optimisation of database models. More general definitions can be found in Rawlins, (1991) and Torn and Zilinskas, (1989).

The process of database design is usually preceded by the process of information analysis. The result of information analysis is a conceptual data model, specifying what kind of data must be stored in a system (Chen, 1976; Nijssen and Halpin, 1989). The process of database design is then initiated with this conceptual model. It results in an internal representation (database structure), specifying in what way the data is represented on a computer (van Griethuysen, 1982).

For a given conceptual data model, the set $S$ of all possible internal representations may be very large. This set $S$ is usually called the search space (or solution space) of the optimisation problem under consideration. Some elements of $S$ will result in an efficient system, others will not. We consider the problem how to find a good element of $S$, i.e., an efficient internal representation.

For this purpose we introduce a fitness function $F : S \rightarrow \mathbb{R}$, assigning a specific (real valued) fitness to each element of $S$. It will be obvious that this fitness function expresses a measure for the performance of the internal representation at hand. An introduction into the necessary performance considerations can be found in e.g. Prabuddha, Park and Pirkul, (1988); Ceri and Gottlob, (1985); Anderson and Berra, (1977); Blanken, (1984) and Weddell, (1987).

Let $s_1, s_2 \in S$ be candidate internal representations. Then, a fitness function $F$ with domain $S$ should be interpreted as follows:

$$F(s_1) > F(s_2) \iff s_1 \text{ is better than } s_2,$$

In general the fitness $F$ of an internal representation $s \in S$ takes into account the expected storage requirements $\text{Stor}$ and the expected access time $\text{Time}$ (van Bommel, 1993 and van Bommel, Lucasius and van der Weide, 1993):

$$F(s) = f(\text{Stor}(s), \text{Time}(s))$$

The function $f$ specifies how the storage requirements $\text{Stor}$ and the access time $\text{Time}$ influence the overall fitness of an internal representation, depending on the aim of the optimisation process. Example optimisation options are:

1. $\text{Stor}$ should be as low as possible. In this case the optimisation process will yield an internal representation with minimal storage cost, irrespective of the average access time.
2. $\text{Time}$ should be as low as possible. Now the result will be a representation with minimal average access time.
3. $\text{Stor}$ may not exceed a certain threshold and $\text{Time}$ should be as low as possible. This situation may occur when a large amount of information must be stored on a storage medium with a predefined size (e.g. CD-ROM).

The exact computation of $\text{Stor}$ and $\text{Time}$ is outside the scope of this paper. In van Bommel and van der Weide, (1992b) the computation of the expected storage requirements was specified in terms of the expected population of the original conceptual data model. The computation of the expected average access time will be part of future research. More details are found in van Bommel, Lucasius and van der Weide, (1993).

For the generation of different elements of $S$, we use mutation operators. A mutation operator transforms an element of $S$ into another element of $S$. In this way, navigation through the search space $S$ becomes possible.

Mutation operators form the basis for advanced evolution strategies. An evolution strategy usually processes subsets of $S$, rather than individual elements of $S$. This increases the power of such strategies (Davis, 1991; Goldberg, 1989 and Rawlins, 1991).

3 THE SEARCH SPACE

In this section we define the search space for the optimisation of database models. We first discuss conceptual data models (3.1). Then, we define wellformedness conditions for the internal representation of such conceptual data models (3.2). For a given conceptual data model, the search space consists of all internal representations, satisfying these conditions. We conclude with an encoding mechanism for the elements of the search space (3.3).

3.1 The conceptual model

In this paper we restrict ourselves to (the broad class of) conceptual data modelling techniques with an underlying object-role structure (Chen, 1976; Leung and Nijssen, 1988 and Nijssen and Halpin, 1989). The information structure in these models consists of the following basic components (van Bommel, ter Hofstede and van der Weide, 1991 and van der Weide, 1993):

- A set $O$ of object types. For example, in figure 1 we have $A \in O$.
- A set $P$ of predicators. A predicator is intended to model the connection between an object type and a role in a fact type. In figure 1 for example we have $p \in P$. The associated object type is found by the operator $\text{Base}: P \rightarrow O$. Note that both object type and role are included in a predicator,
although for transformational purposes only the object type is necessary.
— A set of fact types $\mathcal{F}$, defined as a partition of $P$. In figure 1 we have $\mathcal{F} = \{f,g,h\}$ and $f = \{p,q\}$.

3.2 Internal representations
In this section we discuss a mechanism for the internal representation of conceptual data models. This (nested relational) representation and the generation of such representations was introduced in Van Bommel and Van der Weide, (1992). Intuitively, a tree representation is the result of lifting up certain object types of the information structure. The roles will hang and cause other object types to be lifted up. In order to remove cycles it may be necessary to cut certain connections.

Formally, a forest (set of trees) $T = (N,E,L)$ is called a tree representation of information structure $I$, if it satisfies the following wellformedness conditions:

$t_1$: The set of nodes $N$ is a partition of $P$, where all predicicators in a node are mutually attached.

$t_2$: All predicicators in a node belong to different fact types.

$t_3$: $E \subseteq N \times N$ is a set of directed edges. The pair $(m,n)$ represents an edge from node $m$ to node $n$.

$t_4$: Edges are labelled by fact types in the obvious way. The function $L: E \rightarrow \mathcal{F}$ assigns $L((m,n)) = f$, if both the source node and the destination node contain a predicicator from $f$. Note that for a binary fact type $f = \{p,q\}$ this will result either in $\langle \{p\}, \{q\} \rangle$ or in $\langle \{q\}, \{p\} \rangle$, even if $f$ is a homogeneous fact type ($Base(p) = Base(q)$).

$t_5$: Fact types are located around a single parent: $L((m_1,n_1)) = L((m_2,n_2)) \Rightarrow n_1 = n_2$.

We define $R \subseteq N$ as the set of nodes where each node is the root of some tree:

$$R = \{ x \in N \mid \forall_{i \in node,E} [x \neq m] \}$$

Example 3.1 In figure 2 we see a tree representation of the information structure in figure 1. This tree representation (forest) has root set $R = \{ n_1 \}$.

The relation between nodes and edges is directly coupled to predicicators. We introduce two main properties. The proof of these properties was given in Van Bommel and Van der Weide, (1992). The properties describe the situation that edges are anchored to nodes by predicicators.

Firstly, from conditions $t_2$ and $t_4$ we see that an edge $(m,n)$ is anchored to a unique predicicator $p \in L((m,n))$:

**Lemma 3.1** For each edge $e = (m,n)$ with $L(e) = f$ we have: $|m \cap f| = 1$.

This unique predicicator is denoted as $Anchor(m)$. Secondly, for the nodes, being the destination of some edge, an analogous property holds:

**Lemma 3.2** Edge $e = (m,n)$ with $L(e) = f$ has the following property: $|n \cap f| = 1$.

For fact type $f$ this unique predicicator is as denoted $Hook(f)$.

Our concept of nested representations of an information structure is very close to the concept of nested relations, as discussed in Golby, (1990); Roth, Korth and Silberschatz, (1988); Schek and Scholl, (1986) and Thom, Kent and Sacks-Davis, (1991). In order to demonstrate this, we show how tree representations are translated into nested tables.

Suppose node $m$ has descendants as depicted in figure 3. We then construct a table with a column for $m$-values, and for each fact type hooking to node $m$. As a consequence, we get a nested table for each such fact type. This transformation is described by the function $\emptyset$. The general behaviour of $\emptyset$ is shown in figure 3.
3.3 String encoding

In this section we present an encoding mechanism for the description of tree representations as strings. This encoding has two purposes.

Firstly, we want to exploit the fact that the evolution of string individuals (Goldberg, 1989 and Holland, 1975) has a more powerful theoretical basis than the evolution of hierarchical individuals (Bickel and Bickel, 1987 and Koza, 1991). The use of this theoretical basis for our encoding is discussed in section 4.6.

Secondly, the manipulation of string individuals, resulting from our encoding, will be much simpler than the manipulation of the original tree representations. We illustrate this in section 4.1.

The string encoding we introduce is based on the concept of anchors (see lemma 3.1). Let \( \alpha \) be a function with domain \( \mathcal{P} \). For predicate \( p \in \mathcal{P} \) the function \( \alpha \) yields either a tree (in case \( p \) is in a root) or a predicate (otherwise). In order to simplify the encoding, predicates will be numbered from 1 to \( |\mathcal{P}| \) and trees from \(-1\) to \(-|\mathcal{R}|\). More precisely the function \( \alpha \) is defined as:

1. If Node\((p)\) \(\in\) \(\mathcal{R}\) then \(\alpha(p)\) yields the corresponding anchor: \(\alpha(p) = \text{Anchor}(\text{Node}(p))\)
2. If Node\((p)\) \(\notin\) \(\mathcal{R}\) then \(\alpha(p)\) yields the tree number.

The function \(\alpha\) is called the anchor function.

Example 3.3 In figure 5 we see the tree representation from figure 2, using predicates 1, ..., 7 and tree \(-1\). The function \(\alpha\) assigns e.g. \(\alpha(1) = 1\), \(\alpha(2) = -1\) and \(\alpha(6) = 5\).

64 THE AUSTRALIAN COMPUTER JOURNAL, VOL. 25, No. 2, MAY 1993
4.1 Construction of tree representations

In this section we give the necessary definitions for the construction and destruction of tree representations. Construction will be performed by the nest operator, while destruction will be performed by the unnest operator.

The concept of nesting and unnesting has been widely studied (Colby, 1990; Roth, Korth and Silberschatz, 1988; Schek and Scholl, 1986 and Thom, Kent and Sacks-Davis, 1991). Usually the effect of these operators is expressed in a rather complex way (Schek and Scholl, 1986). In order to simplify our definitions, we express these operators in terms of encoded tree representations.

First, we introduce the nest operator. Let \( p \) be an unused predicator \((\alpha(p) = 0)\) and let \( q \) be a used predicator \((\alpha(q) \neq 0)\), such that they have the same base \((p \sim q)\). Then, the operator \( \nu(p,q) \) will nest \( \text{Fact}(p) \) as follows:

\[
\alpha(x) = \begin{cases} 
q & \text{if } x = p \\
F & \text{if } x \in \text{Fact}(p) \setminus \{p\}
\end{cases}
\]

In case preconditions \( \alpha(q) \neq 0 \) or \( p \sim q \) are not satisfied, this definition will result in an incorrect tree representation. Then, node \( \{p\} \) becomes the root of a new tree:

\[
\alpha(x) = \begin{cases} 
-|R| - 1 & \text{if } x = p \\
F & \text{if } x \in \text{Fact}(p) \setminus \{p\}
\end{cases}
\]

Note that during construction a tree representation will grow until it is a complete representation, i.e. all fact types are represented. As a consequence \(|R| = 0\) in the initial stage and \(|R| \leq |F|\) after termination. More details are found in van Bommel and van der Weide, (1992a).

Example 4.1 The tree representation in figure 2 (and figure 5) may be constructed by the following sequence of nest applications: \( \nu(q,q), \nu(r,q), \nu(u,t) \). Note that other nest applications may lead to the same result.

Example 4.2 In figure 6 we see another tree representation. This representation may be the result of nest applications: \( \nu(s,s), \nu(q,r), \nu(u,t) \).

Obviously, the operator \( \nu(p,q) \) involves each predicator in \( \text{Fact}(p) \). As a consequence, the complexity \( C \) of this operator is as follows:

\[
C(\nu(p,q)) = |\text{Fact}(p)| \leq |p|
\]

Next, we introduce the unnest operator \( \mu \). We apply this operator only to leaf fact types. A fact type \( f \) is called a leaf fact type, if it has no descendant fact types lower in the tree:

\[
\forall p \in f, Hook_0[\text{Node}(p) = \{p\}]
\]

For such a fact type \( f \), the effect of \( \mu(f) \) is simply expressed as follows:

\[
\alpha(p) = 0 \text{ if } p \in f
\]

4.2 Basic mutations

In this section we introduce two basic mutation operators. These operators can be used for the definition of evolution strategies (see section 4.4), or for the definition of more advanced reproduction operators (e.g. crossover van Bommel and van der Weide, 1992b).

The first operator we introduce is the glue operator \( \gamma \). The result of \( \gamma(p,q) \) is the transportation of \( \text{Fact}(p) \) with all its descendants to \( \text{Node}(q) \). Let \( p \) be the hook of some fact type in anchor function \( \alpha \) (see section 3.3) and let \( q \) be the anchor of some node (see lemma 3.1), such that \( p \sim q \). Then, the effect of \( \gamma(p,q) \) is expressed by \( \alpha(p) := q \).

In case condition \( p \sim q \) is not satisfied, this definition will result in an incorrect tree representation. Then, node \( \{p\} \) becomes the root of a new tree: \( \alpha(p) := -|R| - 1 \). Note that the situation where \( \text{Node}(q) \) is a descendant of \( \text{Node}(p) \) will also result in an incorrect tree representation.

Obviously, the operator \( \gamma \) requires the adjustment of exactly one \( \alpha \)-assignation. As a consequence, the complexity \( C \) of this operator is as follows:

As a consequence we have:

![Figure 6. Another tree representation.](image-url)
Lemma 4.4 \( C(\gamma(p,q)) = 1 \leq |p| \)

The second operator we introduce involves the promotion of a predicate. In figure 7 we see the local effect of promotion \( \text{Promote}(p) \).

Let \( p \) be a predicate and let \( \alpha \) be an anchor function with \( \alpha(p) = p \). Furthermore, let \( q = \text{Hook}(\text{Fact}(p)) \) be a predicate where \( \text{Node}(q) \) is a root. The promotion \( \text{Promote}(p) \) results in anchor function \( \alpha' \), defined by:

\[
\alpha'(x) = \begin{cases} 
q & \text{if } x \in \text{Node}(q) \\
\alpha(q) & \text{if } x \in \text{Node}(p) \\
\alpha(x) & \text{otherwise}
\end{cases}
\]

As a consequence, the complexity \( C \) of promotion \( \text{Promote}(p) \) is as follows:

\[
C(\text{Promote}(p)) = |\text{Node}(p)| + 1 \leq |p| \]

**4.4 A simple evolution algorithm**

In general, an evolution process consisting of \( N \) generations can be specified as follows:

**Schema Mutator:**
- while \( N > 0 \) do
  - Randomised Generation:
  - \( N := N - 1 \)
- end.

The input of the process is an initial subset of \( S \), i.e., a set of correct internal representations for a given conceptual data model. This subset is said to evolve into a new subset (van Bommel, 1992b and van Bommel, 1993). In each randomised generation the following actions are performed:

1. In the first part a promotion is performed. This is done by creating a context for promotion, followed by the actual promotion:
   - A context for promotion is created as follows:
     \( x := \text{Best} \)
     \( p_1 := \text{SelectionPromotion}(x) \)
   - During the promotion, the current worst element is replaced by a mutation of the current best element:
     Replace \( (\text{Worst}, \text{Promote}(p_0)) \)

2. In the second part a predicate will be glued to another predicate. This is done by creating a context, followed by the actual glueing process:
   - A context is created as follows:
     \( y := \text{Best} \)
     \( p_{-1} := \text{SelectionPredicate}(y) \)
     \( p_{+1} := \text{SelectionGlue}(y, p_{-1}) \)
   - During the glueing process, the current worst element is replaced by a mutation of the current best element:
     Replace \( (\text{Worst}, \gamma(p_{-1}, p_{+1})) \).

A basic property involves the reachability of elements of \( S \).

**Lemma 4.7** Let \( x \in \text{Gen}(i) \subseteq S \) be a (complete) internal representation \( (i > 0) \). Then, each element \( y \in S \) can be reached from \( x \) by evolution:

\[
\forall y \in S \exists i \in \text{Gen}(j) \quad (y \in \text{Gen}(j))
\]

**Proof:** Let \( x, y \) be correct complete representations. We show how \( x \) can be transformed into \( y \) by the application of the mutation operators \( \gamma \) and \( \text{Promote} \).

First, we break down the trees in \( x \) into a set of \( |F| \) trees, i.e., each fact type is represented in a separate tree. This can be done by the glue operator \( \gamma \).

Then, the promotion operator \( \text{Promote} \) is used to represent each fact type in the same way as it is represented in \( y \); each predicate which is a hook in \( y \) will have to be promoted (see lemma 3.2).

Finally, \( y \) can be constructed by the application of the glue operator \( \gamma \).
The complexity $C$ of the evolution process is estimated as follows:

**Lemma 4.8 (Schema Mutator)** $C \leq N \cdot (|P| + 2)$

**Proof:** In each step of the algorithm, promotion $Promote$ will cost $|Node(p_i)| + 1$ (see lemma 4.5) and glue $\gamma$ will cost 1 (see lemma 4.4). As a consequence, the total complexity of each step is:

$$|Node(p_i)| + 2 \leq |P| + 2$$

Now the result is easily derived, since there are $N$ steps.

4.5 An elaborated example

In this section we discuss a very simple example in order to illustrate how the schema mutator introduced in section 4.4 operates. This example uses the information structure from figure 1. Furthermore the following query $Q$ on this information structure was used:

$Q$: Find all $E$ values being associated with $D$ values being associated with value $C = c_1$

We assume that this query is evaluated from $C$ values via $D$ values to $E$ values. We will use the second optimisation option mentioned in section 2. As a consequence, the optimisation process will search for the internal representation that has the lowest access time for query $Q$.

For the sake of simplicity the mutator will run only one iteration, i.e. $N = 1$ (see section 4.3 and 4.4). The evolution process is started from the initial generation $Gen(0)$. From this initial generation the mutator will create $Gen(1)$. These two steps will now be discussed in more detail.

$Gen(0)$: The initial subset of the search space $S$ is $Gen(0)$, being the result of the application of the nest operator (see section 4.1). Let $T_1 \in Gen(0)$ be the tree representation shown in figure 2. We assume that this representation is the best element from the initial generation.

$Gen(1)$: The subset of $S$ resulting from one iteration of the mutator is $Gen(1)$. We describe this iteration in four steps:

1a Creating a context for promotion. The procedure $SelectPromotion$ will select a predicate from representation $T_1$. Suppose predicate $s$ is chosen (see figure 2).

1b Performing the promotion. The result of promoting predicate $s$ is shown in figure 6. This new internal representation $T_2$ is added to the current subset of $S$, and the worst element is removed.

2a Creating a context for glueing. Representation $T_2$ is more efficient than representation $T_1$, since the evaluation of query $Q$ is completely in line with the tree structure of $T_2$ (i.e. the evaluation is completely downwards). In representation $T_1$ this is not the case. As a consequence, the best element in the current generation is $T_2$. Now the procedures $SelectPredicator$ and $SelectGlue$ will select two predicates from $T_2$.

2b Performing the glue mutation. In this final step the worst element is replaced by the result of the glue operator. Suppose the predicates used for glueing did not have the same base. Then the result $T_3$ of the glue mutation will be a tree representation, consisting of two separate trees. Obviously, representation $T_3$ will be less efficient than representation $T_2$, since the evaluation of query $Q$ in $T_3$ requires inter-tree accesses, while in $T_2$ this is not the case.

As a consequence, the most efficient internal representation that was found during the evolution process is representation $T_2$. We realise that the illustrative power of this example is very restricted. Firstly, the information structure under consideration is extremely simple. Secondly, we considered only one query evaluation, while storage space requirements were not considered at all. Thirdly, the mutator made only one iteration. In very large databases with complex access profiles the number of necessary iterations may be very large. The computational complexity of the evolution process will profit by the intelligent string encoding we introduced. In order to further reduce this complexity, more advanced evolution strategies may be necessary. One could think of a crossover operator, that combines two internal representations into a new representation (van Bommel, 1992b and van Bommel, Lucasius and van der Weide, 1993). To be able to investigate this field of computational complexity, we introduce a preliminary basis for the analysis of evolution strategies for internal representations in the next section.

4.6 Analysis of evolution strategies

In this section we define the basis for the analysis of evolution strategies, expressed in terms of our encoded tree representations. The analysis of string-based evolution has been based on the concept of similarity templates (Goldberg, 1989; Holland, 1975 and Rawlins, 1991). We will apply this concept to the encoding of our tree representations.

A similarity template specifies a set of string individuals with similarities at certain string positions (Goldberg, 1989 and Holland, 1975). For a formal definition the symbol * is used as don't care symbol. A similarity template $\tau$ then is an extension of the anchor function introduced in section 3.3:

$$\tau : P \rightarrow P \cup \{\cdot\}$$

Let $\alpha$ be an anchor function and let $\tau$ be a similarity template. Function $\alpha$ matches template $\tau$ if each predicate $p \in P$ satisfies one of the following conditions:

1. $\tau(p) = \alpha(p)$
2. $\tau(p) = \cdot$

**Example 4.4**

```
1 -1 -1 4 5 5 7
```

matches

```
1 -1 -1 * * * *
```

The complexity $C$ of the evolution process is estimated as follows:

**Lemma 4.8 (Schema Mutator)** $C \leq N \cdot (|P| + 2)$

**Proof:** In each step of the algorithm, promotion $Promote$ will cost $|Node(p_i)| + 1$ (see lemma 4.5) and glue $\gamma$ will cost 1 (see lemma 4.4). As a consequence, the total complexity of each step is:

$$|Node(p_i)| + 2 \leq |P| + 2$$

Now the result is easily derived, since there are $N$ steps.
The main purpose of similarity templates is the investigation of similarities among individuals, such that a causal relationship between similarities, fitness and evolution can be expressed. The effect of new reproduction operators can be evaluated by considering the probability of survival of similarity templates contained in a generation of individuals. We will not elaborate this further in this paper.

We conclude with two elementary properties. Let \( \tau \) be a similarity template with \( d \) don't care positions. We then have the following property.

**Lemma 4.9** The number of matchings of \( \tau \) is restricted by:

\[
(2 \cdot |P|)^d
\]

**Proof:** Each don't care position can be filled either with a predicator or with a tree number (see section 3.3). In both cases the number of possibilities will not exceed \(|P|\). Now the result is easily derived, since there are \( d \) don't care positions.

Obviously, the number of don't care positions will not exceed the number of predicators \((d \leq |P|)\). Using this information and lemma 4.9, a rather pessimistic upperbound for the size of the search space is as follows:

**Lemma 4.10** \(|S| \leq (2^d \cdot |P|)\)

5 CONCLUSIONS AND FUTURE RESEARCH

In this paper we introduced a formal framework for non-deterministic evolutionary optimisation of database models. We defined a pair of basic mutations, which were applied in a simple evolution algorithm.

We conclude that our notion of internal representations can be easily used in evolutionary optimisation algorithms. The string encoding we discussed enabled us to give simple definitions of rather complex operators. This has made theoretical foundation and practical implementation easier.

However, we feel that our approach should be worked out further. We recognise the following directions for future research:

1. A theoretical foundation to estimate the expected convergence of different evolution strategies. An important question will be the following: do more complex evolution strategies lead to a higher rate of convergence?
2. A further incorporation of database aspects, such as index structures (Saxton and Ragharan, 1983) and the computation of access times. These aspects will be expressed in terms of the tree representations, used in this paper. Obviously, this results in more complex mutation operators.
3. A further incorporation of conceptual aspects, such as constraints and specialisation/generalisation of object types (ter Hofstede and van der Weide, 1993; Chen, 1976 and Nijssen and Halpin, 1989).

ACKNOWLEDGEMENT

I would like to thank Dr. Ir. Th. P. van der Weide, Prof. Dr. Ir. E.D. Falkenberg and the anonymous referee for their constructive discussions and comments on this research.

REFERENCES


BIOGRAPHICAL NOTE

Patrick van Bommel received his masters degree in Computer Science from the University of Nijmegen, Netherlands in 1990. He is now a junior researcher at the Department of Information Systems at the University of Nijmegen. His main research interests include the transformation of conceptual data models into efficient internal representations.
Realisable Memory Management Algorithms

MR Hannaford and DWE Blatt
Department of Computer Science,
University of Newcastle,
Newcastle, NSW 2308, Australia

In this paper, a number of memory management paging algorithms have been developed making use of the BLI Transition Model, a model of program behaviour based on the Bounded Locality Interval (BLI) concept. Although the model itself assumes lookahead knowledge of program referencing behaviour, the algorithms presented here do not, as they develop enough "on the fly" knowledge of the reference string to allow intelligent decisions to be made on whether a locality is to be kept memory resident, or released, creating page faults in the future. As well as making decisions of this kind, it is possible to anticipate actual future page faults, and issue requests for prepaging service. Two algorithms based on this principle are also presented. All the algorithms developed have been tested by simulation of a VM environment whilst executing a set of real reference strings. Their performance is compared to existing realisable algorithms as well as a near optimal lookahead algorithm.

Keywords: Paged Virtual Memory, Prepaging, Space Time Integral, Bounded Locality Interval, Reference String.

CR categories: 4.32, 4.35.

COPYRIGHT © 1993, AUSTRALIAN COMPUTER SOCIETY INC. GENERAL PERMISSION TO REPRINT, BUT NOT FOR PROFIT, ALL OR PART OF THIS MATERIAL IS GRANTED, PROVIDED THAT THE ACJ'S COPYRIGHT NOTICE IS GIVEN AND THAT REFERENCE IS MADE TO THE PUBLICATION, TO ITS DATE OF ISSUE, AND TO THE FACT THAT REPRINTING PRIVILEGES WERE GRANTED BY PERMISSION OF THE AUSTRALIAN COMPUTER SOCIETY INC.

Manuscript received: March 1988, revised 1992.

1 INTRODUCTION

A Virtual Memory (VM) System maps programs from their address space (their virtual memory) into real memory such that only a subset of the program occupies real memory at any time and such that the system maintains automatic control (invisible to the programmer) of the allocation and deallocation procedures. The program can now be stored on virtual memory, which in practice is some secondary device, generally larger and less costly than main memory. The program address space is divided into segments and the mapping and transfer processes are carried out on a segment by segment basis. If physical memory is divided into equal sized frames, and virtual address spaces are divided into equal sized pages, then the system is called a Paged VM System. A paged system has the advantage of straightforward transfers between virtual and real memory as all pages and frames are equal in size. We will consider only paged systems in this study.

The main benefit of VM to the programmer is that the address space of programs is only limited by the virtual address space available which may be some orders of magnitude larger than the real memory available on the system. This added size can be used without regard of the memory management procedure operating. The main benefit to the computer system is to increase the level of multiprogramming possible since each program will be allocated less real memory than would be the case in a non-virtual system.

When a program references part of its address space which is currently held in real memory, the Virtual Memory Management System maps the desired virtual memory page to the corresponding real memory page where that part of the program is held and program execution proceeds. If reference is made to a part of the address space not currently resident in real memory then a 'Memory Fault' (or 'Page Fault') occurs, processing is suspended and the required part of the program is loaded into real memory, whereupon program execution proceeds normally. Memory faults do not affect the execution of the program, but do cause processing to be suspended while they are serviced, therefore adding to the amount of real time for which the program needs to occupy its real memory allocation. Also, faults increase traffic to the secondary storage system and impact other users. Virtual Memory Policies therefore seek to obtain the most efficient compromise between the use of real memory and the amount of real time 'wasted' while memory faults are serviced.

Policies for controlling memory usage in VM Systems fall into several categories. Any VM policy must contain algorithms which implement sub-policies for the allocation and deallocation of memory to/from each program running on the system. The Placement Algorithm decides where each resident program segment will reside in physical memory. For a paged system this is trivial. The Fetch Algorithm decides when each required program segment will be loaded into physical memory. The usual fetch policy is Demand Paging, loading a page only when it causes a memory fault (or page fault). If a page may be loaded before the program references
that page then the fetch policy is a Prepaging Policy. If prepaging occurs only when a page fault for some other page is being serviced then the policy is referred to as a Demand Prepaging Policy. The main prepaging policies in use are clustering or sequential prefetching, (Levy and Lipman, 1982; Smith, 1978), where a page fault is serviced by loading several pages of address space in the vicinity of the page which has actually caused the fault. Programmer requested “optional” prepaging has also been implemented (Teng and Gumael, 1984), and algorithms requiring the analysis of program behaviour have also been presented (Lau, 1982; Martinez, 1982). Any prepaging policy needs very accurate prediction of future referencing patterns of programs or else the transfer traffic between secondary storage and main memory can become swamped by transfers of pages which will not actually be required by the program.

When competition for memory occurs within a VM system (a page fault occurs with no physical memory available), the Replacement Policy determines which segment is to leave memory to make way for the required segment. The pages considered as candidates for replacement can be restricted to those pages already allocated to the program which has the page fault (called a Local Replacement Policy), or the choice can be made from all the pages on the system (a Global Replacement Policy). Replacement algorithms may also enforce a Fixed allocation of memory on programs or can allow the memory allocation to vary over time (Variable Space Policies). The main fixed space algorithms in use are First-In-First-Out (FIFO), (Levy and Lipman, 1982) and Least Recently Used (LRU), (Innes, 1977), these algorithms being implemented as local strategies. FIFO and LRU can be implemented as local or global strategies (Chusho and Hayashi, 1981). LRU is important because it uses local referencing behaviour to replace the page which has been unreferenced for the longest time.

Any global policy is by its nature a variable space policy since a page fault may replace a page of one program with a page of another program which has given a page fault. A global algorithm used widely is CLOCKS which may be summarised as First-In-Not-Used-First-Out (Denning, 1980). A variation of CLOCK which uses two ‘hands’ is used in 4.3BSD Unix (Quarterman, Silberchatz and Peterson, 1985).

The Working Set (WS) algorithm (Denning, 1980) marks pages as candidates for replacement when they have not been referenced for a particular time period. When a page fault occurs, one of the marked candidates is used for replacement. This time period τ is termed the window size for WS and it has been claimed that τ can be chosen to give performance within 10% of optimum value for demand paging, when measured by the Space Time Integral of memory usage (STI). WS has also been the basis for several other replacement algorithms (Carr and Hennessy, 1981; Ferrari and Yiu-Yo, 1983; Lieflander, Schmutz, Silberbusch and Stiemle, 1983; Smith, 1976). The Page-Fault Frequency (PFF), (Chu and Opderbeck, 1972) monitors the fault rate of a program to decide whether its memory allocation should be increased when a page fault is serviced.

The VM policy must work in conjunction with a load control mechanism to prevent too high a level of multiprogramming as this will result in insufficient memory being allocated to the running tasks on the system. This can be carried out as part of the replacement policy, or else a completely separate scheduling policy may be implemented. The WS algorithm provides a built-in load control mechanism by only allowing a program to be loaded into memory if enough memory is available to hold its working set of pages. The WS Clock algorithm (Carr and Hennessy, 1981) also provides a load control mechanism by treating loading tasks and running tasks differently.

1.1 Measuring the Efficiency of Memory Management Algorithms

The most common measure of effectiveness for VM policies is the Space-Time Integral of memory usage (STI). Other measures used include minimising the fault-rate, however this can be a misleading measure for variable space algorithms and in a multiprogramming environment. Minimising the STI for all programs on a multiprogramming system will give the maximum overall throughput for the system. The STI will be the measure used in this study, with an algorithm being considered superior for any particular program if it produces a lower STI value.

The evaluation of the STI assumes that segments are the same size and references occur at even time intervals (we use a time scale of one reference interval), therefore

\[ STI = \sum_{t=1}^{\infty} N_r + \alpha \sum_{f=1}^{NF} T_f \]

where \( N_r \) is the number of segments in memory at time \( x \), \( T_f \) is the time that page fault number \( f \) occurred, \( \alpha \) is the virtual time (equivalent to the number of references) to service a fault, \( NF \) is the total number of memory faults and \( m \) is the length of the reference string. In equation (1), the first term represents the memory cost of program execution and the second term represents the memory cost incurred while servicing page faults.

Some idea of the best possible STI performance for any program in a VM environment can also be useful. If STI values are close to the best possible then there is little point in trying to improve the performance of an algorithm by altering system parameters. In a previous study we have developed the lookahead algorithm PFC, which provides near optimal STI values under demand paging (Hannaford and Blatt, 1986). Being a lookahead algorithm, PFC cannot be used in practice, but it can be used in comparing VM algorithm performance.

We have developed a number of memory management paging algorithms making use of the BLI Transition Model (Hannaford and Blatt, 1987), a model of program behaviour based on the Bounded Locality Interval (BLI) concept. The STI for the PFC algorithm is used as a basis for comparison.
STI values for each algorithm on each of a number of reference strings are compared to “Optimal” results for the WS and LRU algorithms. These $W_{opt}$ and $LRU_{opt}$ results are obtained by finding the respective parameter value for each algorithm which will yield the minimum STI value on each reference string.

All the algorithms developed in this study have been tested by simulation of a VM environment whilst executing a set of real reference strings. The reference strings used for testing are the 39 strings used in Madison and Batson (1976) and are augmented by a very long FORTRAN reference string of 20.7 million references (designated string number 43) (Hannafor, 1985). The FORTRAN program iteratively solved a system of non-linear differential equations, running to completion in 6 iterations. Reference String 43 consisted of the entire program run. Subsequent experiments presented in section 6.4 of this paper make use of 3 other reference strings (numbers 40, 41 and 42) which were respectively the first iteration, the first two iterations and the last iteration of the same program. Strings 40 and 42 were therefore about 3.4 million references, while string 41 was about 6.9 million references.

1.2 The BLI Transition Model

In a previous study (Hannafor and Blatt, 1987) the BLI Transition Model was developed and validated. This model describes the referencing behaviour of programs as a two stage process and is based on the concept of a Bounded Locality Interval (BLI) of Madison and Batson (1976). Firstly it uses a macromodel to describe program execution as a hierarchy of localities and categorises these localities and the interactions between them. It then uses a micromodel to describe the behaviour within each locality at the lowest level of the hierarchy as a cyclical BLI. The model also provides a deterministic reference string generator which can recreate reference strings from model information, maintaining the locality hierarchy structure of the original reference string.

The micromodel plays no part in this study as behaviour at the micro-level will involve segments being referenced so closely together that all the segments comprising the BLI must be resident during the lifetime of the locality. Page faults to build up the resident set at the beginning of a locality are unavoidable and no extra information can be gained at the micro-referencing level to assist in VM memory allocation.

The macromodel, however, can provide information useful to VM strategies as it describes transitions from one locality to another. The BLI Transition Model provides three different macromodels, each categorises BLIs in different ways. These macromodels are designated ‘A’, ‘PA’ and ‘PAR’. The ‘A’ model classifies BLIs solely according to their activity sets, the set of segments which are referenced during the BLI’s lifetime. The ‘PA’ model adds a hierarchy position to the classification, while the ‘PAR’ model further adds a lifetime range. Because the hierarchy position of a BLI cannot be known until the program has terminated, only the ‘A’ model can be used as a basis for memory management, and so this is the only model we will present here.

Under the ‘A’ macromodel BLIs with the same activity set are considered to be of the same class. BLI Class number $k$ can therefore be defined as:

$$Bk = k(A, p, \tau, H, T)$$

where $A$ is the activity set, and $p, \tau, H$ and $T$ are ordered pairs of mean and standard deviation of the rank (number of cycles), lifetime, head (activity set formation lifetime) and tail (part cycle at the end of the lifetime). The complete set of BLI lifetime information can therefore be held in a BLI Class Summary.

Transitions between BLI’s are modelled by means of a directed graph of transitions between the classes. Some of the transitions themselves may have non-zero lifetimes (GAPs) and information about these GAPs (segments referenced and lifetimes) are stored in a GAP Summary. The directed graph can also be called a Transition Matrix.

2 THE BLI TRANSITION MODEL AND MEMORY MANAGEMENT

In this research project, a number of memory management algorithms have been developed which make use of the BLI Transition Model. These algorithms can be divided into two main categories.

The Category 1 Algorithms rely on information collected during a previous execution of a program as a basis for memory management decisions to be made during a subsequent execution. These category 1 algorithms should not be looked upon as lookahead algorithms but as a type of algorithm suited particularly to an environment where roughly the same workload is presented to a machine from one day to the next, perhaps just with different data. Within the first category, the algorithms are divided between those which use Demand Paging and those which allow Prepaging.

The Category 2 Algorithms collect information during execution for use during that particular program run. These algorithms use Demand Paging only.

Other researchers have cast doubts over the potential usefulness of Bounded Locality Intervals in controlling the allocation and deallocation of memory to a program in a virtual memory environment. Lenfant (1976) states that BLI’s are unlikely to be useful in memory management because a large proportion of BLI’s will not be recognised until they are too close to completion to be of use in decision making. Batson (1976) presents program behaviour results of symbolic reference strings which show a very good coverage of “reasonable length” BLI’s and states that BLI’s should therefore be useful in the control of symbolically segmented virtual memory. It is added, however, that since paged reference strings can be expected to be different in structure to symbolic reference strings, the BLI model would be less likely to be useful in normal memory management. Häkala and Pohjjanlaiti (1983) report that the good coverage of BLI’s with lifetimes of useful length which have been found in data reference strings (up to 97%) fell to only 50% when studying instruction reference traces.
However, Haikala and Pohjanlahti also report a close correspondence between BLI Activity Sets and the memory occupancy sets generated by the VMIN Lookahead Algorithm (Prieve and Fabry, 1976), and this leads to the suggestion that the BLI Model may be used as the basis for memory management policies, especially in the case of data segments. This, coupled with the reports by Hagmann and Fabry (1982) that most of the large programs they studied were dominated by the referencing behaviour of the data segments, lends weight to the usefulness of BLI’s in memory management. Haikala and Pohjanlahti also suggest the application of the BLI Model in areas where short stable phases can be utilised. It has also been suggested that program behaviour in the transitions between localities will form the most important part of virtual memory management (Maddison and Batson, 1976).

The algorithms developed here are based on the BLI Transition Model. This model concentrates on the transitions which occur between BLI’s as well as seeking to obtain typical lifetime information of the BLI Localities. The objections of Lenfant (1976) are not seen as important in this case because these algorithms make use of (in the first instance) the fact that a BLI has just completed in order to make a decision on current memory occupancy of the program, rather than making use of information regarding currently active BLI’s. These algorithms, therefore, concentrate on the transitions between BLI’s rather than the expected lifetime of a BLI which has just become active. The BLI coverage of a reference string is less important. It is possible to make use of BLI transitional information even for BLI’s with quite short lifetimes as we will attempt to predict the referencing behaviour of the program in the near future by making use of BLI termination events. The basic philosophy of these algorithms is, "Given that a BLI from Class Number X has just completed, what decision should be made regarding the program’s memory occupancy for the next stage of processing?"

Schematic diagrams of the two different categories of algorithms are now presented.
3. The algorithms will be described, and then presented in the form of flowcharts which begin at the intermediate connector 1 of Figure 3 and end at intermediate connector 2.

3 THE CATEGORY 1 DEMAND PAGING ALGORITHMS

The Category 1 Algorithms rely on BLI Transition Model information which has been gained during an earlier execution of a program. Demand paging algorithms limit the allocation of memory to a program to those references which cause a page fault, i.e. no segment which is not currently resident in main memory can be placed into memory before it has been referenced.

The category 1 demand paging algorithms cannot cause the transfer of segments into main memory (this being possible only by the segments being referenced) but can only estimate which segments will no longer be needed and remove these segments. These category 1 demand paging algorithms are given the code letter “D” (for Demand) for identification purposes, and are referred to as BLID1, BLID2, etc. The BLID algorithms differ in the amount of processing they carry out in deciding which segments should be removed from memory.

3.1 Algorithm BLID1

This algorithm is the simplest of the BLI based algorithms. When a BLI terminates, this algorithm excludes all segments from memory except for those which are certain to be required in the next locality. This algorithm therefore calculates those segments which belong to all possible transitions and excludes any other segment which may be in memory at that time. This strategy is based on the premise that if a transition is to occur then a higher page fault rate can be expected in the near future and so the best Space-Time integral result will come from having these faults occur at the lowest possible memory occupancy level.

One obvious drawback of this strategy is that one possible transition to a BLI whose activity set is disjoint from all other possible BLI transitions will result in the memory being emptied, even though most of the segments resident in memory at that time may be useful in most of the possible transitions. This would result in the fault rate being much higher in all cases for the sake of a transition which may occur only once in the program.

The main advantage of this algorithm will be in the case where most BLI’s have a transition to only a single BLI when they terminate. In this case a minimum amount of processing will be needed in order to adjust memory occupancy for the most efficient passage from one BLI to the other.

Most of the processing for this algorithm can be done prior to the program execution beginning and can be incorporated into the Initialise process of Figure 3 for the purposes of our simulation experiments. The BLI Transition Matrix is scanned and the intersection of all possible future activity sets is taken. This ensures that any set element remaining in memory will be used. This processing finishes with a list of BLI’s and for each a Memory Occupancy Set.

Definition: The Memory Occupancy Set

The Memory Occupancy Set is the set of segments used to adjust the memory occupancy within any algorithm and is the set of all segments which may remain in memory, if they are already there, i.e. the memory occupancy is adjusted by taking the intersection of the current memory set and the Memory Occupancy Set. When the program executes, the termination of a BLI results in the correct entry in the list being found and all segments in memory which are not in the Memory Occupancy Set being excluded from memory.

3.2 Algorithms BLID2, BLID3 and BLID4

Algorithm BLID1 is a very simple algorithm which reduces the BLI Transition Model information to a single decision for each BLI which may terminate during a program run. Subsequent algorithms seek to make greater use of the BLI Transition Information and to attempt to decide which of the possible transitions will occur next and to adjust memory occupancy accordingly.

If a BLI of Class X terminates, then X is referred to as the terminating Class, and it is assumed that execution will now pass through a GAP (possibly of zero lifetime) to a BLI of some Class Y. If X = Y then the GAP must have a non-zero lifetime. The BLI Transition Information is used to isolate and identify Y to whatever extent is possible and then to control the memory occupancy so that the transition to BLI Y is done efficiently (when measured in Space-Time terms).

Definition: The Terminating Segment

One additional source of information is also available, viz. the segment which has caused the BLI to terminate. This segment must either belong to the Activity Set of the next BLI or the GAP membership set (if a non-zero lifetime GAP occurs
between the BLI’s). This “terminating segment” can be used
to eliminate some of the possible transitions and so allow a
more accurate decision to be made. For example, if a BLI with
activity set {ABC} is terminated by reference to segment D,
then transition to a BLI with activity set {EFG} cannot be
possible unless the GAP between the two BLI classes contains
segment D.

BLID2 uses a “status” indicator along with the BLI Transi­
tion information to control memory occupancy during “ex­
ecution” of the reference string. This status indicates whether
BLID2 is currently executing within a BLI (status INBLI), or
within a GAP (status INGAP), or elsewhere (status UN­
SURE).

**Definition: Unique Transition**
BLID2 also makes use of a concept referred to from here
onwards as a “unique transition”. This unique transition is
defined in terms of the current status. A unique transition is
one where the status is known with certainty. A transition is
unique if all valid transitions involve zero lifetime GAP’s, i.e.
the status must be INBLI. A transition is also unique if there
is a single valid transition involving a non-zero GAP, i.e. the
status is INGAP and the activity set of the next BLI is known.

BLID2 takes the terminating BLI Class (X) and its termi­
nating segment (r) and examines the transitions possible from
X which include r (i.e. the valid transitions). If a unique
transition is found then all segments in memory which are not
part of the transition are removed. If the transition is direct to
a BLI then the Memory Occupancy Set is the intersection of
all valid BLI’s for that transition. If transition is to a GAP then
the memory occupancy set is the GAP membership and
BLID2 will monitor references until a reference is found
which does not belong to the GAP, whereupon any segments
not belonging to BLI Class Y are then removed. If BLID2
cannot isolate a unique transition from BLI Class X and
segment r then no action is taken to remove any segments.

**BLID3** carries out a similar decision process to BLID2
with one enhancement. When a unique transition is found
which involves a non-zero GAP (i.e. to a unique BLI Class Y),
then the expected lifetime of the GAP is tested so that the least
cost way of entering the next BLI (Y) can be found. If the GAP
lifetime is relatively short then any segments from BLI Y
which are currently in memory are not excluded and the
Memory Occupancy Set is the union of the GAP membership
and BLI activity sets for the transition. GAP references are
still monitored to find the End of GAP so that the GAP
segments which are not in the BLI activity set can then be
excluded.

If the GAP lifetime is long enough to draw Space-Time
benefit from doing so, the transition is carried out in two
stages:
(i) To exclude segments not referenced in the GAP, and then
(ii) When the end of the GAP is detected, to exclude those
segments not referenced in BLI Class Y.

**Definition: The Terminating Set**
The terminating reference concept is extended to a terminat­
ning set, i.e. the set of segments referenced since the BLI of
Class X terminated.

BLID4 delays the final transition decision until a unique
transition is found. Once a unique transition is found then the
BLID3 operation is carried out.

The “Adjust Memory to GAP Set” process is as described
for BLID3, i.e. the expected GAP lifetime is used to predict
the lower cost passage into the next BLI.

**3.3 Algorithms BLID5 and BLID6**
Algorithm BLID5 is a further extension of the Category 1
Demand Paging Algorithms mentioned above. BLID5 uses a
different method for the isolation of a unique transition.
BLID5 returns to the use of terminating segments, i.e. the
terminating set segments are used to eliminate valid transi­
4 THE CATEGORY 1 PREPAGING ALGORITHMS

As with the category 1 demand paging algorithms, these algorithms use BLI Transition Model information which has been obtained during a previous program execution. These algorithms, however, do not restrict memory allocation to those segments which have caused a page fault, but attempt to predict the future referencing of segments, where possible, and to transfer these segments into memory beforehand so that when they are referenced a page fault does not occur. The algorithms are given the code letter “P” (Prepage) for identification purposes, and are called BLIP1 and BLIP2.

The Space-Time Cost calculated is the true value, with the current memory occupancy at any time taken to include all prepaged segments and also any segment which is in the process of being prepaged. The cost benefit of prepaging comes about because normal processing does not need to be suspended for the transfer to take place as is the case with the transfer of a segment which has caused a fault. The transfer cost for prepaging a segment is \( a \) (the transfer time) because one extra memory page is “frozen” during the transfer, rather than \( aN \) (where \( N \) is the amount of memory occupied) for the service of a normal fault, during which time the complete memory occupancy is “frozen”.

The BLIP algorithms do not allow either prepaging to take place during the service of a fault, or a fault to be serviced until any current prepaging transfer is complete. If a fault occurs during a prepaging transfer, then a “partial fault” occurs, whereby the remaining part of the prepaging transfer is costed at the normal “time x memory occupancy” rate. This is necessary because the prepaging transfer holds back the service of a demand for a page, thus “freezing” the entire memory allocation in the same way as does an ordinary fault. The fault for the demanded page is then serviced.

When a BLI (of class X) terminates a decision process similar to that employed in the BLID6 algorithm is used to decide which segments should be removed from memory in order to give the best Space-Time Cost. If there is a GAP between the BLI just terminated (X) and the expected BLI (Y) then the lifetime of the GAP is tested (as in BLID3) to determine whether passage to BLI Y should be done in two stages (through the GAP firstly and then to BLI Y) or whether the GAP and BLI Y should be taken together when calculating which segments should be removed.

The expected lifetime of the next phase of execution (whether a BLI or a “long” GAP, as described above) is obtained from the BLI Transition Information. If the next phase is a GAP of long enough lifetime, then the segments needed for the next BLI (Y, which follows this GAP) can be scheduled for prepaging. These segments will be those which are not expected to be in memory at the time that the GAP finishes, i.e. the segments which belong to BLI Y but do not belong to the GAP between X and Y and are not in memory after X terminates.

If the next phase is a BLI (Y) or a short GAP followed by a BLI (Y), then the segments which can be expected to be needed when BLI Y terminates can be scheduled for prepaging. The segments prepaged are restricted to those for which we can be sure that they will be needed, and that they will not in any case be brought into memory for the execution of Y. They must therefore belong to all possible transitions from BLI Class Y and be expected to not be in memory when Y terminates. This scheduling process can also be carried out for BLI Y in the “long GAP” case outlined above.

No attempt is made to prepage segments at other times because a BLI has just terminated and so execution can be expected to enter a transition phase where faults will occur more frequently. If the GAP between two BLI’s has a short lifetime then faults brought about by being within the GAP are expected to occur too frequently to allow the use of prepaging. The strategy just outlined is implemented in algorithm BLIP1.

BLIP2 differs from BLIP1 by preventing prepaging in other circumstances which are expected to provide little or no benefit from attempting to prepage segments. If execution is within the Head of the new BLI (building up the new activity set) then BLIP2 prevents prepaging. Also, BLIP2 uses the expected BLI lifetime standard deviation to decide if prepaging is likely to be effective. If this standard deviation exceeds a specified limit then prepaging is not attempted. The basic tenet for both BLIP1 and BLIP2 is that if prepaging is to provide benefit then it must be carried out during stable execution phases such as the Body or Tail of a BLI, or perhaps also during a GAP which has a lifetime of sufficient duration.

Prepaging is carried out by scheduling a virtual time at which the transfer of a segment is to begin. Once loaded into memory, a prepaged segment is given an expiry time (equal to some constant times \( a \)) so that if it is not referenced by this time then the segment is deallocated. In this way the incorrect or injudicial prepaging of segments is prevented from having too great an affect on the Space-Time Cost. Prepaging is also limited to one segment at a time, so if more than one segment is to be prepaged for a particular virtual time (T) then the first must be scheduled to begin transfer at T-\( a \), the
The more sophisticated forms of the BLI Transition Model previous executions of the program is used in controlling the category 2 algorithms are demand paging algorithms (as are the BLID algorithms from Section 3), but differ from the algorithms already presented in that no information from previous executions of the program is used in controlling memory allocation and deallocation. BLI Transition Model information is gathered during the current program execution and it is this information which is then used. These algorithms are identified by the code letter “Q” and are called BLIQ1 and BLIQ2.

The BLI Transition Information is built up during execution according to the ‘A’ macromodel (section 1.3), i.e. BLI’s are classified according to their Activity Sets alone. The more sophisticated forms of the BLI Transition Model cannot be used because the position of any BLI within the BLI hierarchy is not known until execution has ceased, and this hierarchy position is required for the other macromodels.

The information building process simply keeps data from the last BLI to terminate, and when another BLI terminates then a transition is recorded between the two. Knowledge is also needed of all segments referenced since the last BLI terminated so that the GAP membership (disjoint to both BLI’s) can be obtained. Because reprocessing of the reference string cannot be done in order to gain the GAP information, the complete GAP membership set cannot be obtained. It is impossible to tell which (if any) of the segments from either of the two activity sets of the BLI’s also belong to the GAP and so only the segments referenced in the GAP which do not appear in either BLI activity set can be gained. The correct lifetime of the GAP is obtainable, however, because both of the BLI’s have start and finish times which are known.

The transition matrix resulting from the “on the fly” building process will have a different structure to that gained by the direct analysis of a reference string by the ‘A’ macromodel (as in section 1.3) because the BLI’s arrive, ready to be incorporated into the matrix, in finish time order.

![Figure 7: BLI Hierarchy and Arrival Order for the BLIQ Algorithms. BLI’s will arrive in finish time order 7,3,4,1,5,6,2.](image)

Transitions are classified into three types (Maddison and Batson, 1976; Hannaford and Blatt, 1987) dependent upon the relative levels of the BLI’s in the hierarchy. Type 1 transitions are those dropping to a lower level, type 2 remain at the same level and type 3 transitions go to a higher level. Type 1 transitions (1 to 3, 3 to 7 and 2 to 5) will not appear in the transition matrix for the reference string of fig 7. Type 3 transitions will appear unchanged (7 to 3, 4 to 1 and 6 to 2). Ordinary type 2 transitions will also occur as before (3 to 4 and 5 to 6). Type 2 transitions may also occur across levels, e.g. transition (1 to 5) above. The type 2 transition from BLI 1 to BLI 2 in the above figure is not observed because BLI number 2 does not terminate until after number 5.

Algorithm BLIQ1 carries out memory adjustment decisions in exactly the same way as algorithm BLID6, the last of the Category 1 Demand Paging Algorithms. The only changes between BLID6 and BLIQ1 are in respect to the data collection of BLI termination events and transitions between BLI’s. One part of BLID6 not mentioned previously which now becomes far more important is the procedure followed when there is no known transition from a particular BLI. This is not so important when previous run transition information is available because most BLI’s occurring within one program run can be expected to occur in a subsequent run. This, of course, is dependent on the stability of the program’s behaviour as input data changes. In BLIQ1, however, no previous run information is available and so the first occurrence of any BLI will result in there being no known transition through which the program execution can be expected to pass. The decision made within BLIQ1 (and also BLID6) is to reduce the memory occupancy to the minimum level, i.e. memory will hold only the terminating segment for the BLI that has just finished.

BLIQ2 has only minor modifications to BLIQ1. BLIQ2, rather than emptying memory when there is no known transition for a BLI, leaves the memory occupancy unchanged. BLIQ2 was developed to investigate the impact of the building up of the memory occupancy during the early parts of a run on the overall Space Time Cost.

By comparing results from these two programs it should be possible to see the relative effects of memory occupancy and page faults on the Space Time integral. BLIQ1 can be expected to give more faults than BLIQ2 because memory occupancy is reduced whenever the first BLI of a particular class terminates, but these extra faults will, because of this decision, occur at a much lower memory occupancy level. BLIQ2 looks to the converse situation, less faults, but the faults which do occur will occur at a higher memory occupancy level. The differences in memory level will affect the relative Space Time Costs due to execution, BLIQ1 having an expected advantage. Using the Space Time Integral allows a comparison such as this to be carried out. A higher fault rate may be more economical in the use of memory provided the memory level can be reduced enough by allowing the extra faults.

6 RESULTS AND DISCUSSION ON THE BLI BASED ALGORITHMS

The algorithms described in the preceding sections were applied to the complete set of 39 ALGOL reference strings (section 1.2). Also the algorithms were applied to the very
Table 1. Performance of the BLID Algorithms.

<table>
<thead>
<tr>
<th></th>
<th>( W_S^{opt} )</th>
<th>( LRU^{opt} )</th>
<th>BLID1</th>
<th>BLID2</th>
<th>BLID3</th>
<th>BLID4</th>
<th>BLID5</th>
<th>BLID6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Strings</td>
<td>40</td>
<td>40</td>
<td>40</td>
<td>40</td>
<td>40</td>
<td>40</td>
<td>40</td>
<td>40</td>
</tr>
<tr>
<td>Within 10% of PFC</td>
<td>13</td>
<td>4</td>
<td>3</td>
<td>15</td>
<td>16</td>
<td>18</td>
<td>14</td>
<td>13</td>
</tr>
<tr>
<td>Within 25% of PFC</td>
<td>28</td>
<td>17</td>
<td>7</td>
<td>25</td>
<td>26</td>
<td>25</td>
<td>16</td>
<td>18</td>
</tr>
<tr>
<td>Better Than ( W_S^{opt} )</td>
<td>—</td>
<td>10</td>
<td>4</td>
<td>20</td>
<td>21</td>
<td>21</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>Within 5% of ( W_S^{opt} )</td>
<td>—</td>
<td>13</td>
<td>7</td>
<td>27</td>
<td>28</td>
<td>27</td>
<td>17</td>
<td>20</td>
</tr>
<tr>
<td>Better Than ( LRU^{opt} )</td>
<td>30</td>
<td>—</td>
<td>13</td>
<td>32</td>
<td>33</td>
<td>33</td>
<td>22</td>
<td>22</td>
</tr>
</tbody>
</table>

6.1 The Category 1 Demand Paging Algorithms (BLID)

Results for the Category 1 Demand Paging Algorithms (BLID1 to BLID6) are presented in terms of their performance with respect to the PFC algorithm (Hannaford and Blatt, 1986) and with respect to \( W_S^{opt} \) and \( LRU^{opt} \) (which appear as “WS” and “LRU” in Table 1). The measure of performance is the Space-Time Integral (STI). The results presented are for the number of reference strings which satisfy each of the criteria listed in column 1 of the table.

It can be seen from these results that the most efficient BLID algorithms are BLID2, BLID3 and BLID4, and that these algorithms give performance which is better than the \( W_S^{opt} \) results in about half of the reference strings. The worst results are for BLID1 which was an extremely rudimentary algorithm, only developed as a first attempt in order to gain some insight into the ways that the BLI Transition Model might be used effectively. We shall ignore BLID1 in the discussion from this point onwards.

All algorithms (apart from BLID1) outperformed \( LRU^{opt} \) by a significant margin. The performance of the BLID algorithms also is not dependent on the amount of processing done in attempting to use BLI Transition Information. The best results are obtained for BLID2, BLID3 and BLID4 and these are the simplest of the algorithms which use the terminating segment in making a decision. BLID5 and BLID6 attempt to implement more intricate methods of isolating possible transitions, but these algorithms yield poorer results.

These results at first glance show WS in a very favourable light. However, a number of points must be stressed when comparing \( W_S^{opt} \) with BLID2, BLID3 and BLID4.

(i) The WS results used are those gained for the optimal window size for each reference string, considered individually. If a single “optimal” window size was chosen and applied to the complete set of reference strings then the BLID results would appear much more favourably because the BLID algorithms have only one system parameter, viz. the secondary memory speed \( \alpha \), and results such as those above will be gained from the application of the algorithms in all cases since no program parameters need be supplied. When a single WS parameter of \( 5\alpha \) was used for the WS algorithm, BLID2, BLID3 and BLID4 produced better results on 28, 29 and 30 of the 40 strings respectively. The 5% comparison figures for WS in this case were 31, 32 and 33 respectively.

(ii) The optimal WS window size varied widely, from less than \( \alpha \) to around 10\( \alpha \). The Space-Time Integral for WS was found, in many cases, to be very susceptible to small changes in WS window size, especially in the direction of underestimation (String 43 suffered a 40% increase in STI if the WS parameter was underestimated by only 5%).

As the BLID algorithms only alter memory occupancy when a locality has terminated, they gain ground on WS when transitions between localities are short and sharp, by holding less segments in memory during these high page fault periods. It is an unavoidable consequence of the WS strategy that memory occupancy will be highest during such transitions. WS gained ground on the BLID algorithms by always being able to (eventually) alter memory occupancy to the resident set of a locality. If the BLID algorithms failed to remove an unnecessary segment, then this segment would cause extra costs to the STI until the termination of the next BLI. This suggests the possibility of further investigation into a paging strategy combining the strengths of the two approaches.

6.2 The Category 1 Prepaging Algorithms (BLIP)

Results for the Category 1 Prepaging Algorithms (BLIP1 and BLIP2) appear in Table 2 and these results are presented in the same form as for the BLID algorithms. In addition to the comparisons with PFC, \( W_S^{opt} \) and \( LRU^{opt} \), BLIP1 and BLIP2 are compared with BLID6, which is the demand paging
algorithm which is closest in strategy to these pre-pageing algorithms. This is done so that the efficiency of the pre-pageing part of the strategy may be investigated. NF in Table 2 is the Number of Faults.

The results for these algorithms are very similar and suggest that the differences in their respective strategies may not be important. Both algorithms have results equivalent to the best BLID algorithm results from Table 1. It can be seen that for 3 in every 4 reference strings the BLIP algorithms gave better results than WSopr.

The comparison of BLIP1 and BLIP2 with BLID6 shows that around one quarter of the strings showed a drop in both the STI and the number of faults (NF). On average the pre-pageing reduced the overall number of faults by 16% with the cost of this being an average increase in the STI of 3.5%. Between one third and one half of the strings reduced faults by 15% while keeping the STI increase to less than 3%.

The effectiveness of the pre-pageing scheme used varied widely over the reference strings. For about 25% of the strings pre-pageing was barely attempted. This probably occurred because of a high degree of activity set overlap between adjacent BLI’s which meant that segments which might benefit from pre-pageing could not be isolated. When pre-pageing could be attempted, it was successful on around 56% of the occasions it was used, i.e., the preaged segment was referenced before the expiry date and so the STI of the reference string could be reduced.

6.3 The Category 2 Demand Paging Algorithms (BLIQ)

Results for the Category 2 Demand Paging Algorithms (BLIQ1 and BLIQ2) are presented in Table 3 with criteria and measures being the same as those used previously in Table 1.

These results are significantly worse than the BLID results, but it must be remembered that BLIQ1 and BLIQ2 do not have any prior knowledge of the behaviour of the program. BLIQ1 is still able to obtain results within 10% of PFC for 40% of the reference strings studied. In addition, BLIQ1 uses the decision processes of algorithm BLID6 which is one of the lower performance BLID algorithms. Better results may be possible if the process from either BLID3 or BLID4 were used.

The poorer results for BLIQ2 reflect the added STI cost involved at page faults by this algorithm during the initial stages of the programs’ execution. BLIQ1, upon the discovery of a new BLI Class (with no known transitions from this class) empties the memory occupancy. This incurs extra page faults, but because the memory occupancy is low (as the next locality set is built up) the STI does not rise too quickly. BLIQ2 does not alter memory occupancy when a new BLI Class is discovered. When the next locality is nearly disjoint from the newly discovered class, extra page faults occur anyway, but the memory occupancy is now much higher than for the equivalent case in the BLIQ1 algorithm and so the STI for BLIQ2 rises rapidly.

6.4 Experiments on Previous Run Information

The FORTRAN reference strings, as mentioned in Section 1.2, were all taken from the one run of a program. The strings (numbered 40 to 43) were from a program which iteratively solved a system of non-linear differential equations, completing this task in 6 iterations. Reference String Number 43 was a symbolic array reference trace of the complete program run. The strings, identified as 40, 41 and 42, were for the first iteration, the first 2 iterations and the last iteration, respectively, of the run, and so had lengths which were about 1/6th, 1/3rd and 1/6th that of string 43, respectively (i.e., 40 and 42 each had around 3.4 million references, and string 41 about 6.9 million references).

Strings 40 to 42 provided a means of testing the performance of the BLI based algorithms when information from only part of a previous program run has been analysed. In preliminary experiments the BLI Transition Information from string 40 was supplied to BLID5 “executing” string 41. The STI was
Table 4. Results using String 41 to provide BLI Transition Information for String 43. Amount (%) that the STI exceeds that for PFC on String 43.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Results using String 43 information</th>
<th>Results using String 41 information</th>
</tr>
</thead>
<tbody>
<tr>
<td>WS</td>
<td>3.85</td>
<td></td>
</tr>
<tr>
<td>BLID5</td>
<td>6.92</td>
<td>7.13</td>
</tr>
<tr>
<td>BLID6</td>
<td>1.35</td>
<td>1.51</td>
</tr>
<tr>
<td>BLIP1</td>
<td>1.77</td>
<td>1.92</td>
</tr>
<tr>
<td>BLIP2</td>
<td>2.65</td>
<td>2.66</td>
</tr>
<tr>
<td>BLIQ1</td>
<td>14.59</td>
<td></td>
</tr>
</tbody>
</table>

found to increase by only 1.2% with the number of faults being only slightly higher. The string 40 information was also supplied to BLID6 “executing” string 42, and the STI was found to increase by only 0.5%, with approximately the same number of faults once again.

String 41 was analysed by the BLI Transition Model and the information gained was then used as the input to several of the BLI based algorithms “executing” string 43. Each of the algorithms tested gave results which were extremely close to those obtained when string 43 had been “executed” using its own BLI Transition Information. A summary of these results appears in Table 4, where all figures quoted in this table are for the percentage by which the STI exceeded the PFC result for STI. Results for WS and BLIQ1 are included solely for purposes of comparison.

It can be seen that the degradation in performance caused by the use of data gained from a partial run of the program was as low as 0.03% and rose to a maximum of only 0.21%.

These results demonstrate the versatility of the BLI based (category 1) algorithms. The main requirement for these algorithms to provide a good level of performance is knowledge of the major localities of the program rather than a complete knowledge of the BLI hierarchy. These algorithms will also be most useful in situations where a program is being run very often at which stage the extra processing involved becomes a more economic proposition in order to gain performance improvements. But these experiments show that the amount of preprocessing may be quite small. One example might be a payroll system which could be effectively controlled in a VM system after the processing of a trace pertinent to one “typical” employee.

Another possibility opened by these results is the analysis of a trace from the start of a program which is used as a starting (or initial) condition for the BLI Information Building Algorithms. This would help overcome the high costs incurred by the BLIQ algorithms early in the program run as they build up sufficient information to allow the memory occupancy decisions to be made.

7 CONCLUDING REMARKS

We have presented new memory management algorithms which use the BLI Transition Model. These algorithms differ markedly from previous approaches by concentrating on memory control during the highly volatile transition phases of the program’s execution. The category 1 algorithms show that previous run information can be used to tune the VM control to the behaviour of individual production programs running on the system. Furthermore, knowledge of transitions between localities can be used to anticipate future behaviour and make prepping decisions, as in the BLIP1/2 algorithms presented. The amount of previous run information can be extremely small, as was demonstrated in the experiments of section 6.4.

The major criticism which could be levelled at the BLI based algorithms is in the increased level of processing required to carry out the decision making process. It must be remembered, however, that in the case of the category 1 algorithms most of this will be done prior to a program actually being run, and also (as demonstrated in section 6.4) the amount of preprocessing required may be very low. Also, the BLI based algorithms are only making a memory management decision whenever a BLI terminates, and this can be very infrequently when the program involved has a high degree of locality; e.g. String 43 of 20.7 million references had only 18000 BLI’s and so decisions were being made on average only once every 1000 references or so. The processing complexity question basically reduces to a comparison between the process of finding the BLI’s during execution and current processing techniques, such as, the updating of the LRU stack for the LRU algorithm, or the recording of reference times and recognition of page expiry for WS. While algorithms such as WS and LRU make use of artificial extra parameters such as window size and stack depth, the BLI based algorithms provide a means of efficient allocation of memory to programs in a VM system, the only external parameter being the secondary memory speed.

REFERENCES


**BIOGRAPHICAL NOTES**

*M.R. Hannaford received his PhD in Computer Science from The University of Newcastle in 1986. From 1985 to 1989 he was a member of the Department of Computer Sciences, Purdue University, USA, as a Visiting Assistant Professor, then Assistant Professor of Computer Science. He returned to The University of Newcastle in 1989, where he is currently a Senior Lecturer in Computer Science. His current research interests are: Operating Systems, Distributed Computing and Internetworking.*

*D.W.E. Blatt received his PhD from The University of Sydney in 1974. He was a Fulbright Scholar at The University of Virginia, USA in 1975. Prior to 1990 he was an Associate Professor of Computer Science at The University of Newcastle. He is currently Software Development Manager, Computer Systems Australia Pty Ltd, Lambton, NSW. His current research interests are in Computer Graphics.*
The following books have been received by the Australian Computer Journal. A person who feels able to review one of these books should contact the Associate Editor for Book Reviews via email at acjbooks@durian.citr.uq.oz.au or by fax on (07) 878 2842. Potential reviewers are referred to the announcement of new book review policy guidelines published on ‘aus.acs.books’ on AARNet or in the August, 1992, issue of the Journal.


The following is a list of reviews of books which have been published on AARNet newsgroup aus.acs.books since the last printed issue of the Journal.


This book’s subtitle, “Cooperative Design of Computer Systems” establishes the focus and sociological flavour of the book. The authors’ central arguments are that computer systems are best designed through a strong cooperative process between the design professionals and the system users. An underlying presumption for most of the book is therefore, that new systems are introduced for established staff in existing work-places and functional contexts. Such systems should be designed to enhance the skills of the workers and improve the quality of their results, rather than merely improve productivity.

Furthermore, good cooperative and workplace-based design will find solutions to the inevitable — political and organisational — conflicts which arise in the system design process.

Over thirteen chapters, the editors and their fourteen contributing authors provide an interesting and well-argued case for cooperative design. Rather than presenting a set of narrow “how to” prescriptions, they offer the reader a blend of academic insights and practical experience principally from systems in the typography and information fields. In the first part of the book, and taking perspectives from psychology (here focusing on “human actors” rather than “human factors”), sociology, linguistics and anthropology, the nature of work practice is thoroughly explored. Suchmann and Trigg’s chapter, “Understanding Practice” illustrates how video can be used to observe and assist better the understanding of work practice.

The second part of the book examines the practice of design itself. Designers are placed at the centre of the process: they are the key facilitators of workplace idea generation and have to act as conflict managers; and are the possessors of the technical solutions. They have the ability to use metaphor and fantasy to generate far-sighted, yet feasible ideas. The most applied contributions here are on the use of “cardboard computers”, and other mock-ups and simulations, and cooperative prototyping, using modern graphics software to support rapid changes of presentation. The penultimate chapter discusses adaptive design as the process in which users’ practice requires systems to be re-evaluated and adapted. The book concludes with a reflective chapter in which the process of designing the book itself is given, illustrating how the design concepts presented in the computer systems context can — and cannot — be applied to other situations.

Here then is a good-value, readable, well-referenced and thought-provoking book on largely non-technical, but essential, issues of computer systems design. The approach is unashamedly rooted in the Scandinavian industrial tradition, but is of no less value for that. This book would be useful to workplace professionals who desire a non-theoretical guide to improve their skills. The approach is unashamedly rooted in the Scandinavian industrial tradition, but is of no less value for that.
nature. Most books today are about a prescriptive object-oriented methodology or an object-oriented programming language. The book is based on the author's experience in teaching of training courses and in the software engineering object-oriented context.

There are five chapters as follows:

Chapter 1 — Preamble. This chapter is basically an explanation of where the material comes from and how it will be used. Exhibit 6 identifies six good books and six journals as background readings. The books are a good source of information for clarification and details on the various topics and methodologies that follow in the later chapters.

Chapter 2 — An Introduction to the Object-Oriented Philosophy and Terminology. Some might say that this chapter does not live up to its title. The chapter introduces a wide range of topics in object orientation — none in too much detail or depth. The intent is to first provide a wide perspective of the topics and then in later chapters embellish particular topics. This chapter might leave the inexperienced a bit frustrated.

Chapter 3 — Object-Oriented Software Engineering. This chapter centres around discussing ten (10) software engineering goals within the object-oriented context. These include such topics as correctness, robustness, reusability, integrity, compatibility, portability, ease of use, and maintainability. Within these topics the author discusses such areas as abstract data types and classes, data dependency, and information hiding. Some excellent exhibits that can be used — as the author says — as a "ready-made course supplement" are included in the chapter.

Chapter 4 — Object-Oriented Systems Development. This chapter looks at several software development methodologies at an introductory level. The notations and concepts of these methodologies are presented. A large portion of the chapter rests on the author's own object oriented development methodology. Two case studies are included that use and illustrate the author's object-oriented systems development approach. The approach has a good logical basis and has been referred to in several journal articles during the last year.

Chapter 5 — Some Implementation Concepts. This chapter expands on and builds upon object-oriented concepts from a language and an implementation standpoint. If you are looking at building an object-oriented environment this chapter provides some useful classifications of languages and definitions.

A great deal of ground is covered in this introductory book. As with any book there are unanswered questions. That is as it should be as it would take a rather large volume to cover all the areas, especially to any depth, in the growing world of object-orientation. The book does provide a broad base upon which to build an understanding of object orientation and what is being done by many of the notable authors in the field today. There are simpler books and more up to date books that might provide a less difficult transition into object-orientation, however, they would not provide as broad a base as this book does.

Having read and used portions of this book, as well as many other object oriented books, I felt fairly comfortable about the content and presentation. However, during a conversation with a practitioner I began to realise my perception versus that of others who might use this book. This is not a book that will make practitioners become object-oriented or answer all the questions about object-orientation that they will have. The book concentrates on what object-orientation is about more than how to do it. I would recommend this book be used in a training environment under the direction of a qualified instructor. I believe on its own it can and will confuse an inexperienced object-oriented reader. However, used in a training environment and supplemented by several of the references it will serve as an effective introductory book.


Here we have a monograph which presents (almost) state of the art research in Generalised LR (GLR) parsing and its applications. It contains a selection of papers presented at the First International Workshop on Parsing Technologies (Pittsburg, 1989), GLR parsing, introduced by the editor in 1985, uses a graph-structured stack to cope with action conflicts in the parsing table thereby providing a deterministic parsing technique for a language which is ambiguous or has conflicts not resolvable by LR(k).

The text starts by presenting Tomita's GLR parsing algorithm. The next chapters concern themselves with performance issues, comparing the approach with chart parsing, and complexity. Chapter 4 introduces an improvement to ensure the parser always runs within a time bound O(n^2). Another modification to the algorithm allows null right hand sides of grammar rules and is developed in the following chapter.

How to improve efficiency by a parallel adaption of Tomita's algorithm is then discussed based on an implementation in a concurrent logic programming language. Two additional papers on efficiency follow. Scoring possible grammar rules with a view to early removal of unlikely candidates also speeds up parsing by restricting the search space. Similarly, associating occurrence probabilities with grammar rules enables most likely parses to be investigated early and its application to continuous speech is developed. This algorithm is applicable when the input is uncertain. Correct parsing in spoken or written language is assisted by GLR parsing. Three papers are presented on this topic, one looking at erroneous input, one at parsing noisy input, and the final one at speech recognition.

The monograph achieves in a harmonious fashion the introduction of its reader to the cunning of Tomita's algorithm, to performance issues and improvements and to applications of the parsing methodology in several interesting areas. The reader will need an understanding of bottom-up parsing and some introductory undergraduate mathematics (probability).

Tomita has made available a Common Lisp version of the GLR parser. If, as you study the text, you are excited by a myriad of possible research applications, you will thereby be able to make early progress. Unfortunately, Lisp's syntax is designed for those with keen eyesight. Clearly not an undergraduate text, this book will most likely appeal to a post-graduate class or individuals researching into parsing strategies. The individual papers are mostly well written. I would recommend it as a must for researchers in this area.
support for arrays in various languages provide a broad view of current approaches, and can be recommended to language designers. The paper on Haskell, a new language from the functional programming community, clearly demonstrates the progress (or otherwise) in array support at the language level. Finally, the papers on implementation range from discussions on optimal storage strategies to the exploitation of highly parallel hardware such as the CM-2.

This volume can be recommended to any researcher in array theory, language design, or language implementation, regardless of the paradigm within which they are working not that all the papers are good. Some suffer from lack of detail or purpose, others are difficult to follow due to style or language. Rather as it is an up-to-date overview of array research.

In the preface the editors state "There has been very little communication between the language design and implementation groups in these different communities (functional, imperative). As a result many ideas in the functional programming community have been re-discovered independently". Their aim was to bring together these communities to share their results and help avoid such problems in the future. They succeed in their first goal, it is up to today’s researchers to achieve the second. A good start would be reading this volume.

N. Perry
Massey University
New Zealand


Despite the rather general sound of the title, this book deals with a fairly specific subject, namely the algebraic approach to language theory by means of monoids and varieties. As a result of this approach, some interesting aspects of languages and automata such as context sensitive languages and alternating automata are not treated at all. The subject matter is interesting and may well be useful for courses including only mathematically sophisticated students.

Unfortunately the book cannot be recommended as a reference or a textbook because of its appalling production. It looks as if the text was substantially modified at a late stage when the index and cross references were already complete. The result is that a substantial number of index entries and cross references direct one to a totally wrong page or section or to a theorem which does not exist. This means that that is enormously time consuming and frustrating to try and read any section in isolation. Locating the relevant definitions and previous results has to be done by guesswork and browsing.

J.M. Robson
Australian National University


This book succeeds in describing what restrictions and technical progress need to be made in order to construct predictable real-time systems. It also provides solutions in many of the problem areas. As such it will be useful to both those in industry and academe, although most readers will find only some of the chapters relevant to them.

The book’s opening chapter provides a good introduction to real-time systems for those unfamiliar with the domain. Examples of a Chemical Process, a Power Plant and a Fighter Aircraft illustrate a variety of features commonly found in real-time problems.

Chapter 2 deals with high level languages and their real-time features. Some 20 pages of text are devoted to analysing current and past languages for their suitability for real-time applications. Relevant features of Real-time Euclid (a language developed in part by one of the authors), Ada, and PEARL, are treated in detail in a further 50 pages. The comparisons made between these languages are enlightening and the authors go a long way towards determining what new language features are needed for real-time and, equally importantly, which features must be abandoned.

The book stresses the need for analysability at all levels of the system and chapter 3 is devoted to language independent timing and schedulability analysis of real-time programs. This chapter covers both language analysis and real-time system modelling. The chapter provides a broad survey of scheduling theory and although it omits important recent work in this field regarding rate monotonic theory (e.g. Sha, Goodenough (1990)), its value lies in the very fact that its approach is both different from, and complementary to, the research at CMU & MIT.

Chapter 4 proposes a novel architecture for real-time nodes consisting of a task processor and an interrupt/scheduling co-processor. This idea is fully developed with both proposals for hardware and supporting kernel design. Many areas often glossed over in other works are thoroughly treated such as providing DMA without the unpredictable delays associated with cycle stealing.

The remaining four chapters of the book concentrate on further developing the overall model described in chapters 2, 3 and 4. Algorithms are provided in detail for most of the kernel, including time management. This is refreshing as most English descriptions of such algorithms leave too much ambiguity to assess any claims made about them without significant detective work. Implementation experience with the kernel and language features are discussed, however the proposed hardware was only at the design stage at the time of writing and so less detail is provided here. The schedulability analyser and Real-Time Euclid are also evaluated by writing small real-world applications and running the analyser on them.

This book will be useful as a first survey of the field having many useful references that could be followed up within the reader’s own area of interest. The bibliography is particularly diverse and will be of use to researchers in the field. The authors summarise the overall state of the art well, although they concentrate on the ‘earliest deadline first’ school of scheduling theory. Nevertheless, they provide a good introduction to scheduling analysis, and those wishing to actually perform analysis will be able to do so after accessing the cited material. I would regard this book as essential reading for anyone planning to develop real-time languages. It is also well suited to those wishing to understand the fundamental nature of real-time systems for the first time. The great strength of this work is that it takes a system approach and so answers many of the questions left begging by other works in this area.

Reference

Michael Pilling
Defence Science and Technology Organisation


Why would a book on psychology/philosophy be reviewed in the ACS Journal? The answer is that the emerging field of connectionism encompasses not only cognitive psychology and philosophy but also linguistics and computer science.

As John Tienson points out in the introduction “Connectionism suggests the possibility of an alternative to the conception of the mind on the model of the modern digital computer — the so called computer metaphor — which has dominated thinking about the mind, both popular and philosophical, since the advent of AI in the 1950s.” That is, it presents a possible alternative to the idea that the human mind can be replicated by a computer programme, together with the notion that data structures correspond to mental states like beliefs and desires.

This text comprises 19 papers loosely divided into six areas: overview, connectionism vs classical cognitive science, connectionism and conditioning, does cognition require syntactically structured representations?, can connectionism provide syntactically structured representations? and connectionism and philosophy.

About one third of the papers originated at the 1987 Spindel Conference (an annual conference on philosophical topics held at Memphis State University) on “Connectionism and the Philosophy of Mind”. The remainder of the
papers are as a response to issues raised at that conference or as a direct response to papers.

There are many interesting ideas raised in this book. To cite one example: Cummins & Schwarz (section 2) speculate that some (most?) areas of human knowledge cannot be compiled into an expert system that captures human intelligence. For computationalism is only correct where human performance is rule describable i.e. recursively specifiable. But it is plausible, they argue, that human performance is recursively specifiable only when the domain cognized is itself characterisable by computable functions. If the domain is not recursively characterisable, our cognising of the domain will not be either.

It may be the case that domains that admit of a special science e.g. thermodynamics, are recursively characterisable. But this hardly seems likely for clothing for example. Human beings know a great deal about clothing: how to dress; what to iron; etc. But it does not form an autonomous domain of inquiry. There are many sciences that say something about clothing, but there is no science of clothing. There are no laws of clothing, and hence, no laws of human cognising concerning clothing. And most areas of human knowledge seem more like clothing than thermodynamics. (p70)

It remains unclear whether connectionism can provide a new conception of the nature of mind and if it can what that conception would be. However this is a very approachable book on the subject and I recommend it.

Lawrie Hanson
Marengo Telecommunications


For those looking for a book that is the next logical step from the spate of popular books on Chaos, for example, GLEICK (87) or STEWART (89), this is not the book for you.

This is a collection of specialist papers in the NATO Advanced Science Institutes Series. This ASI — which was also the 20th session of the Seminaire de mathematiques superieures of the Universite de Montreal — was devoted to Fractal Geometry and Analysis.

There are 10 papers, and as the title suggests, they cover the geometry of fractal sets and the analytical tools used to investigate them. The main topics covered include dimension theory, construction of fractal sets, iterated function systems, dynamics in the complex plane, multifractals and applications of fractal geometry to mathematical analysis.

Each of the papers is divided into lectures or sections presenting advanced theorems, problems and aspects of the topics mentioned above.

The mathematics used is final tertiary year or postgraduate level. Some, but not all lectures give examples. All the papers are supplemented by a comprehensive set of references. Also included is an overall index, although this is not overly extensive.

It should be noted that 2 of the papers “Interpolation fractale” (DUBUC) and “Produits de poids aleatoires independants et applications” (KAHANE) are in French. Moreover these are both lengthy works, comprising 148 of the 469 pages in the general body of the text.

In summary, this is a good reference for the advanced student with a strong mathematical background. For those who would like an intermediate text that is a little more accessible I recommend PEITGEN & SAUPE (88).

References

Lawrie Hanson
Marengo Telecommunications


TEX (which the author reminds us should be pronounced “Tehk”) refers to a family of typesetting programs, designed to produce books or research articles, primarily in the mathematical and physical sciences. The basic language is often known as PLAIN TEX, and this is the subject of Borde’s book. Most Australian academics seem to prefer other TEX packages, such as AMS-TEX (for preparing research papers in the style of the American Mathematical Society) or LA-TEX (for typesetting books), but these are quite understandably not discussed by the author.

The strength of the TEX language is that it is, for the most part, a simple program to use, and can be learned quite quickly by example. For this reason, Borde’s book should represent an effective and enjoyable way for a beginner to pick up at least a working knowledge of the language. The book is written in an entertaining and sometimes humorous style, and it has the rather novel feature that only the pages on the right-hand side actually contain the text material. The left-hand pages display the TEX commands required to produce the text, along with some explanatory footnotes, and so the reader can see at once what effect is produced by the various TEX commands.

Consistent with the theme of learning by example, the author wisely avoids a technical or theoretical discussion, and instead moves quickly into a sequence of 17 mini-chapters, referred to as “Examples”, each of which focuses on a particular feature of the TEX program. A reader possessing a mathematical background would have no difficulty understanding and using the material in the first several of these examples, and this would probably be sufficient for the preparation of most technical reports and papers. Later examples show the reader how to perform more elaborate tasks in TEX, such as defining new commands, programming in TEX, and placing text within frames and boxes, but these tasks would surely be beyond the reach of most beginners. Nevertheless, the advantage this book provides is that it would be possible simply to copy Borde’s examples, altering them as desired, and so to perform sophisticated TEX programming without necessarily understanding every detail of what had been done.

Although the book is aimed in the first instance at the beginner, there is also much to interest a more experienced TEX user. This is reinforced by a reasonably extensive Appendix, in which the need for humour has also not been overlooked. Thus the entry under “TEX” says: “For more information on the topic, read the rest of this book.” In addition, the rather cumbersome task of typesetting tables in TEX is made much easier by copying the code given in the Appendix. An engaging introduction to producing tables is provided in Example 6, in which Borde, a native of India now living in America, pays statistical tribute to the great Indian cricketer Sunil Gavaskar. It is hard to judge how this delightful chapter will be received by its American audience, but the author’s extensive knowledge of the game of cricket leaves this reviewer to ponder over his possible relationship to another Indian cricketer, the all-rounder C.G. Borde.

L.K. Forbes
University of Queensland


Since Lotfi Zadeh first advocated fuzzy sets in 1965 there have been thousands of articles on them and their applications in systems and multivalued logic (fuzzy logic), but very few English language books. We are now in the third wave of activity in the subject, prompted by the huge commercial success the Japanese are currently having with domestic products. Who has not seen, heard of, or purchased their fuzzy logic cameras or washing machines, for example? Academic Press have provided a translation of the 1987 Japanese language book that must have been influential in producing this success.

The three authors are well known in the field and have written a book that starts with a description of fuzzy sets and fuzzy relations. For the novice reader these are as good as I have seen. This is followed by five chapters on fuzzy extensions to regression models, statistical decision methods (including fuzzy-Bayes), quantification theory (based on Hayashi’s idea for dealing with qualitative data), mathematical programming and evaluation methods to set the systems theory ground for the reader. The next eight chapters cover practical applications in diagnosis, control (the source of domestic appliance applications), human activities (multistage decision making with fuzzy dynamic programming), robots, image recognition, databases, information retrieval and an expert system for damage assessment (including an extension of Dempster-Shafer theory to fuzzy sets). Some of the later chapters are a bit...
BOOK REVIEWS


Regression analysis is the art of fitting curves and surfaces to experimental data. Thus it finds many natural applications in model building in the sciences, economics and social sciences. Indeed, regression analysis is one of the most useful fields of mathematics. Every major general purpose statistical package has regression analysis facilities and many hand calculators have a regression button.

Supporting this wealth of applications are two sub-fields of knowledge: (1) the statistical theory of regression and (2) a blend of mathematics and computing knowledge which is used to develop regression facilities in computer packages. This book deals with this second sub-field.

The main part of the book deals with linear regression analysis using various Lp norms with special emphasis on the cases where p is 1, 2, or infinity. The rest of the book deals with more specialised topics which include robust regression, ridge regression and linear regression with constraints. Nonlinear regression methods are outside the scope of this book.

There is no discussion of statistical aspects of regression. As the title suggests, the book concentrates on numerical aspects of algorithms for linear regression.

The text contains many data sets, FORTRAN 77 programs and output from these programs: these are used to illustrate the algorithms. Reading output sometimes requires referring to the description of the data set which is given in Chapter 1, referring to the actual data set which is given in the Appendix and, of course, referring to the output itself which may be in any chapter. In this sense, the book is not very user-friendly. More effort should have been placed on the arrangement of material in the book.

There is a substantial bibliography scattered throughout the book. It would have been better to place all references at the end of the book because then it is easier for the reader to locate them.

The book is a translation of the German edition with a few improvements. It appears to be a text which is written around the many programs that the author has developed. Thus, while the expert with an interest in statistical computing may be enticed to look at the book, the novice will not be further attracted to the field by this book.

Prerequisites for reading the book would be a knowledge of linear algebra, numerical analysis, linear regression analysis and FORTRAN 77.

Like all books in this series (the black covered books in the Academic Press series "Computer Science and Scientific Computing"), the book is well bound and nicely printed. The price is reasonable.

In summary the book would be a useful addition to any university library.

Research workers in statistical computing or numerical analysis should be aware of the book. Academics who teach scientific computing will find in this book useful ideas to incorporate in lectures and assignments. Those who teach statistical methods will find a large number of data sets which would be useful for illustrations or assignment questions. For example, the illustrations used by the author show that there is no such thing as the "line of best fit" — it all depends on the norm. This is a useful fact for all users of statistics to appreciate!

T.M. Mills
La Trobe University College of Northern Victoria


These collected papers dedicated to Seymour Ginsberg on the occasion of his 64th birthday are very mixed in their subject matter, in their level, and in their age.

The subject matter ranges from theoretical properties of context free languages, through data base query and modification to an empirical comparison of polygon clipping algorithms.

The level ranges from the familiar easy to read tutorial style of Donald Knuth (on a study of context free multilanguages planned to be included in Volume 11 of The Art of Computer Programming when that is published in 2008) to a very dense presentation of new material in several other papers which are sometimes made even harder to follow by poor proofreading.

Some of the papers describe recent research whereas others are old publications reprinted or old unpublished work dusted off for the occasion. For instance, in one of them the most recent citation is to a 1971 publication.

There are certainly some interesting new results to be found here, notably (from my personal perspective) in the papers by Harrison on methods for proving that a language does not lie within certain subsets of the set of context free languages, and the one by Rosenthall on the complexity of deciding properties of context free languages.

Since all the worthwhile material in this collection will be published elsewhere in a more polished form and since it does not constitute a coherent account of current research in any clear subject area, I imagine that only libraries will wish to purchase this book.

J.M. Robson
Australian National University


Formal Concepts is a text in artificial intelligence methods, aimed at the later undergraduate years. It concentrates in four areas:

- computational logic (theorem proving);
- natural language processing;
- expert systems;
- searching of state space.

Its coverage of computational logic and state space searching seems to be quite comprehensive. The chapters on natural language processing give a comprehensive grammar of English and a few paragraphs on scripts and frames. The exercises seem to expect that with this material the student can construct fairly sophisticated story interpretation programs. I am not a specialist in natural language processing, but I would be extremely surprised if with the methods presented a student could write a program which would parse a sentence with a usefully small degree of ambiguity. The chapters on expert systems consider only propositional and probabilistic systems: no mention is made of concepts like instantiations or the RETE matching algorithm. The probabilistic reasoning is fairly comprehensive.

The problem with natural language processing exemplifies what I consider the main weakness of the book: there is no evidence that the author has ever written a program in anger, nor that he expects his students to. There are taxonomically comprehensive collections of algorithms, but no discussion of strengths and weaknesses nor any guidance as to when to use which one. A particularly glaring instance resulting from this weakness occurs in the chapter on probabilistic reasoning. He states that the joint probability of a combination of events is often difficult to compute if the events are dependent, and advocates the use of fuzzy logic combination rules to estimate the

THE AUSTRALIAN COMPUTER JOURNAL, VOL. 25, No. 2, MAY 1993 87

This book is a volume in NEURAL NETS: FOUNDATIONS TO APPLICATIONS edited by S.F. Zometzer, J. Davis, C. Lau, and T. McKenna. It contains twenty-two chapters written by leading researchers in neuroscience, biophysics, and bioengineering. The book is divided into four sections: Computation in Dendrites and Spines; Ion Channels and Patterened Discharge, Synapses, and Neuronal Selectivity; Neurons in Their Networks; and Multistate Neurons and Stochastic Models of Neuron Dynamics. The whole book is well organised and presented.

Although the book emphasises the computational aspect of single neurons, it covers the network level as well. Artificial neural network people should find this book useful and inspirational, let alone neuroscientists and biophysicists. In particular, several chapters in the book, e.g., Chapter 4 and 12, discuss some of the significant differences, e.g., the spatial, the temporal, and the probabilistic difference, between the artificial neuron and the biological one. A better understanding of such differences would certainly help computer scientists and engineers to design new generations of artificial neural networks.

This book is mainly written by neuroscientists and biophysicists, but computer scientists and engineers won't find it difficult to understand most of the material. All the basic ideas are presented clearly. The twenty-two chapters can be read separately according to the reader's interest. In summary, this is a very good book for people who are interested in neuroscience, biophysics, and artificial neural networks, although some artificial neural network people might find that there is too much material on neuroscience and biophysics in the book. It would certainly be useful to have a copy of the book in your organisation's library if you can't find a place on your own bookshelf, among those computer science and engineering books.

Xin Yao
Australian Defence Force Academy


This is a very good introductory book to image pattern recognition and machine learning. It can be used as a textbook in machine learning, image pattern recognition, or advanced AI courses. The author has done well in combining the two subfields in AI together by treating both recognition and learning as transformation of representations.

The book begins by discussing the representation issue in the first three chapters. Then it presents feature extraction and pattern understanding in chapter 4 and 5 respectively. The rest of the book, from chapter 6 to 10, is devoted to learning. The topics covered in these 5 chapters include both symbolic and subsymbolic approaches to machine learning, like learning by neural networks. But the topic of learning by evolution systems, etc., is not mentioned.

Each chapter in this book is very much self-contained so that selective reading can be made for those who are only interested in certain topics. Algorithms and their applications are described clearly in the book so that readers can develop their own implementation of these algorithms easily.

This book was used as a textbook on pattern recognition and machine learning in the author's own university, Keio University, in Japan. The exercises after each chapter will be a very useful aid in helping students understand the material in the book. Although the book was first published in Japanese in 1989, it is still a good textbook for introductory pattern recognition and machine learning course. It is very easy to read as well. Third year undergraduate students and graduate students should have no problem in understanding the text by themselves.

Xin Yao
Australian Defence Force Academy


This book comes from the authors of "Learning Turbo Pascal" and develops concepts from that text. The book is divided into two parts. Part I contains seven chapters, two of which show the reader how to use the system and user defined units that are peculiar to that Pascal dialect. The remaining five serve as an introduction to data structures, covering records, files (including direct access files using Turbo Pascal's seek procedure and filesize function), recursion and recursive data structures — stacks, queues, lists and trees. Part II consists of three chapters discussing some applications in Turbo Pascal-program validation, file maintenance and reporting programs and techniques for sorting and searching files.

A five page appendix attempts to give the reader an outline of Object-Oriented Programming and the use of Objects within Turbo Pascal. A second appendix contains solutions to the end of chapter exercises.

The book contains a poor and quite idiosyncratic index and has neither a bibliography nor end of chapter references. There is quite an amount of program code contained in the text; this is quite clear and in most cases well laid out.

In the Introduction, the authors claim that this book would be a useful reference for undergraduate computer science students and for those undertaking diploma or certificate courses. I am afraid that I see it falling well short of the needs of Australian students at these levels. It may be useful to someone wanting a non-theoretical text on data structures that uses Turbo Pascal as its illustrative language, but it certainly lacks any explanation of the theoretical underpinning that I would expect computer science students to learn. In addition to this, the authors state that the reader need not be familiar with their previous text, however they have a rather annoying habit of referring to it throughout the book.

I found this book quite disappointing and certainly could not recommend its purchase for any reason except as an illustration of how a book like this should not be written. It contains a number of types and instances of poorly used terminology (e.g. records being referred to as directories), many of the explanations were quite poor and some of the diagrams dealing with pointers were quite misleading. Many topics are presented with very little theoretical background or even references to where this could be found. For example a program is presented for quicksort but there is no discussion of its pitfalls or indeed how it works. There are no references to where further information about this important algorithm can be found.

There are quite a number of texts that do a far better job of covering this material and some use Turbo Pascal in the programming examples — my advice is to look around and spend your $55 on an alternative.

Andrew Wenn
Victoria University of Technology (Footscray)


This is a non-technical textbook emphasising the organisational aspects and conceptual representations within an informal framework. The reader should have some basic (very basic) knowledge of information technology and computing. This book would best be used as an introductory textbook in the second or third semesters for an undergraduate course of study.
There are 15 chapters divided into four (4) sections as follows:

1. Introduction
2. Database Approach
3. The Database Environment
4. The Approach in Action

Sections 1 and 4 cover three chapters. These sections are used to support some of the material presented in the other two sections. Sections 2 and 3 address many topics (not all) and none in any depth.

Section 2 covers such topics as: business analysis, conceptual schema, logical schema, physical schema, and implementing applications. I found the business analysis topic interesting with some interesting perspectives, but it delivered nothing new. The material on conceptual and logical schema design is very basic and provides a broad overview. These chapters would help students to gain some understanding and knowledge about these topics, but are insufficient in teaching students how to do a conceptual or logical schema design. The physical schema (file access and organisation) is nice information but not necessary in the book’s overall theme. There is a place for this material in today’s computing but not in this type of textbook. The implementing applications chapter includes a basic overview of Multiview. This material is supported and extended using an application of Multiview in section 4. I found it interesting but the material was insufficient to actually be able to use Multiview in a development environment.

Section 3 discusses and gives some basic examples of different database management systems, such as IMS, IDMS, DB2, Adabase, Oracle, DBaseIV, and OrmisV. There is a basic introductory chapter on SQL using DB2 dialect. There is a chapter on the distributed databases. This chapter talks about the fundamental principles of distributed databases, five distinct levels of distribution, and six components of distributed database management systems. The distributed database chapter looks at some of the basic architecture and implementation issues but does not go into depth on any issues, such as modelling distributed information systems within a database approach. There is a brief chapter on tool support, 4th generation languages, CASE, and one on data dictionaries. A useful overview on the use of tools is also provided. I found this chapter difficult to concentrate on because of lack of detail. I had two other people look at the material and they also found it difficult to read. I am not sure whether it was the type face, the point size, the paper the text was printed on or some external factor.

Overall this is an introductory textbook There is nothing technical about the book and there is very little in the way of “how to do it” in the book. If you are looking for a book that just tells about information system development with the database approach thrown in, this book might fit the bill. There are other books on the market that do not cost as much, deliver the material just as well if not better, and are easier to read. I would not spend $60.00 for the book nor would I expect students to spend that much.

Ric Jentzsch
University of Canberra


This book consists of fifteen chapters contributed by eighteen authors with considerable experience in widely different aspects of parallel computation. Each chapter is essentially an up to the minute review of that topic and provides valuable information and reference material for further reading. It is most unlikely that any reader will wish to read every contribution due to the diverse nature of the material presented. In general the presentations are at an advanced level, and persons wishing to acquaint themselves with a new topic may be well advised to seek an introduction from other sources. Nevertheless it is possible to work through an unfamiliar topic provided the reader is willing to exert considerable effort.

Due to the relatively short nature of each contribution (about 20 pages) the particular interests of the author often show in the material presented. Consequently, the material presented cannot be regarded as summarising all existing knowledge in the topic areas.

The authors come from diverse backgrounds and this is reflected in the material presented. For example, some of the articles are reasonably applied in nature and represent the parallel world as it is largely practiced. Other articles reflect the world as theorists would like to see it practised. Consequently, there tends to be a gap, particularly at the level of computer languages. The book has a good presentation on topics such as functional languages, divide-and-conquer paradigms but very little is said about the changes taking place in FORTRAN for the parallel world. An exposition on this topic may have encouraged the ordinary practitioner to read the other series of articles; its omission may well send the message that the computer scientists are at it once again. This is most unfortunate for the book has much to offer that class of practitioner.

I found the book most rewarding and enjoyed spending considerable time in several chapters in which I had previous experience but had not kept current. The book is highly recommended for similar readers and for those contemplating taking a serious interest in any of the fields covered.

Jerdar Barry
Australian Supercomputing Technology


The title of this book should have been Designing Screen Interfaces In C in Seven Easy Steps. I found that the author designed the book so that it was easy to read and comprehend. The added bonus was the fact that the source code examples were straightforward.

The book is divided up into seven chapters. Chapter 1 covers display adapters that obviously plays a large part in the book. This discusses the various types of graphic adapters, text attributes directly accessing the video buffer and utilising interrupts. Chapter 2 discusses windows and screen functions, creating libraries and command line switches. Chapter 3 takes the reader through different menu designs discussing Pop Up, Pull Down and bar menus. Chapter 4 explores data input screens complete with field editing features.

Chapter 5 describes how to develop list selection functions with both point-and-shoot and speed search features such as utilising a preselected character to make the selection. Chapter 6 expands on the previous chapter by using directory searches and allowing the user to select a file from a given directory. Finally Chapter 7 covers the most important area for the user, Help Screens. The author creates a help system that is linked to the application and reads the information from disk.

I believe the book would be quite useful to anyone interested in creating an application using windowing techniques without having to purchase an expensive third party package to do similar functions. As well as being useful to those who are merely interested in how the application that they may be using or similar applications actually utilises hardware (etc.) to perform what may seem to be minor tasks.

In summary, I found the book to be worth the time spent reading it, although it may not be quite suitable for the advanced readers. Andrew Iggleden
Melbourne


This work is a very detailed cookbook for producing a large complicated ODSS. It attempts to cover every aspect, ranging from the traditional activities, such as data modelling and model design, to less often discussed topics (out usually just as important) such as the reality of organisational politics. And I think it is not a bad attempt. While essentially a reference work of the theory of ODSS building, it is illustrated using the authors’ experience of a ten year project to deliver such a system. The system was to assist the US Air Force manage all aspects of its personnel (training, remuneration, promotion, retirement), with a view to providing the required number and mix of staff at all times. This is known as the Enlisted Force Management System (EFMS).

The approach adopted is to alternate discussion of theory (with significant references to other publications) and presentation of their case study. I found the theory to be quite good, and with sufficient technical depth to be useful as a “how to” guide. It was also sprinkled with comments authors’ views of pitfalls- to be aware of, and suggestions for good practices to adopt. Some of this advice was in hindsight, so is not as proven as those practices they did employ successfully.
As mentioned, a significant amount of the book is devoted to the authors’ experience in developing the EFMS. If you read this book with a view to truly understanding the lessons in ODSS design and development, then be prepared to learn a significant amount of detail regarding the USAF personnel administration and decision making. The authors warn that this is necessary to appreciate their examples, and I agree that it is essential to appreciate the principles and lessons presented.

I found some of the project discussion quite fascinating, as it not only demonstrates ODSS design, but also many familiar problems encountered with bureaucracy and politics in systems development. In fact, quite a number of the recommendations for staffing and administering the team, lines of reporting, etc., were not unique to ODSS, but to any medium to large scale development project.

I believe it to be quite a useful book, with something of relevance to most people in the industry. While not an easy read, it would certainly make an adequate text or reference book on the topic. Without knowing its price, it is hard to say if it is value for money, but I would reasonably expect to pay around $60.

Peter Cooper
Macquarie Bank, Sydney


Most traditional programming courses and text books don’t have program efficiency high on their priority lists. Programmers are taught to write correct code, and only worry about speed and reasonable memory usage once the application is running. The underlying assumption of this approach is that, if you’re lucky, the program will be fast enough already, and optimisation will not be necessary.

Now, as many application programmers know, this approach is not always a luxury that projects can afford. Speed and minimal memory usage are crucial for a huge range of applications, not only real-time control systems, but also many systems incorporating advanced user-interfaces and extensive data manipulation. Applications can succeed or fail based only on their performance.

It’s therefore quite refreshing to read a book which covers, in detail, optimisation techniques. Although the examples cover only C and C++, many of the fundamental ideas can be carried over and used in other languages. This is especially true of chapter 3, which introduces a number of ways of making algorithms more efficient. Other topics covered include performance measurement tools, code transformations space-efficient techniques and some useful hints for reducing overheads in C++. All this information is nicely rounded off with a section on example applications, and, for the mad amongst us, compiler optimisation techniques.

I enjoyed this book. It’s readable, full of useful information and examples which get the message over. If writing fast, small programs in C or C++ is your life, it’s well worth a look, it might save you those crucial few microseconds or bytes one day. My only objection — paying $20 for a disk containing the source code for the examples. This is not necessary!

Ian Gorton
University of New South Wales


Originating in the statistical process control of W. Edwards Deming, ‘Total Quality’ (TQ) is credited with making post-war Japan a manufacturing giant. TQ has been broadened, generalised, adapted and adopted to form a set of performance drive(sic), short-term, innovative with great-leaps-forward, and West. She characterises the US industry as being “results-oriented, setting piece bringing out the differences in corporate approach between East from Japanese TQM Applications to Software Engineering, is a fine scene—the project management and TQM seminar circuit. Her chapter, Learning book applies the principles to Software and its construction.

As society in the 1990s seems to be increasingly applying a corporatist philosophy to deal with the organisation of government, industry and commerce at all levels, TQM has an expanding market. In our own field, trends in software engineering and the management of information has changed dramatically as the technological itself has changed and its applications expanded. TQM offers some methods and attitudes that we will all need to address even if only to refute. Schulmeyer’s book is a good introduction to the issues.

Craig McDonald
Charles Sturt University


According to the preface, this book is intended as an introduction to the subject of Local Area Networks for students of computing and electronic engineering and for professionals moving into LANs for the first time. I believe it misses its objective to both groups.

There is no clear structure. The book jumps from theory to practicalities of hardware design to applications to standards. Nothing is discussed in any great depth.

Perhaps the worst aspect is the appalling proof-reading (or lack of it!). I reached a dozen errors and typos before giving up counting. The errors range from spelling mistakes to incomplete and/or inaccurate diagrams. Pity the student attempting to understand the material being presented.

At the end of each chapter there is an irritating graphic with the caption “Sage the Owl Recommends” followed by a list of terms to review e.g. bridges, routers, gateways. This serves no purpose. It would have been far more helpful if instead there had been worked examples and/or questions to reinforce material presented.

The ideas presented are quite dated (in the context of what is currently available in LAN technology). As an example the chapter on “Futures” covers ISDN and FDDI.

The index is very poor and no references are given. The only time other publications are mentioned, they are works by the same publisher and the author of this work.

The overall impression I reached was that this book has been hastily revised from its original edition and the quality and content have suffered impact. The author dedicates the book to his wife — the “long suffering supporter” — it could equally be dedicated to the “long suffering reviewer”.

Not recommended.

Lawrie Hanson
Burwood, Victoria


The back cover of this book declares that “now there is a complete, detailed guide to effectively and efficiently programming with the powerful Intel 1386/486 microprocessor ... in this one source, you will find unmatched coverage of the enhanced features of the new unit and programming methods that make the most of the 386/486’s exceptional capabilities.” In short, the reader is offered invaluable tips into unlocking the arcane, and utilizing the processor to its maximum possibilities.
Topics that are covered are a brief introduction into the architecture of the 386/486, real-address mode, protected mode, and virtual 8086 mode. The section on protected mode is the largest, with coverage of memory management, entering and leaving protected mode, page programming, control transfers through differing privilege levels, multitasking, debugging, output, and exceptions and interrupts.

Three appendices are included (as well as a glossary and index). These are two sample 486 programs, an instruction set listing, and a brief two line batch file listing that invokes masm and then link, when supplied with an assembly language filename.

I am sure that people with an interest in programming these microprocessors in assembly language, and who already have a good knowledge of such, may find this book to be a useful resource. I am equally certain that those without such familiarity will find no value in this book. The reason I say this is that the bulk of each chapter is an assembly language listing that supposedly demonstrates the feature that the chapter is concerned with. The rest of the chapter is a very brief discussion of the topic, very brief meaning virtually non-existent, and so mostly unhelpful in deciphering the code. I do not mean to sound harsh, but I cannot believe that this book will be clear to anyone but those quite proficient in the realm of 386/486 programming and who take well to reading other people’s assembly language code (this is perhaps the intended meaning of the word ‘advanced’ in the title).

David M. Williams
Tomago Aluminium


This book appears to be quite a departure from previous books I have seen from Academic Press, in that it is not merely an intellectual discourse, but also very practical. In fact, it is quite a complete guide to programming graphics in C (and some C++), particularly for the IBM and compatibles. Numerous example programs are listed, demonstrating the points made in the text, all of which appear on a 5 1/2” disk provided. Several pages are also provided dedicated to the display of 32 colour palettes.

Topics range from the several diverse and brief topics such as programming style and hardware considerations, to plotting points, then lines and rectangles, clipping techniques, drawing and filling circles, ellipses, circular arcs, polygons and rectangles, smooth curves (via Bezier and B-Spline functions), displaying text, rotating geometric figures, solid modelling techniques and then finally saving and restoring images in several different formats.

The book is most comprehensive in its coverage on the topic. Those who have done a Computer Graphics course may well recognise such names as Bresenham’s algorithm — now finally working versions in a form that may be used immediately are provided. For those who have not done such a course, the text covers the theory and mathematics of each topic.

Anyone who is interested in programming graphics in C will find this book a very useful reference for their shelf. Those who wish to do so on an IBM or compatible will find it even more valuable.

David M. Williams
Tomago Aluminium


To a functional programmer such as myself, the title "Language Independent Design" conjures images of a glorious semantic development framework in which one chooses between lambda functions and GOSUBs, between exception handling and GOTOs. Unfortunately crystal palaces of denotational semantics do not sell as well as introductory texts on program design, and it is the latter the author has chosen to write.

The book briefly covers Pascal with particular emphasis on functional decomposition. Once introduced, Pascal is slowly transformed into a form of structure diagram similar to those used in Jackson Structured Programming. Finally, the book demonstrates how these structure diagrams may be encoded in BASIC, with which the student is expected to be familiar. The theme of the book is that by using structure diagrams, one can design programs in a language independent way.

It is not so much the contents of the book which disturbs me so much as the computing curriculum which it implies. To teach students to program in BASIC, then afterwards teach them program design is a fundamentally flawed approach. Structured design techniques should be a fundamental component of an introductory programming course, not an added extra. In any case, JSP appears to be a rather antiquated development tool. Consequently, I do not see a role for this book in any course offered by a self-respecting centre of computer science education.

Furthermore, this edition of the book is plagued with typographical, grammatical, syntactic and factual errors (page 19, "pi = 3.1423"; page 78, "WHILE is generally preferred to REPEAT — it’s safer"). Inexperienced students of programming would consequently not be able to understand the book, and I would not recommend it as text until many errors were corrected. The book is also terse to the point of confusion, and does not state its aims clearly.

In summary I believe that in a suitable course, which must exist at least at the author’s home institution, there is a role for this book. However I also believe that any curriculum in which this book may participate needs to be redesigned. Finally, I would not recommend purchase of this book until more care has been taken in its preparation.

John Farrell
James Cook University


This is a hard copy publication of the proceedings of the Second IMACS International Conference on Expert Systems for Numerical Computing, held at Purdue University, U.S.A, in April, 1990. As such, this book falls squarely in the category of reference text for the local research library.

As the conference title implies this collection deals almost entirely with mathematical problem solving applications of expert systems. In total, 20 papers are included, all of good quality, presenting an up to date view on the state of the art in this field. The papers are divided into the following categories: Mathematical Expert Systems; Invited Talks; Expert User Interfaces; Engineering Expert Systems; and Symbolic and Formal Specification of Modelling Programs.

A common theme running through most of the categories above is the development of expert systems for the purpose of selecting the best from a potentially vast number of algorithms, to solve mathematical problems in specific problem domains. These domains all seemed to concentrate on the difficult task of solving differential equations in one form or another. Every single one of the systems discussed emphasised the boundary between the mechanical application of an algorithm, and the "intelligent" thought process required to select the most suitable candidate algorithms based on high level attributes of the problem at hand.

For a higher level discussion on current trends in numerical methods, the papers presented by the invited speakers are most applicable. These present support for the view that the next generation of scientific software systems will be "intelligent" and that existing technology is able to provide the required functionality.

Though the topics appear to be rather limited, these are covered in a wide variety of approaches, views and opinions. The majority of the papers are well presented, though they are terse and very summarised in places. This is understandable, given that they are technical papers, originally to be presented to colleagues well versed in the topics. This book is a worthwhile addition to the local expert systems or numerical methods research library.

Laz Davila
Computer Sciences of Australia


The Japanese after World War II were probably the driving force pushing the rest of the world into seeking more efficient means of quality production. The Western world responded by seeking to enhance production using computer technology as a basis. Systems for controlling machine tools, materials handling and integrating design with the manufacturing process soon attained
workable forms, and parallel with these developments, more sophisticated computer packages were developed for every conceivable aspect of factory design and management. The work was normally undertaken in Universities, and side by side with searches for tools to replicate human intelligence. The author of this book has gathered together a wide range of examples of computer packages developed to help industry and has reviewed their structure and application. Sadly however the composition of the book also illustrates the reasons that the majority of packages have been found wanting. This point is not referred to directly but it might have been better illustrated had one or two realistic examples of manufacturing been referred to. However this does not detract from its quality as a well researched and compendious account of industrial computer systems development.

A strange aspect of the book is its avoidance of the real world of manufacturing and, for a book so deeply concerned with communication and decision making it is strange that no overall representation (such as the Ottawa Reference Model) had been set up to allow communication paths and decision making areas in a factory to be visualised for the different systems outlined in the book. The author has concentrated on programmes involving knowledge bases and elements of artificial intelligence. The author comments that the latter is particularly important in Decision Support Systems because of its emphasis on representation of problem spaces in ways which map easily on to the forms of representation naturally used by the human brain.

Whilst there are differences in understandings of what constitutes artificial intelligence, it is anticipated that the book will find acceptance by many academics seeking a text to fill out courses in Manufacturing Management. Sadly, however, there has to be some doubt about the extent to which the adoption of the techniques outlined would benefit many companies in Australia with industry in its present stage of development.

Jacob A. Cartmel
Engitech Pty Ltd


Have you ever considered reading a dictionary from cover to cover? No? It is unlikely then that you would consider reading a computer dictionary this way. However, you can gain much from such an exercise, provided that it is a well written dictionary. This dictionary can be read this way.

There are, however, inherent problems in compiling any dictionary because of the amount of information available. The subject areas to be covered must be selected and then limited to a finite number of terms. As well, a market for the dictionary needs to exist. A weakness of this dictionary is that it covers many subject areas, resulting in a lack of depth in any one. It is a relatively small book (only 250 pages out of its 300 pages contain definitions), compared to other specialised dictionaries on my bookshelf: a data communications dictionary (over 500 pages) and a computer user's dictionary (550 pages).

The first thing noticed about the dictionary is its presentation. The pages are uncluttered, the font selection and use of diagrams are visually appealing, and where possible, terms are cross referenced allowing you to gain more information. Different naming conventions for the same topic are included, for example, BACKUS NAUR form and BACKUS normal form (BNF). Abbreviations are catered for, which means that if you only know the mnemonic, the definition can still be found. For example, the entry "DDL" is cross-referenced to the entry "data definition language DDL". The definitions of terms are written in a clear concise manner, with examples being used where further clarification is needed. The subject areas covered include Artificial Intelligence and Expert Systems, Databases, Data Structures/Computer Science, Information Retrieval, Bibliometrics/Infometrics, Information Science, and Communication and Hardware.

The dictionary also contains two appendices. The first is a subject outline, showing a breakdown of the terms used in the dictionary by subject. A particular term can then be seen in its context, for example the term "blackboard" appears under a sub-heading "Tools" which is part of the subject area

"Expert Systems" which is a sub-heading of "Artificial Intelligence and Expert Systems". The second appendix is a list of references to related academic papers and books, which provides a greater depth of knowledge than is possible to present in a dictionary. Most terms defined in the dictionary have two entries listed against them, referring to these two appendices.

This dictionary would be of benefit to anyone wanting a general overview of this subject matter. The inclusion of the two appendices is a major strength of the book, an idea that I have not seen before in a dictionary. I personally would not buy this dictionary because its subject matter is not relevant to my daily work. However, I would have found it most beneficial in my student days, especially for a subject such as information retrieval.

Tony Stevenson
Melbourne


As one who came to the Information Technology Industry via engineering, and who believes that computers are valuable only if they facilitate practical outcomes, why does this reviewer find a book written to determine "whether these computer graphics systems really perform the task initially conceived for them" vaguely unsatisfying? "Are they of benefit to design? Or are they merely image creators?" asked Professor Medland when he first wrote The Computer-based Design Process in 1985. These questions must have been rhetorical. Why else would he be pursuing the development of the ideal CAD system seven years on?

Perhaps my dissatisfaction in reading this book is his reflected. The Holy Grail remains undiscovered. Tantalisingly closer than it was seven years ago, the ultimate CAD solution still eludes the author. While it may be the nature of people such as Professor Medland, and it may be as well for scientific progress that they don't stop for self-congratulation, nevertheless, The Computer-based Design Process chronicles a series of significant discoveries and achievements.

The development of Computer-aided design and manufacture (CAD/CAM) has been vigorous over the past 25 years. It has come as a response to the manufacturing industry's drive to optimise its design activities and translate from plan to product with minimum effort. A strength of this book is the clarity of presentation of a massive subject area. The development of the design process is covered, avoiding the detail, but leaving a clear understanding of the concepts. The problems faced by designers are presented and explained, and although, in reality, they can only have been solved by long and persistent effort, the significance of the accomplishment is not diminished by the brevity of the telling. Professor Medland has the knack of carrying his reader along with him. (One can understand why his research team have been so faithful).

Throughout the book the recurrent question is how the design process can optimise the host of constraints which are imposed. The geometry of the design presents constraints; the functionality of the product introduces other constraints; and manufacturing or cost constraints overlay the entire process. Since this is the heart of the problem, it is appropriate that the second half of The Computer-based Design Process is devoted to modelling techniques and the management of constraints. The last two of the eight chapters are new to this Second Edition. It is in the last that the most recent problem-solving techniques are introduced, setting the course perhaps for the next seven years.

This is a book written by an enthusiast. Essentially it is a primer for those who want to know about CAD from its origins to present day. Anyone looking for a depth of understanding of the engineering or mathematics involved will not find it here, although the details are likely to be in one of the references provided in the extensive bibliography. However there is a sub-text of desperation: find the solution before industry loses patience.

Roger Price
State Electricity Commission of Victoria
**NEWS BRIEFS**

'News Briefs' is a regular feature which covers local and overseas developments in the computer industry including new products and other topical events of interest.

**UNE's $544,000 Computer Research Project**

The provision of a group of highly sophisticated computers valued at $544,000 to the University of New England's Department of Mathematics, Statistics and Computing Science may have a huge impact on the future transmission of video, graphics and pictures via world computer links.

The equipment comes from Digital Equipment Corporation (Australia) Pty Ltd and provides the department with an outstanding opportunity to enhance its research reputation in this rapidly developing area of science.

Researches will seek to compress the data and to establish an efficient and cost effective way of storing and transmitting it. If successful, their findings will have a dramatic impact in the area of business and commerce especially in media organisations.

At present these organisations use telephone links which are costly and slow.

The contracts for the project were signed at Armidale by the Principal of UNE-Armidale, Professor Cliff Hawkins and Digital's Director of External Research Program, Mr Fred King.

The Head of the Department of Mathematics and Computing Science, Dr Chris Radford, said the contract will provide a tremendous boost to the Department's reputation. He said it provided the University with a great opportunity to show its wares on the world stage.

The equipment from Digital will be provided through its External Research Program and will involve the provision of four sophisticated multi-media computers which are the first of their type in Australia.

There will also be a server, a very fast network system and some of the most advanced software in the world.

The research team will be led by Associate Professor Patrick Lenders and Dr Simantello, believes the growing recognition and acceptance of Natural ReSource sees the West Australian software development company strongly positioned to take advantage of the multi-million dollar market created by the new programming standard.

**ACCOUNTING SYSTEM**

An Australian-made range of financial and distribution systems, Chairman, is fast making inroads into the international market and, with a June 1993 launch of Chairman Open System Interface (OSI) in the USA, will be one of the world's first corporate accounting systems to be produced with Windows.

"Chairman will be one of only four or five companies in the world to offer corporate accounting users the added functionality and user appeal of 'Windows'," said David Jensen, managing director, Chairman.

The software that Chairman is developing will bring all the user-friendly advantages of personal computers into the large corporate accounting world. Chairman OSI will greatly enhance the functionality of the mini- and main-frame environments.

Chairman has been rewritten to take advantage of the latest technology and efficiencies in using Client Server architectures. A user will be able to install Chairman not only as a server to a central UNIX machine, but also on PC clients using Windows 3.1 or NT.

Chairman OSI has been specifically designed to reduce traffic on the network and utilise the processing power of the front-end systems.

The new product's Graphical User Interface will allow use of such tools as mice, fonts, colour control, rollerballs and cursor keys to position around the screen and edit text fields quickly and easily.

The system will use buttons for fast access to functions and tools, search buttons to access records in the database, scroll bars to view text and data without keystrokes, and pull-down menus to add functionality without exiting the screen.

Chairman OSI will run over all the major hardware environments — IBM RS6000, HP, Digital, Fujitsu etc — on databases such as Progress, Oracle, and Sybase, and with DOS, OS/2, UNIX, VMS and OS900 operating systems.

The system's basic concept is one of creating input data, posting it, and reporting on the ledgers — making the system very easy to understand and use.

The suite includes general ledger, debtors, creditors, inventory, assets, student administration, payroll/personnel and project costing to form a fully integrated system.

Chairman is favoured across a diverse range of industries, from liquor distribution, abattoirs and rendering plants, steel manufacturers, banking and finance, computer and automotive retailers, to government departments and educational institutions.

It is the chosen software of the Institute of Chartered Accountants in England and Wales and the New Zealand Police Force amongst many others.

**NEW STANDARD ASSISTS SOFTWARE DEVELOPERS**

Software developers with international links may now make use of a newly-published Australian standard, Guidelines for the application of ISO 9001 to the development, supply and maintenance of software (AS 3900.3).

The new standard, developed by the Standards Australia, has been published by Standards Australia to meet the needs of those organizations dealing with overseas clients who require compliance with the widely recognized ISO 9001 quality management standard. It is designed to assist software developers in Australia achieve certification to ISO 9001 and, in doing so, will support trade in leading locally designed and produced software.

The publication of the international document as an Australian standard does not, however, affect the importance of the existing Australian standard, Software quality management systems (AS 3563). The difference between the two documents is that the international standard provides guidelines only to achieving certification to ISO 9001, whereas AS 3563 is a document against which developers' quality systems may be audited and certified.

Companies operating in the international marketplace, then, should consider seeking the wider recognition of certification to ISO 9001. However, companies developing a quality management system may find it more...
straightforward to address the software-specific and more practical requirements of AS 1011. The certificate, which provides the benefits of both standards, is probably the most beneficial approach.

Both ISO 9001 and Part 1 of AS 3563 contain very similar requirements, and companies certified to one should find it relatively simple to achieve certification to the other. Certification to both standards concurrently is probably simpler than successive certification audits.

By publishing the international standard providing guidance for software developers implementing ISO 9001 as an Australian standard, Standards Australia has expanded the options for Australian companies in the international marketplace, while retaining the proven and practical approach of AS 3563.

Copies of Guidelines for the application of ISO 9001 to the development, supply and maintenance of software (AS 3900.3) may be purchased from any office of Standards Australia, or by contacting the National Sales Centre, ph (02) 746 4600.

NEW INFORMATION TECHNOLOGY AND COMPUTER STANDARDS

Standards Australia has recently published the following standards:

AS 3081 Telecommunications installations — Twisted pair cables for telecommunications applications, Part 2: Cables for connection to carrier networks. This standard defines requirements for metallic twisted pair cables to be used in customer premises installations (indoors and outdoors) where there is the intention to interconnect these with telecommunications carrier networks. Electrical characteristics and performance parameters are specified, as are special requirements for arial and underground cables. Two minimum performance levels are included, one to cater for voice and low speed data services and the other to cater for the requirements for primary rate ISDN access.

AS 3082 Telecommunications installations — Optical fibre cables for telecommunications applications, Part 2: Cables for connection to carrier networks. This standard defines requirements for optical fibre cables to be used in customer premises installations (indoors and outdoors) where there is the intention to interconnect these with telecommunications carrier networks. Fibre characteristics, coated fibre requirements and sheath properties are specified, as the physical properties and transmission performance for complete cables.

AS 3083 Telecommunications installations — Coaxial cables for telecommunications applications, Part 2: Cables for connection to carrier networks, defines requirements for coaxial cables to be used in customer premises installations (indoors and outdoors) where there is the intention to interconnect these telecommunications carrier networks. Physical and electrical requirements are specified and attention is drawn to observance of mechanical aspects during installation.

AS 3904 Quality management and quality systems elements Part 3 (Int): Guidelines for processed materials. This interim standard provides a basic guide to quality system elements applicable to processed materials, such as bulk products and discusses means of ensuring effective quality management. It is based on and similar in scope to AS 3904.1/ISO 9004.1 Guidelines, and is identical with and produced from ISO/DIS 9004-3:1993. It is also identical with NZS 9004.3 (Int):1993 and is produced as a joint Australian/New Zealand standard.

Part 4 (Int): Guidelines for quality improvement. This interim standard provides a set of management guidelines for implementing continuous quality improvement methodology based on data collection and analysis. This standard is identical with and produced from ISO/DIS 9004-4:1993, is identical with NZS 9004.4 (Int):1993 and is produced as a joint Australian/New Zealand standard.

AS/NZS 4802 Information processing systems — Local area networks Part 7: Slotted ring access method and physical layer specification, specifies the requirements for a local area network that utilizes the slotted ring access method, and operates at 10 Mbit/s data rate. This standard is identical with, and has been reproduced from, ISO 8802–7:1991, and is produced as a joint Australian/New Zealand standard.

AS/NZS 1011 Information processing systems — Unrecorded 12.7 mm (0.5 in) wide magnetic tape for information interchange — 32 fppm (800 fpi), NRZ1, 126 fppm (3,800 fpi) phase encoded and 356 fppm (9,042 fpi), NRZ2, specifies the characteristics of 12.7 mm (0.5 in) wide magnetic tape with reel to enable magnetic and mechanical interchangeability of such tape between information processing systems. This standard is identical with, and has been reproduced from, ISO/IEC 1864:1992 and is produced as a joint Australian/New Zealand standard. It supersedes the 1986 edition of AS 1011.

AS/NZS 4109 Information technology — Text and office systems — Distributed-office-applications model Part 1: General model, provides a framework to the purpose of the development of protocol standards for distributed-office-applications. It applies to applications distributed over significant physical distances as well as ‘close-coupled’ office systems. This standard is identical with and has been reproduced from ISO/IEC 10031-1:1991 and is produced as a joint Australian/New Zealand standard.

Part 2: Distinguished-object-reference and associated procedures, defines the elements used in the specification of the distinguished-object-reference for use within distributed-office-applications. These elements cover the description of distinguished-object-reference and associated procedures, together with a functional model and the distinguished-object-reference structure and abstract syntax. This standard is identical with and has been reproduced from ISO/IEC 10031-2:1991 and is produced as a joint Australian/New Zealand standard.

AS/NZS 4110 Information technology — SGML and text-entry systems — Guidelines for SGML syntax-directed editing systems, describes a set of functions which an SGML syntax-directed editing system may have in order to help users manipulate documents marked up according to the standard of SGML. This standard is identical with and has been reproduced from ISO/IEC TR 10037:1991 and is produced as a joint Australian/New Zealand standard.

AS/NZS 4116 Information processing systems — Data interchange on 90 mm flexible magnetic tape, provides the benefits of both standards laid down by the co-operative Systems Analysis and Design project of 38 tracks on each side — ISO Type 303, specifies the characteristics of 90 mm flexible disk cartridges recorded at 381 fpprad using modified frequency modulation recording, on 80 tracks on each side. This standard is identical with, and has been reproduced from, ISO/IEC 10994:1992 and is produced as a joint Australian/New Zealand standard.

These standards may be purchased from any office of Standards Australia, or by contacting the National Sales Centre, ph (02) 746 4600.

THE 25,000TH SAD CERTIFICATE

A special presentation to mark the occasion of the 25,000th NCC Certificate in Systems Analysis and Design took place at the ISEB Headquarters in London on 12 May 1993. The 25,000th certificate was presented to Miss Ann Colley who is employed by the Ministry of Defence.

In 1966 the National Computing Centre (NCC) produced a report which drew attention to the serious shortage of systems analysts and the lack of acceptable training courses. This resulted in the formation of an NCC working party, representing industry and education, which defined the contents of a suitable course. The Systems Analysis Examination Board, now named the Information Systems Examinations Board (ISEB), was formed jointly by NCC and BSC to certify successful candidates. The first certificates of this type were awarded in 1967 triggered by the NCC report. The certificate is now regulated by ISEB, whose directors represent NCC, BCS, COSTIT, CCTA and the Training Agency.

The Certificate has become recognised, worldwide, as the leading certificate for IT professionals. It is only awarded to people who successfully complete an approved course of study covering all aspects of systems analysis and design. The syllabus has been continually updated to incorporate changes in technology, the role of the analyst and the development of analysis and design methodologies.

During the 25 years that the Certificate has been awarded, NCC has provided relevant training courses to meet the demanding level of training required. Training materials have been produced that are fully documented and enable other organisations to adopt the standards laid down by the syllabus. For employers the certificate demonstrates that the holder has achieved an internationally recognised level of competence. For organisations offering courses leading to the certificate it shows that their training meets a recognised standard.